title: Optimistic Rollups description: An introduction to optimistic rollups—a scaling solution used by the Ethereum community. lang: en

Optimistic rollups are layer 2 (L2) protocols designed to extend the throughput of Ethereum's base layer. They reduce computation on the main Ethereum chain by processing transactions off-chain, offering significant improvements in processing speeds. Unlike other scaling solutions, such as sidechains, optimistic rollups derive security from Mainnet by publishing transaction results on-chain, or plasma chains, which also verify transactions on Ethereum with fraud proofs, but store transaction data elsewhere.

As computation is the slow, expensive part of using Ethereum, optimistic rollups can offer up to 10-100x improvements in scalability. Optimistic rollups also write transactions to Ethereum as `calldata`, reducing gas costs for users.

Prerequisites {#prerequisites}

You should have read and understood our pages on Ethereum scaling and layer 2.

What is an optimistic rollup? {#what-is-an-optimistic-rollup}

An optimistic rollup is an approach to scaling Ethereum that involves moving computation and state storage off-chain. Optimistic rollups execute transactions outside of Ethereum, but post transaction data to Mainnet as `calldata`.

Optimistic rollup operators bundle multiple off-chain transactions together in large batches before submitting to Ethereum. This approach enables spreading fixed costs across multiple transactions in each batch, reducing fees for end-users. Optimistic rollups also use compression techniques to reduce the amount of data posted on Ethereum.

Optimistic rollups are considered “optimistic” because they assume off-chain transactions are valid and don't publish proofs of validity for transaction batches posted on-chain. This separates optimistic rollups from zero-knowledge rollups that publish cryptographic proofs of validity for off-chain transactions.

Optimistic rollups instead rely on a fraud-proving scheme to detect cases where transactions are not calculated correctly. After a rollup batch is submitted on Ethereum, there's a time window (called a challenge period) during which anyone can challenge the results of a rollup transaction by computing a fraud proof.

If the fraud proof succeeds, the rollup protocol re-executes the transaction(s) and updates the rollup's state accordingly. The other effect of a successful fraud proof is that the sequencer responsible for including the incorrectly executed transaction in a block receives a penalty.

If the rollup batch remains unchallenged (i.e., all transactions are correctly executed) after the challenge period elapses, it is deemed valid and accepted on Ethereum. Others can continue to build on an unconfirmed rollup block, but with a caveat: transaction results will be reversed if based on an incorrectly executed transaction published previously.

How do optimistic rollups interact with Ethereum? {#optimistic-rollups-and-Ethereum}

Optimistic rollups are off-chain scaling solutions built to operate on top of Ethereum. Each optimistic rollup is managed by a set of smart contracts deployed on the Ethereum network. Optimistic rollups process transactions off the main Ethereum chain, but post off-chain transactions (in batches) to an on-chain rollup contract. Like the Ethereum blockchain, this transaction record is immutable and forms the "optimistic rollup chain."

The architecture of an optimistic rollup comprises the following parts:

On-chain contracts: The optimistic rollups's operation is controlled by smart contracts running on Ethereum. This includes contracts that store rollup blocks, monitor state updates on the rollup, and track user deposits. In this sense, Ethereum serves as the base layer or "layer 1" for optimistic rollups.

Off-chain virtual machine (VM): Although contracts managing the optimistic rollup protocol run on Ethereum, the rollup protocol performs computation and state storage on another virtual machine separate from the Ethereum Virtual Machine. The off-chain

VM is where applications live and state changes are executed; it serves as the upper layer or "layer 2" for an optimistic rollup.

As optimistic rollups are designed to run programs either written or compiled for the EVM, the off-chain VM incorporates many EVM design specs. Additionally, fraud proofs computed on-chain allows the Ethereum network to enforce the validity of state changes computed in the off-chain VM.

Optimistic rollups are described as 'hybrid scaling solutions' because, while they exist as separate protocols, their security properties are derived from Ethereum. Among other things, Ethereum guarantees the correctness of a rollup’s off-chain computation and the availability of data behind the computation. This makes optimistic rollups more secure than pure off-chain scaling protocols (e.g., sidechains) that do not rely on Ethereum for security.

Optimistic rollups rely on the main Ethereum protocol for the following:

Data availability {#data-availability}

As mentioned, optimistic rollups post transaction data to Ethereum as`calldata`. Since the rollup chain's execution is based on submitted transactions, anyone can use this information—anchored on Ethereum’s base layer—to execute the rollup’s state and verify the correctness of state transitions.

Data availability is critical because without access to state data, challengers cannot construct fraud proofs to dispute invalid rollup operations. With Ethereum providing data availability, the risk of rollup operators getting away with malicious acts (e.g., submitting invalid blocks) is reduced.

Censorship resistance {#censorship-resistance}

Optimistic rollups also rely on Ethereum for censorship resistance. In an optimistic rollup a centralized entity (the operator) is responsible for processing transactions and submitting rollup blocks to Ethereum. This has some implications:

- Rollup operators can censor users by going offline completely, or by refusing to either produce blocks that include certain transactions in them.

- Rollup operators can prevent users from withdrawing funds deposited in the rollup contract by withholding state data necessary to Merkle proofs of ownership. Withholding state data can also conceal the rollup’s state from users and prevent them from interacting with the rollup.

Optimistic rollups solve this problem by forcing operators to publish data associated with state updates on Ethereum. Publishing rollup data on-chain has the following benefits:

- If an optimistic rollup operator goes offline or stops producing transaction batches, another node can use available data to reproduce the rollup’s last state and continue block production.

- Users can use transaction data to create Merkle proofs proving ownership of funds and withdraw their assets from the rollup.

- Users can also submit their transactions on L1 instead of to the sequencer, in which case the sequencer has to include the transaction within a certain time limit to continue to produce valid blocks.

Settlement {#settlement}

Another role Ethereum plays in the context of optimistic rollups is that of a settlement layer. A settlement layer anchors the entire blockchain ecosystem, establishes security, and provides objective finality if a dispute occurs on another chain (optimistic rollups in this case) that requires arbitration.

Ethereum Mainnet provides a hub for optimistic rollups to verify fraud proofs and resolve disputes. Moreover, transactions conducted on the rollup are only final *after* the rollup block is accepted on Ethereum. Once a rollup transaction is committed to Ethereum’s base layer, it cannot be rolled back (except in the highly unlikely case of a chain reorganization).

How do optimistic rollups work? {#how-optimistic-rollups-work}

Transaction execution and aggregation {#transaction-execution-and-aggregation}

Users submit transactions to “operators”, which are nodes responsible for processing transactions on the optimistic rollup. Also known as a “validator” or “aggregator”, the operator aggregates transactions, compresses the underlying data, and publishes the block on Ethereum.

Although anyone can become a validator, optimistic rollup validators must provide a bond before producing blocks, much like a proof-of-stake system. This bond can be slashed if the validator posts an invalid block or builds on an old-but-invalid block (even if their block is valid). This way optimistic rollups utilize cryptoeconomic incentives to ensure validators act honestly.

Other validators on the optimistic rollup chain are expected to execute the submitted transactions using their copy of the rollup’s state. If a validator’s final state is different from the operator’s proposed state, they can start a challenge and compute a fraud proof.

Some optimistic rollups may forgo a permissionless validator system and use a single “sequencer” to execute the chain. Like a validator, the sequencer processes transactions, produces rollup blocks, and submits rollup transactions to the L1 chain (Ethereum).

The sequencer is different from a regular rollup operator because they have greater control over the ordering of transactions. Also, the sequencer has priority access to the rollup chain and is the only entity authorized to submit transactions to the on-chain contract. Transactions from non-sequencer nodes or regular users are simply queued up in a separate inbox until the sequencer includes them in a new batch.

Submitting rollup blocks to Ethereum {#submitting-blocks-to-ethereum}

As mentioned, the operator of an optimistic rollup bundles off-chain transactions into a batch and sends it to Ethereum for notarization. This process involves compressing transaction-related data and publishing it on Ethereum as `calldata`.

`calldata` is a non-modifiable, non-persistent area in a smart contract that behaves mostly likememory. While `calldata` persists on-chain as part of the blockchain's history logs, it is not stored as a part of Ethereum's state. Because`calldata` does not touch any part of Ethereum's state, it is cheaper for storing data on-chain.

The `calldata` keyword is also used in Solidity to pass arguments to a smart contract function at execution time.`calldata` identifies the function being called during a transaction and holds inputs to the function in the form of an arbitrary sequence of bytes.

In the context of optimistic rollups, `calldata` is used to send compressed transaction data to the on-chain contract. The rollup operator adds a new batch by calling the required function in the rollup contract and passing the compressed data as function arguments. Using `calldata` reduces user fees since most costs that rollups incur come from storing data on-chain.

Here is an example of a rollup batch submission to show how this concept works. The sequencer invoked the `appendSequencerBatch()` method and passed the compressed transaction data as inputs using`calldata`.

State commitments {#state-commitments}

At any point in time, the optimistic rollup’s state (accounts, balances, contract code, etc.) is organized as aMerkle tree called a “state tree”. The root of this Merkle tree (state root), which references the rollup’s latest state, is hashed and stored in the rollup contract. Every state transition on the chain produces a new rollup state, which an operator commits to by computing a new state root.

The operator is required to submit both old state roots and new state roots when posting batches. If the old state root matches the existing state root in the on-chain contract, the latter is discarded and replaced with the new state root.

The rollup operator is also required to commit a Merkle root for the transaction batch itself. This allows anyone to prove the inclusion of a transaction in the batch (on L1) by presenting a Merkle proof.

State commitments, especially state roots, are necessary for proving the correctness of state changes in an optimistic rollup. The rollup contract accepts new state roots from operators immediately after they are posted, but can later delete invalid state roots to restore the rollup to its correct state.

Fraud proving {#fraud-proving}

As explained, optimistic rollups allow anyone to publish blocks without providing proofs of validity. However, to ensure the chain remains safe, optimistic rollups specify a time window during which anyone can dispute a state transition. Hence, rollup blocks are called “assertions” since anyone can dispute their validity.

If someone disputes an assertion, then the rollup protocol will initiate the fraud proof computation. Every type of fraud proof is interactive—someone must post an assertion before another person can challenge it. The difference lies in how many rounds of interaction are required to compute the fraud proof.

Single-round interactive proving schemes replay disputed transactions on L1 to detect invalid assertions. The rollup protocol emulates the re-execution of the disputed transaction on L1 (Ethereum) using a verifier contract, with the computed state root determining who wins the challenge. If the challenger's claim about the rollup’s correct state is correct, the operator is penalized by having their bond slashed.

However, re-executing transactions on L1 to detect fraud requires publishing state commitments for individual transactions and increases the data rollups must publish on-chain. Replaying transactions also incurs significant gas costs. For these reasons, optimistic rollups are switching to multi-round interactive proving, which achieves the same objective (i.e., detecting invalid rollup operations) with more efficiency.

Multi-round interactive proving {#multi-round-interactive-proving}

Multi-round interactive proving involves a back-and-forth protocol between the asserter and challenger overseen by an L1 verifier contract, which ultimately decides the lying party. After an L2 node challenges an assertion, the asserter is required to divide the disputed assertion into two equal halves. Each individual assertion in this case will contain as many steps of computation as the other.

The challenger will then choose what assertion it wants to challenge. The dividing process (called a “bisection protocol”) continues until both parties are disputing an assertion about a *single* step of execution. At this point, the L1 contract will resolve the dispute by evaluating the instruction (and its result) to catch the fraudulent party.

The asserter is required to provide a “one-step proof” verifying the validity of the disputed single-step computation. If the asserter fails to provide the one-step proof, or the L1 verifier deems the proof invalid, they lose the challenge.

Some notes about this type of fraud proof:

1. Multi-round interactive fraud proving is considered efficient because it minimizes the work the L1 chain must do in dispute arbitration. Instead of replaying the entire transaction, the L1 chain only needs to re-execute one step in the rollup's execution.

2. Bisection protocols reduce the amount of data posted on-chain (no need to publish state commits for every transaction). Also, optimistic rollup transactions are not constrained by Ethereum's gas limit. Conversely, optimistic rollups re-executing transactions must make sure an L2 transaction has a lower gas limit to emulate its execution within a single Ethereum transaction.

3. Part of the malicious asserter's bond is awarded to the challenger, while the other part is burned. The burning prevents collusion among validators; if two validators collude to initiate bogus challenges, they will still forfeit a considerable chunk of the entire stake.

4. Multi-round interactive proving requires both parties (the asserter and the challenger) to make moves within the specified time window. Failure to act before the deadline expires causes the defaulting party to forfeit the challenge.

Why fraud proofs matter for optimistic rollups {#fraud-proof-benefits}

Fraud proofs are important because they facilitate *trustless finality* in optimistic rollups. Trustless finality is a quality of optimistic rollups that guarantees that a transaction—so long as it’s valid—will eventually be confirmed.

Malicious nodes can try to delay the confirmation of a valid rollup block by starting false challenges. However, fraud proofs will eventually prove the rollup block’s validity and cause it to be confirmed.

This also relates to another security property of optimistic rollups: the validity of the chain relies on the existence of *one* honest node. The honest node can advance the chain correctly by either posting valid assertions or disputing invalid assertions. Whatever the case, malicious nodes who enter into disputes with the honest node will lose their stakes during the fraud proving process.

L1/L2 interoperability {#l1-l2-interoperability}

Optimistic rollups are designed for interoperability with Ethereum Mainnet and allow users to pass messages and arbitrary data between L1 and L2. They are also compatible with the EVM, so you can port existing dapps to optimistic rollups or create new dapps using Ethereum development tools.

1. Asset movement {#asset-movement}

Entering the rollup

To use an optimistic rollup, users deposit ETH, ERC-20 tokens, and other accepted assets in the rollup'sbridge contract on L1. The bridge contract will relay the transaction to L2, where an equivalent amount of assets is minted and sent to the user's chosen address on the optimistic rollup.

User-generated transactions (like an L1 > L2 deposit) are usually queued until the sequencer re-submits them to the rollup contract. However, to preserve censorship resistance, optimistic rollups allow users to submit a transaction directly to the on-chain rollup contract if it has been delayed past the maximum time allowed.

Some optimistic rollups adopt a more straightforward approach to prevent sequencers from censoring users. Here, a block is defined by all transactions submitted to the L1 contract since the previous block (e.g., deposits) in addition to the transactions processed on the rollup chain. If a sequencer ignores an L1 transaction, it will publish the (provably) wrong state root; therefore, sequencers cannot delay user-generated messages once posted on L1.

Exiting the rollup

Withdrawing from an optimistic rollup to Ethereum is more difficult owing to the fraud proving scheme. If a user initiates an L2 > L1 transaction to withdraw funds escrowed on L1, they must wait until the challenge period—lasting roughly seven days—elapses. Nevertheless, the withdrawal process itself is fairly straightforward.

After the withdrawal request is initiated on the L2 rollup, the transaction is included in the next batch, while the user's assets on the rollup are burned. Once the batch is published on Ethereum, the user can compute a Merkle proof verifying the inclusion of their exit transaction in the block. Then it is a matter of waiting through the delay period to finalize the transaction on L1 and withdraw funds to Mainnet.

To avoid waiting a week before withdrawing funds to Ethereum, optimistic rollup users can employ a**liquidity provider** (LP). A liquidity provider assumes ownership of a pending L2 withdrawal and pays the user on L1 (in exchange for a fee).

Liquidity providers can check the validity of the user's withdrawal request (by executing the chain themselves) before releasing funds. This way they have assurances that the transaction will be confirmed eventually (i.e., trustless finality).

2. EVM compatibility {#evm-compatibility}

For developers, the advantage of optimistic rollups is their compatibility—or, better still, equivalence—with theEthereum Virtual Machine (EVM). EVM-compatible rollups comply with specifications in theEthereum Yellow Paper and support the EVM at the bytecode level.

EVM-compatibility in optimistic rollups has the following benefits:

i. Developers can migrate existing smart contracts on Ethereum to optimistic rollup chains without having to modify codebases extensively. This can save development teams time when deploying Ethereum smart contracts on L2.

ii. Developers and project teams using optimistic rollups can take advantage of Ethereum's infrastructure. This includes programming languages, code libraries, testing tools, client software, deployment infrastructure, and so on.

Using existing tooling is important because these tools have been extensively audited, debugged, and improved over the years. It also removes the need for Ethereum developers to learn how to build with an entirely new development stack.

3. Cross-chain contract calls {#cross-chain-contract-calls}

Users (externally owned accounts) interact with L2 contracts by submitting a transaction to the rollup contract or having a sequencer or validator do it for them. Optimistic rollups also allow contract accounts on Ethereum to interact with L2 contracts using bridging contracts to relay messages and pass data between L1 and L2. This means you can program an L1 contract on Ethereum Mainnet to invoke functions belonging to contracts on an L2 optimistic rollup.

Cross-chain contract calls happen asynchronously—meaning the call is initiated first, then executed at a later time. This is different from calls between the two contracts on Ethereum, where the call produces results immediately.

An example of a cross-chain contract call is the token deposit described earlier. A contract on L1 escrows the user's tokens and sends a message to a paired L2 contract to mint an equal amount of tokens on the rollup.

As cross-chain message calls result in contract execution, the sender is usually required to cover gas costs for computation. It is advisable to set a high gas limit to prevent the transaction from failing on the target chain. The token bridging scenario is a good example; if the L1 side of the transaction (depositing the tokens) works, but the L2 side (minting new tokens) fails due to low gas, the deposit becomes irrecoverable.

Finally, we should note that L2 > L1 message calls between contracts need to account for delays (L1 > L2 calls are typically executed after some minutes). This is because messages sent to Mainnet from the optimistic rollup cannot be executed until the challenge window expires.

How do optimistic rollup fees work? {#how-do-optimistic-rollup-fees-work}

Optimistic rollups use a gas fee scheme, much like Ethereum, to denote how much users pay per transaction. Fees charged on optimistic rollups depends on the following components:

1. **State write**: Optimistic rollups publish transaction data and block headers (consisting of the previous block header hash, state root, batch root) to Ethereum as `calldata`. The minimum cost of an Ethereum transaction is 21,000 gas. Optimistic rollups can reduce the cost of writing the transaction to L1 by batching multiple transactions in a single block (which amortizes the 21k gas over multiple user transactions).

2. **`calldata`**: Beyond the base transaction fee, the cost of each state write depends on the size of `calldata` posted to L1. `calldata` costs are currently governed by EIP-1559, which stipulates a cost of 16 gas for non-zero bytes and 4 gas for zero bytes of `calldata`, respectively. To reduce user fees, rollup operators compress transactions to reduce the number of `calldata` bytes published on Ethereum.

3. **L2 operator fees**: This is the amount paid to the rollup nodes as compensation for computational costs incurred in processing transactions, much like gas fees on Ethereum. Rollup nodes charge lower transaction fees since L2s have higher processing capacities and aren't faced with the network congestions that force validators on Ethereum to prioritize transactions with higher fees.

Optimistic rollups apply several mechanisms to reducing fees for users, including batching transactions and compressing `calldata` to reduce data publication costs. You can check the L2 fee tracker for a real-time overview of how much it costs to use Ethereum-based optimistic rollups.

How do optimistic rollups scale Ethereum? {#scaling-ethereum-with-optimistic-rollups}

As explained, optimistic rollups publish compressed transaction data on Ethereum to guarantee data availability. The ability to compress data published on-chain is crucial to scaling throughput on Ethereum with optimistic rollups.

The main Ethereum chain places limits on how much data blocks can hold, denominated in gas units (the average block size is 15 million gas). While this restricts how much gas each transaction can use, it also means we can increase transactions processed per block by reducing transaction-related data—directly improving scalability.

Optimistic rollups use several techniques to achieve transaction data compression and improve TPS rates. For example, this article compares the data a basic user transaction (sending ether) generates on Mainnet vs how much data the same transaction generates on a rollup:

Parameter	Ethereum (L1)	Rollup (L2)
Nonce	~3	0
Gasprice	~8	0-0.5
Gas	3	0-0.5
To	21	4
Value	9	~3
Signature	~68 (2 + 33 + 33)	~0.5
From	0 (recovered from sig)	4
Total	**~112 bytes**	**~12 bytes**

Doing some rough calculations on these figures can help show the scalability improvements afforded by an optimistic rollup:

1. The target size for every block is 15 million gas and it costs 16 gas to verify one byte of data. Dividing the average block size by 16 gas (15,000,000/16) shows the average block can hold **937,500 bytes of data**.
2. If a basic rollup transaction uses 12 bytes, then the average Ethereum block can process**78,125 rollup transactions** (937,5000/12) or **39 rollup batches** (if each batch holds an average of 2,000 transactions).
3. If a new block is produced on Ethereum every 15 seconds, then the rollup's processing speeds would amount to roughly **5,208 transactions per second**. This is done by dividing the number of basic rollup transactions an Ethereum block can hold (**78,125**) by the average block time (**15 seconds**).

This is a fairly optimistic estimate, given that optimistic rollup transactions cannot possibly comprise an entire block on Ethereum. However, it can give a rough idea of how much scalability gains that optimistic rollups can afford Ethereum users (current implementations offer up to 2,000 TPS).

The introduction of data sharding on Ethereum is expected to improve scalability in optimistic rollups. Because rollup transactions must share blockspace with other non-rollup transactions, their processing capacity is limited by data throughput on the main Ethereum chain. Danksharding will increase the space available to L2 chains to publish data per block, using cheaper, impermanent "blob" storage instead of expensive, permanent `CALLDATA`.

Pros and cons of optimistic rollups {#optimistic-rollups-pros-and-cons}

Pros	Cons
Offers massive improvements in scalability without sacrificing security or trustlessness.	Delays in transaction finality due to potential fraud challenges.
Transaction data is stored on the layer 1 chain, improving transparency, security, censorship-resistance, and decentralization.	Centralized rollup operators (sequencers) can influence transaction ordering.
Fraud proving guarantees trustless finality and allows honest minorities to secure the chain.	If there are no honest nodes a malicious operator can steal funds by posting invalid blocks and state commitments.
Computing fraud proofs is open to regular L2 node, unlike validity proofs (used in ZK-rollups) that require special hardware.	Security model relies on at least one honest node executing rollup transactions and submitting fraud proofs to challenge invalid state transitions.
Rollups benefit from "trustless liveness" (anyone can force the chain to advance by executing transactions and posting assertions)	Users must wait for the one-week challenge period to expire before withdrawing funds back to Ethereum.
Optimistic rollups rely on well-designed cryptoeconomic incentives to increase security on the chain.	Rollups must post all transaction data on-chain, which can increase costs.
Compatibility with EVM and Solidity allows developers to port Ethereum-native smart contracts to rollups or use existing tooling to create new dapps.	

A visual explanation of optimistic rollups {#optimistic-video}

More of a visual learner? Watch Finematics explain optimistic rollups:

Use Optimistic rollups {#use-optimistic-rollups}

Multiple implementations of Optimistic rollups exist that you can integrate into your dapps:

Further reading on optimistic rollups

- How do optimistic rollups work (The Complete guide)
- Everything you need to know about Optimistic Rollup
- The Essential Guide to Arbitrum
- How does Optimism's Rollup really work?
- OVM Deep Dive
- What is the Optimistic Virtual Machine?