Data availability is a critical consideration for blockchain scaling designs; rollups are the preferred scaling solution for Ethereum today because availability of data required to reconstruct the L2 state is guaranteed by the L1 network. But data availability—especially in the context of rollups—has many subtle, easy-to-miss details that are nevertheless important for security.

With “L2 summer” finally happening, and modular data availability layers moving from theory to production, educating the community on the implications of rollups’ approach to data availability for security and decentralization has become more important than ever. It is especially important for users to know how differences in where transaction data is stored determines if a rollup is “secured by Ethereum” or not.

As a contribution to this effort, I put together a comprehensive article that covers the topic of data availability from first principles, discussing the (in)famous “data availability problem” in Layer 1 (L1) and Layer 2 (L2) blockchains—before exploring the different approaches rollup blockchains adopt to guarantee data availability. To summarize, data availability policies in rollups can currently be grouped into three major groups:

1. Publishing transaction data on Ethereum vs. using a bespoke data availability layer
2. Publishing data on-chain vs. using data availability committees (DACs)
3. Publishing raw transaction data on-chain vs. publishing state diffs on-chain

Each data availability policy comes with unique tradeoffs and advantages. For example, publishing full transaction data guarantees a high level of security for a rollup, as Ethereum (L1) guarantees the rollup’s liveness

(honest rollup nodes can produce new blocks if the sequencer is accidentally/deliberately offline, and users can create Merkle proofs to prove balances to the rollup’s bridge contract and trustlessly withdraw funds from the L2) and safety

(optimistic rollups operate with a 1-of-N

security model that requires at least one honest node to re-execute transactions and challenge invalid updates to the L1’s view of the L2 state—this requires that honest nodes should be able to derive L2 blocks solely by reading the L1 chain without relying on another party (e.g., the sequencer) to provide data to recreate state transitions).

However, Ethereum limits block sizes (to improve decentralization), which hinders rollup scalability (size of L2 batches must be constrained—especially since rollup transactions must compete with other L1 transactions for (limited) blockspace on Ethereum) and increases operating costs for sequencers (which indirectly increases fees charged to users since the costs are passed to transaction senders—[calldata

costs currently make up a large part of the transaction fee mechanisms in Ethereum rollups today] (https://forum.celestia.org/t/ethereum-rollup-call-data-pricing-analysis/141)).

These factors make external data availability services, which are able to scale horizontally—for example, by implementing data availability sampling so that light nodes can contribute to the network’s throughput and security—attractive for newer rollup designs. Moreover, modular data availability layers can (theoretically) have bigger blocks compared to Ethereum as computational resources are not shared between execution and state storage.

However, posting rollup data to an external data availability service means a rollup can no longer be completely secured by Ethereum. For example, collusion by a malicious set of nodes on the data availability service can degrade liveness (for both optimistic and validity rollups) and safety (for optimistic rollups; since challengers cannot independently verify state updates if the sequencer/validator, in collusion with nodes on the data availability network, hides transactions data for new blocks (i.e., block bodies), but proposes block headers for finalization on Ethereum).

Ethereum rollups also have different choices for storing state data on Ethereum: (1) Publish raw transaction data, i.e., all inputs required to (re-)execute one or more rollup transactions in compressed form

on L1 (2) Publish state diffs, i.e., outputs of transactions that modify the rollup’s state trie.

The former is a default option for optimistic rollups since transaction batches (and state roots computed by executing batched transactions) do not carry any proof attesting to correct execution. The latter, however, is available to zero-knowledge (ZK) rollups, a.k.a., validity rollups, that submit transaction batches and state roots along with validity (SNARK/STARK) proofs attesting to the integrity of each state transition (and thus do not have to publish all data required to re-execute transactions).

Option (1)

has more benefits than merely guaranteeing an ORU’s safety—as I note in a section of the article comparing the two aforementioned policies, publishing full transaction data provides a trustless source of a rollup’s history of transactions from genesis. This is useful for many applications, including (but not limited to): DEXes, lending markets, blockchain forensics, cross-chain bridges, DAOs, and NFT marketplaces. Publishing transaction inputs also reduces delay for users waiting to know if a transaction succeeded (and was included a block) or not, as users can re-execute a batch once it is posted on L1 and deterministically ascertain the result of one or more transactions.

Option (2)

reduces on-chain footprint for rollups, especially as state diffs can omit extra data (e.g., signatures), and amortize costs for applications with high-frequency updates (e.g., oracles) by publishing only the last update to a contract's storage on-chain and leaving out intermediary transactions that modified the same storage slot(s). Option (2) reduces overall transaction fees (due to lower per-transaction costs for L1 calldata) compared to option (1)—but results in (potentially) longer finality delays for application users and slightly weaker guarantees of protection from a censoring sequencer (e.g., users cannot force a sequencer to include a transaction). Option (2) has the benefits mentioned previously (full transaction history, (potentially) faster finality guarantees, and stronger anti-censorship guarantees—especially with a force inclusion mechanism enabled), at the cost of higher fees owing to higher use of L1 storage. (L2 Fees' "L1 Security Costs" dashboard illustrates this relationship with real-world stats.)

It is important to note (as I do in the article) that there is no "Holy Grail" for data availability policies in rollups—ideally, rollups would publish full transaction data on Ethereum L1 and constrain any trust assumptions relevant to the rollup's security (liveness and safety) to the base layer. But certain applications may want to deploy on a rollup integrated with an external data availability service or a DAC, especially if (a) the value of transactions is low-to-moderate (e.g., GameFi applications and decentralized social networks) (b) users can tolerate higher trust assumptions in exchange for lower transactions costs and faster execution (e.g., on-chain derivatives and NFT marketplaces), and (c) higher requirements for transaction privacy necessitate keeping transaction data off-chain (e.g., to prevent other traders from copying trading strategies on a DEX).

The key idea, I believe, is for users and developers to understand the implications of a rollup's data availability policy and for rollup projects—or, more broadly, off-chain scaling protocols—to communicate various trust assumptions underpinning approaches to data availability in the interests of transparency. With the latter in mind, I dedicated other sections of the article to explore:

-
    - The relationship between economic security of a data availability layer and the security guarantees it provides to rollups using it.
- The relationship between economic security of a data availability layer and the security guarantees it provides to rollups using it.
-
    - Drawbacks and challenges around using alternative data availability layers, rather than Ethereum, to store rollup data.
- Drawbacks and challenges around using alternative data availability layers, rather than Ethereum, to store rollup data.
-
    - Different approaches to storing L2 data on Ethereum and implications for end users
- Different approaches to storing L2 data on Ethereum and implications for end users

All feedback and comments are welcome. I'd also like to know what others in the community think of the movement towards alternate data availability layers, and how that impacts the future of Ethereum's rollup-centric roadmap.

PS: Also adding some recent tweets from the L2Beat team for context on emerging issues around rollups on Ethereum integrating with modular data availability layers:

- Thread by @donnoh: https://x.com/donnoh_eth/status/1736792589741736290?s=20
- Thread by @Bartek

: https://x.com/bkiepuszewski/status/1738153297146495198?s=20

- Thread by @Bartek

: https://x.com/bkiepuszewski/status/1737422769338421519?s=20

- Thread by @Bartek

: https://x.com/bkiepuszewski/status/1736798073358364973?s=20

P.P.S.: Adding a great post from @barrywhitehat on understanding security properties of sidechains since it highlights key details around the relationship between data availability and security in off-chain scaling protocols