#15 Layer-2 Scaling

Lecture Notes for CS190N: Blockchain Technologies and Security November 24, 2025

This lecture introduces Ethereum's rollup-centric scaling roadmap and explains why scalability has become the central constraint for public blockchains. We first quantify the performance and fee limits of Layer 1 (L1), then motivate modular designs that keep consensus and data availability on L1 while offloading execution to Layer 2 (L2) rollups. Building a concrete mental model of rollup architecture and funds flow, we track deposits, transactions, batching, and withdrawals through the L1 contract, Sequencer, and off-chain state. Finally, we compare Optimistic Rollups that rely on fraud proofs and challenge windows with Zero-Knowledge rollups that use validity proofs for fast finality, highlighting how these choices shape trust assumptions, user experience, and capital efficiency in a rollup-centric Ethereum.

1 WHY SCALING IS NEEDED (MOTIVATION)

1.1 The Performance Ceiling of Layer 1

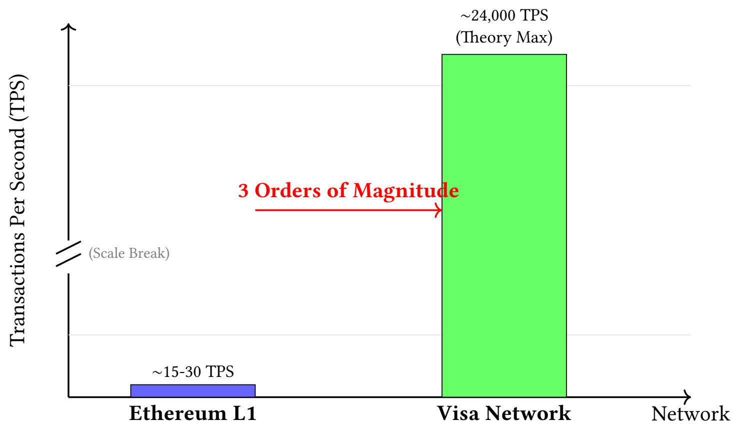

Figure 1. The Throughput Gap: Ethereum vs. Traditional Payments

The core promise of blockchain technology is the creation of a trustless, decentralized global settlement layer. However, the architectural decisions required to achieve decentralization impose severe penalties on performance. Ethereum, the dominant smart contract platform, operates under a set of constraints designed to ensure that a consumer-grade laptop can run a node and verify the chain. This requirement is non-negotiable for maintaining censorship resistance; if running a node required a massive data center, the network would effectively be controlled by a few corporate entities.

The cost of this decentralization is redundancy. In a standard Layer 1 blockchain, every full node must re-execute every transaction to verify the state transition. If Alice sends 1 ETH to Bob, thousands of computers across the globe must independently update their ledgers to reflect this change. While this provides robust security, it is inherently inefficient. Consequently, Ethereum in its current form processes approximately 15 to 30 transactions per second (TPS).

To contextualize this limitation, we must look at the systems Ethereum aims to displace or augment. The Visa payment network, a centralized standard for global commerce, is capable of processing up to 24,000 TPS [5]. This represents a discrepancy of three orders of magnitude (see Figure 1). Furthermore, modern financial markets generate transaction volumes that dwarf even payment networks; the Nasdaq or the New York Stock Exchange process millions of orders per second. For DeFi to become a viable alternative to traditional finance, supporting not just simple transfers but complex interactions like algorithmic trading, high-frequency market making, and micropayments, the underlying throughput capacity must increase exponentially.

1.2 The Economic Consequences of Congestion

The bottleneck at Layer 1 is not just a matter of speed; it is fundamentally a matter of cost. Ethereum blocks have a finite amount of space, measured in "gas." When the demand for block space exceeds the available supply, the network relies on a fee market to prioritize transactions. Users essentially enter an auction, bidding higher gas fees to incentivize validators to include their transactions in the next block.

This mechanism leads to extreme volatility in transaction costs. During periods of high network utilization, such as the "DeFi Summer" of 2020 or the NFT boom of 2021, gas fees spiked to exorbitant levels. It was not uncommon for a simple token swap on Uniswap to cost upwards of $50 to $100 in fees. Even in more recent times, volatility persists. While average fees have moderated, spikes can still render the network unusable for smaller transactions.

Consider the implications of a $50 transaction fee. For a "whale" moving $1,000,000, a $50 fee is a rounding error, representing 0.005% of the principal. However, for a user attempting to send $100 to a family member or buy a $5 coffee, a $50 fee is prohibitive. This creates a regressive economic structure where the "public" blockchain becomes a gated community accessible only to the wealthy. If the objective of cryptocurrency is financial inclusion, a system where users must pay $40 to move $20 is a failure.

Recent data from late 2025 indicates that while Layer 1 fees have stabilized somewhat due to the migration of activity to Layer 2, the base layer remains fundamentally constrained. A simple ETH transfer might cost $0.38 [7], but a complex smart contract interaction can still cost significantly more depending on network congestion. This persistent cost floor necessitates alternative scaling strategies.

1.3 The Scaling Solution Space: On-Chain vs. Off-Chain

Faced with this throughput limit, architects have explored two primary avenues for scaling: on-chain and off-chain solutions.

On-Chain Scaling (Layer 1 upgrades): This approach involves modifying the base protocol to process more transactions.

- *Larger Blocks:* One simple method is to increase the block size or gas limit, allowing more transactions per block. However, this increases the hardware requirements for nodes (more disk space, bandwidth, and CPU), which centralizes the network.
- *Sharding:* A more complex approach involves splitting the blockchain into multiple parallel chains, or "shards," each responsible for a subset of the state. While Ethereum originally planned to shard execution, the roadmap has pivoted. Full execution sharding introduces massive complexity regarding cross-shard communication and security. The current roadmap focuses on "Danksharding," [4] which shards *data availability* rather than execution, enabling the network to store more data for Layer 2s without necessarily processing more transactions itself.

Off-Chain Scaling (Layer 2): The industry has overwhelmingly converged on this second path. The core philosophy here is modularity. Instead of forcing the Layer 1 blockchain to handle consensus, data availability, and execution, we unbundle these functions.

- *Layer 1 (Ethereum):* Acts as the settlement layer. It provides security, consensus, and data availability. It serves as the ultimate arbiter of truth, a supreme court that settles disputes but does not hear every case.
- *Layer 2 (Rollups):* Handles execution. It processes transactions, maintains the application state, and performs the heavy computational lifting off-chain. It then periodically reports the results back to Layer 1.

This lecture focuses exclusively on **Rollups**, which have emerged as the dominant Layer 2 architecture. A rollup is effectively a separate blockchain that runs "on top" of Ethereum, inheriting its security guarantees while offering vastly higher throughput and lower costs.

2 ROLLUP BASICS: ARCHITECTURE AND FUNDS FLOW

2.1 Defining the Rollup

At a conceptual level, a rollup is an off-chain execution environment that bundles, or "rolls up", hundreds or thousands of transactions into a single batch. It then posts a compressed summary of these transactions and the resulting state root back to the Layer 1 blockchain.

This architecture achieves scaling through two mechanisms:

(1) **Off-Chain Execution:** The actual computation (verifying signatures, updating balances, executing smart contract logic) happens on a high-performance server off-chain, not on the congested Ethereum network.
(2) **Data Compression:** Instead of sending full transaction details to the L1, the rollup sends highly compressed data. For example, a standard Ethereum transaction requires $\sim$ 110 bytes. A rollup can compress this to $\sim$ 12 bytes by stripping out signatures (which are verified off-chain) and using short indices for addresses [1].

Definition: A Rollup is a scaling solution where transaction execution occurs outside the main chain (Layer 1), but transaction data and state commitments are posted to the main chain. This ensures that the Layer 1 chain guarantees the availability of data and the correctness of the state.

2.2 Core Architectural Components

To understand how a rollup functions, we must identify its three primary components: the L1 smart contract, the off-chain state, and the Sequencer.

(1) **The L1 Rollup Contract:** This is a smart contract deployed on the Ethereum mainnet. It serves as the "anchor" for the entire system. Its responsibilities are twofold:
    - *Asset Custody:* It acts as a lockbox for all funds deposited into the rollup. When you move ETH to Arbitrum or Optimism, you are actually sending it to this contract on Ethereum.
    - *State Commitment:* It stores a sequential history of "State Roots." A state root is a cryptographic hash (usually a Merkle Root) that represents the entire state of the rollup (all account balances and contract storage) at a specific block height.
(2) **The Off-Chain State (Layer 2):** This is the database of accounts and balances that exists on the rollup. It is structurally similar to Ethereum's state (often a Merkle Tree) but lives on independent servers. It tracks who owns what *inside* the rollup ecosystem.
(3) **The Sequencer (Coordinator):** The Sequencer is a specialized node responsible for operating the rollup. In most current implementations (like Arbitrum One, Base, and Optimism), the Sequencer is a centralized server run by the project team, though plans for decentralization are active. The Sequencer's duties include:
    - Receiving transaction requests from users.
    - Ordering these transactions (First-In-First-Out or via priority fees).

- Executing the transactions to update the L2 state.
- Compressing the transaction data and submitting it, along with the new state root, to the L1 Rollup Contract.

2.3 The Lifecycle of Funds: A Concrete Scenario

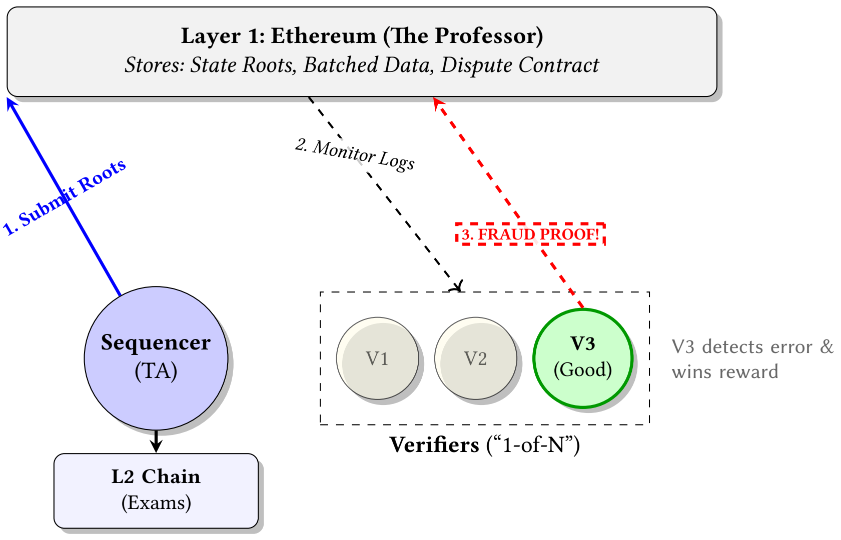

Figure 2. Optimistic Rollup Ecosystem: The "1-of-N" Security Model

To demystify the interaction between these components, let us trace the flow of capital as a user, Alice, moves funds into and out of a rollup (illustrated in Figure 3).

2.3.1 Phase 1: Deposit (L1 $\rightarrow$ L2). Alice begins with 2 ETH on the Ethereum mainnet and wishes to use a DeFi application on a rollup to save on fees.

(1) **Initiation:** Alice sends a standard Ethereum transaction to the **L1 Rollup Contract**, attaching 2 ETH.
(2) **L1 Locking:** The L1 contract receives the 2 ETH and locks them. It cannot spend these funds; it simply holds them in custody. Crucially, the contract emits a `Deposit` event in the Ethereum logs containing Alice's address and the amount.
(3) **L2 Minting:** The Sequencer is constantly monitoring the L1 blockchain for these specific events. Upon detecting the `Deposit` event, the Sequencer creates a special "minting" transaction on the Layer 2 chain.
(4) **State Update:** This L2 transaction credits Alice's L2 address with 2 ETH. Note that these are "wrapped" or "representation" tokens on L2, fully backed by the physical ETH locked in the L1 contract.
(5) **Result:** Alice now possesses 2 ETH on the rollup. She can execute trades, lend assets, or transfer funds to other users like Bob. These subsequent L2 transactions occur instantly and cost fractions of a cent because they only involve the Sequencer updating its local database.

2.3.2 Phase 2: Transact (Execution). Alice decides to transfer 1 ETH to Bob within the rollup.

(1) **L2 Transfer:** Alice signs a transaction `transfer(Bob, 1 ETH)` and sends it to the Sequencer.

(2) **Instant Soft Finality:** The Sequencer checks Alice's balance, confirms she has the funds, and immediately updates the local state: Alice now has 1 ETH, Bob has 1 ETH. The Sequencer confirms to Alice that the transaction is successful. This happens in milliseconds, providing a "Web2-like" user experience.

2.3.3 Phase 3: Settlement (Batching). The Sequencer does not immediately run to Layer 1 with each single transaction.

(1) **Batch Compression:** It waits to collect thousands of transactions from users. It compresses this batch to minimize data size.
(2) **L1 Submission:** It submits the compressed transaction data and the new state roots to the L1 Rollup Contract in a single Ethereum transaction. This step (labeled "Submit Roots & Data" in Figure 3) amortizes the heavy L1 gas cost across all users in the batch, resulting in the massive fee reduction we observe on Layer 2.

2.3.4 Phase 4: Withdrawal (L2 → L1). Alice decides to move her remaining 1 ETH back to the Ethereum mainnet.

(1) **Initiation:** Alice submits a specific `Withdraw` transaction on the rollup.
(2) **L2 Burning:** The Sequencer processes this request by removing (burning) 1 ETH from Alice's L2 balance. It effectively takes the ETH out of circulation on the L2 side.
(3) **L1 Commitment:** The Sequencer includes this withdrawal request in the batch it posts to the L1 contract.
(4) **The Verification Gap:** This is the critical moment. The L1 contract sees a request to unlock 1 ETH for Alice. However, the L1 contract cannot blindly trust the Sequencer. If the Sequencer were malicious, it could submit a fake withdrawal request claiming Alice owns 1,000 ETH. The L1 contract needs proof that the withdrawal is legitimate.
    - The mechanism for providing this proof differs fundamentally between Optimistic and zk-Rollups, defining their respective trade-offs.

2.4 Categorizing Rollups: Optimistic vs. Zero-Knowledge

As hinted in the withdrawal phase, the defining characteristic of a rollup is not how it executes transactions, but how it proves to Layer 1 that those transactions were correct. This "verification gap" has led to two dominant architectural approaches, which we will explore in depth in the following sections:

(1) **Optimistic Rollups (e.g., Arbitrum, Optimism):** These protocols take a "innocent until proven guilty" approach. They assume all transaction batches posted by the Sequencer are valid by default. The L1 contract only intervenes if a watcher detects fraud and submits a proof within a specific challenge window (typically 7 days).
(2) **Zero-Knowledge (zk) Rollups (e.g., zkSync, Starknet):** These protocols take a "don't trust, verify" approach. They require the Sequencer to generate a cryptographic proof (Validity Proof) for every batch. The L1 contract verifies this proof mathematically before accepting the state update, ensuring immediate finality without a challenge period.

3 OPTIMISTIC ROLLUPS

3.1 Core Philosophy: "Innocent Until Proven Guilty"

Optimistic Rollups (such as Arbitrum One, Optimism Mainnet, and Base) are built on a presumption of honesty. The protocol operates "optimistically," assuming that the transactions and state roots submitted by the Sequencer are correct. The L1 contract does not verify the mathematics of every transaction upon submission. Instead, it simply accepts the Sequencer's update and posts it to the blockchain.

However, this trust is not absolute. It is provisional. When a batch is posted, it enters a **Challenge Window**, typically lasting 7 days [3]. During this window, the state is not considered final. It is pending. The system relies on a network of "Verifiers" (or "Watchtowers") to keep the Sequencer honest, as depicted in Figure 2.

3.2 Security Model: The Fraud Proof Game

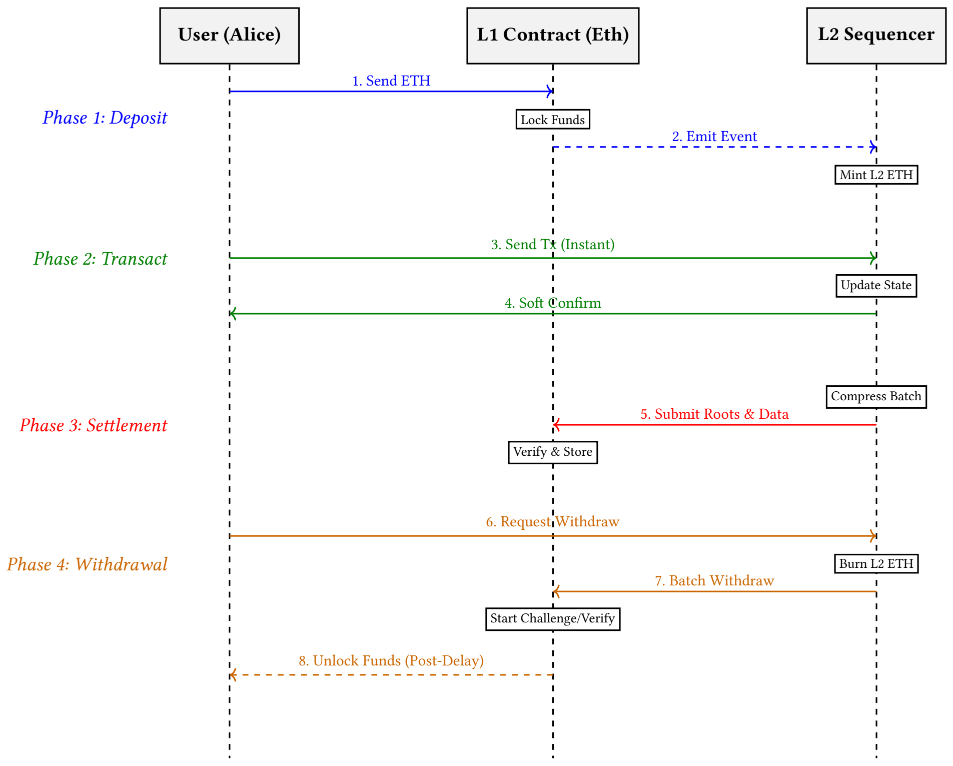

Figure 3. Lifecycle of Funds: From Deposit to Withdrawal

The security of an Optimistic Rollup is underpinned by **Fraud Proofs** (also known as Fault Proofs). This is a reactive security mechanism. The L1 contract acts as a supreme court that only hears cases when a dispute arises.

The Dispute Resolution Process.

(1) **Observation:** Verifiers (who can be anyone running a node) constantly download the data posted by the Sequencer to L1 and re-execute the transactions locally. They check if their resulting state root matches the one published by the Sequencer.
(2) **The Challenge:** If a Verifier detects a discrepancy, for example, the Sequencer claims the new root is `0xABC` but the Verifier calculates `0xXYZ`, they issue a challenge to the L1 contract. This locks the disputed state.
(3) **The Interactive Bisection Game:** To resolve the dispute efficiently, the Sequencer and the Verifier engage in an interactive back-and-forth protocol (Figure 4). It would be too

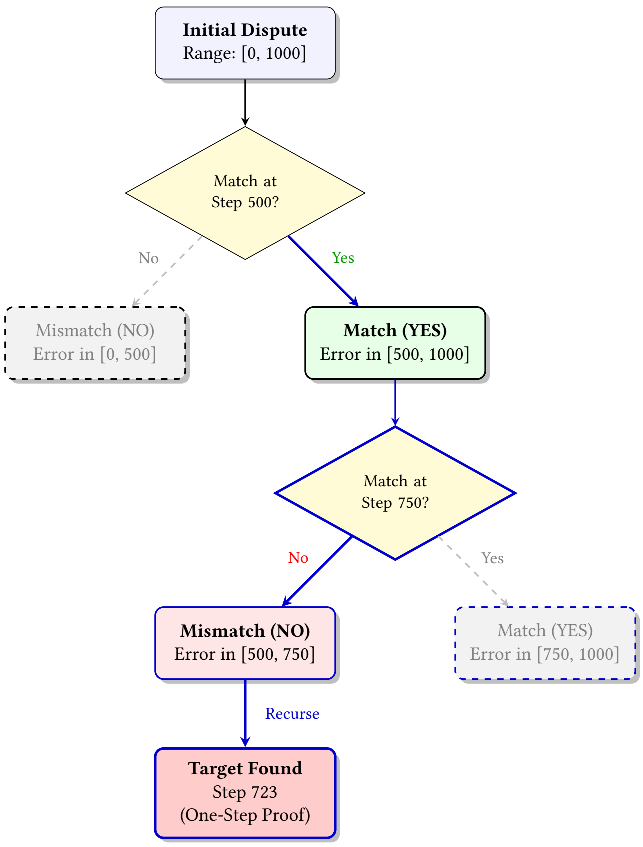
Initial Dispute
Range: [0, 1000]
Match at Step 500?
No
Yes
Mismatch (NO) Error in [0, 500]
Match (YES) Error in [500, 1000]
Match at Step 750?
No
Yes
Mismatch (NO) Error in [500, 750]
Match (YES) Error in [750, 1000]
Recurse
Target Found
Step 723
(One-Step Proof)

Figure 4. Visualizing the Fraud Proof "Interactive Bisection Game" (Tree View)

expensive for the L1 contract to re-execute the entire batch of thousands of transactions to see who is right. Instead, the protocol forces the two parties to narrow down their disagreement.

- They bisect the execution trace: "Do we agree on the state at the halfway point?"
- "Yes." ⟶ "Okay, do we agree at the 75% mark?"
- "No." ⟶ "Okay, the error is in the third quarter."
- This continues until they identify the exact **single instruction** (a single opcode) where their results diverge.

(4) **One-Step Proof:** Once the dispute is narrowed to a single calculation step, the L1 contract executes *just that one step* on-chain. This is computationally cheap. The L1 contract compares its result with the claims.

(5) **Slashing:**
- If the Verifier is correct (fraud occurred), the Sequencer's state update is rejected, the rollup state is reverted to the last valid point, and the Sequencer is "slashed", meaning their security bond (stake) is seized and a portion is given to the Verifier as a reward.
- If the Sequencer is correct (false alarm), the Verifier loses their bond.

The "1-of-N" Trust Assumption. This model relies on a "1-of-N" security assumption. As long as there is **at least one honest verifier** monitoring the chain, fraud will be detected and prevented. The system does not require a majority of honest participants, only a single whistleblower.

3.3 Analogy: The Exam Appeals Process

Imagine a university course with 500 students. The professor (Layer 1) is too busy to grade every exam question immediately. Instead, a Teaching Assistant (the Sequencer) grades all exams and posts the scores on a public bulletin board.

- **Optimistic Phase:** The university assumes the TA is honest. The grades are considered valid, but not yet transcript-official.
- **Challenge Window:** For one week, students (Verifiers) can review their exams.
- **Fraud Proof:** If a student finds a grading error, they don't ask the professor to re-grade the whole exam. They point to Question 3, part b. The professor looks *only* at that specific question. If the TA made a mistake, the grade is corrected, and the TA is fired (slashed). If no one complains for a week, the grades become permanent.

3.4 User Experience Implications

(1) **Low Cost:** Because the L1 contract does not perform heavy computation by default (it only verifies signatures and stores data), Optimistic Rollups are very cheap to operate. The cost is dominated by the data storage fees on L1 (calldata or blobs).
(2) **The 7-Day Withdrawal Delay:** The most significant downside of the optimistic model is the withdrawal latency. If Alice wants to move her funds back to Ethereum Layer 1, she cannot do so immediately. The L1 contract forces her to wait for the 7-day Challenge Window to elapse to ensure that the state update including her withdrawal is not fraudulent.
    - *Mitigation:* In practice, regular users rarely wait 7 days. Third-party **Liquidity Bridges** (like Across, Hop, or centralized exchanges) exploit the fact that they can verify the chain themselves. A bridge sees Alice's withdrawal, verifies it is valid, and instantly sends her funds on L1 (minus a small fee). The bridge then claims Alice's funds from the rollup after the 7-day wait. This effectively outsources the wait time to market makers with deep pockets.
(3) **EVM Equivalence:** Optimistic Rollups are structurally very similar to Ethereum. Because they don't need to generate complex cryptographic proofs, they can run the Ethereum Virtual Machine (EVM) almost exactly as is. This makes it incredibly easy for developers to deploy existing dApps (Uniswap, Aave) to Arbitrum or Optimism without code changes.

4 ZK ROLLUPS (ZERO-KNOWLEDGE ROLLUPS)

4.1 Core Philosophy: "Don't Trust, Verify"

Zero-Knowledge Rollups (such as zkSync Era, Starknet, Linea, and Scroll) reject the "innocent until proven guilty" paradigm. They operate on a principle of cryptographic certainty. The L1 contract does not trust the Sequencer for even a second. Instead, every time the Sequencer proposes a new batch of transactions, it must provide a **Validity Proof**, a cryptographic certificate that mathematically proves the correctness of the execution.

4.2 Security Model: Validity Proofs

The mechanism relies on advanced cryptography known as Zero-Knowledge Proofs (specifically SNARKs or STARKs).

The Verification Process.

(1) **Execution and Proving:** The Sequencer (or a specialized Prover node) executes the batch of transactions off-chain. It then performs a computationally intensive task: generating a mathematical proof.
   - This proof asserts: "I started with State Root A. I applied transactions T1 through T1000 according to the rules of the EVM. The result is State Root B."

(2) **Submission:** The Sequencer submits the transaction data, the new State Root, and the Validity Proof to the L1 smart contract.

(3) **On-Chain Verification:** The L1 contract contains a "Verifier Circuit." It runs a computation to verify the proof. Crucially, verifying a proof is exponentially cheaper and faster than generating it. It takes constant time regardless of how many transactions are in the batch.

(4) **Immediate Finality:** The L1 contract outputs a binary result: True or False.
   - If **True**, the state transition is accepted. The new state root is finalized immediately. There is no need to wait for challenges because it is cryptographically impossible (under current computational assumptions) to generate a valid proof for an invalid transaction.
   - If **False**, the transaction is rejected outright.

4.3 Analogy: The Locked Box and The Calculator

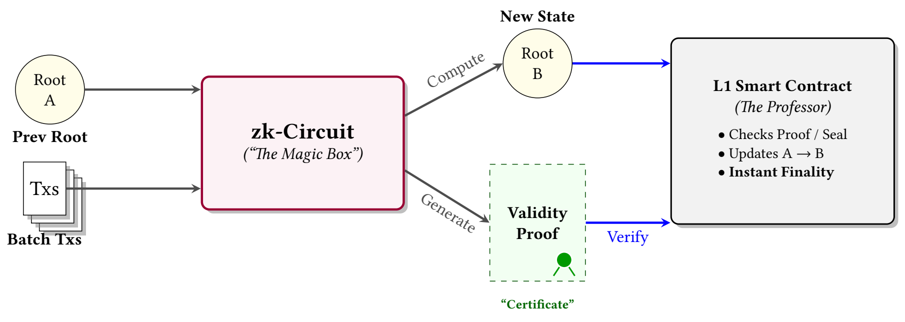

Figure 5. zk-Rollup Architecture: Inputs (Root A + Txs) → Proof → L1 Verification

Returning to our exam analogy:

- **zk Approach:** The TA (Sequencer) grades the exams. However, before submitting the grades to the Professor (Layer 1), the TA must feed the exams into a "Magical Glass Box" (the zk-Circuit, see Figure 5).
- Inside the box is a calculator that checks every step. If the grading is perfect, the box prints a "Certificate of Correctness" with a green seal.
- The TA hands the grades and the Certificate to the Professor. The Professor does not look at the exams. They simply check the green seal. If the seal is valid, the grades are accepted immediately. The Professor knows with 100% certainty that the grades are correct without ever reading them.

4.4 The Challenge of the zkEVM

For years, zk-Rollups were limited to simple payments (e.g., Loopring, early zkSync Lite) because generating validity proofs for the complex, Turing-complete EVM was considered nearly impossible. The EVM was not designed for zero-knowledge proofs; its opcode structure is difficult to translate into polynomial equations required for SNARKs.

However, recent breakthroughs have led to the development of **zkEVMs** (Zero-Knowledge Ethereum Virtual Machines). To navigate this complexity, Vitalik Buterin introduced a taxonomy of zkEVMs [2], categorizing them based on the trade-off between **Compatibility** (how easily existing Ethereum tools/contracts work) and **Performance** (how fast proofs are generated), as illustrated in Figure 6.

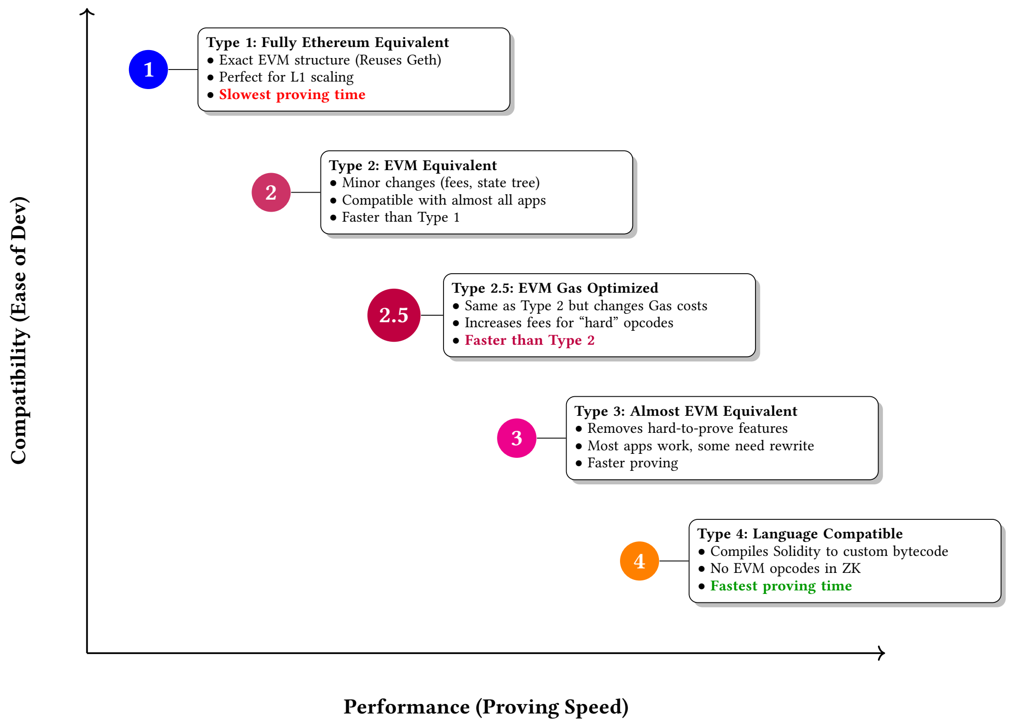
Compatibility (Ease of Dev)
Performance (Proving Speed)
1
Type 1: Fully Ethereum Equivalent
• Exact EVM structure (Reuses Geth)
• Perfect for L1 scaling
• Slowest proving time
2
Type 2: EVM Equivalent
• Minor changes (fees, state tree)
• Compatible with almost all apps
• Faster than Type 1
2.5
Type 2.5: EVM Gas Optimized
• Same as Type 2 but changes Gas costs
• Increases fees for "hard" opcodes
• Faster than Type 2
3
Type 3: Almost EVM Equivalent
• Removes hard-to-prove features
• Most apps work, some need rewrite
• Faster proving
4
Type 4: Language Compatible
• Compiles Solidity to custom bytecode
• No EVM opcodes in ZK
• Fastest proving time

Figure 6. Vitalik's zkEVM Taxonomy: The Compatibility vs. Performance Trade-off

- **Type 1 (Fully Ethereum-Equivalent):** These rollups strive to be indistinguishable from Ethereum itself. They reuse execution clients (like Geth) directly. The benefit is perfect compatibility (everything works out of the box), but the downside is extremely slow proof generation times (hours), as the EVM was not built for ZK.
- **Type 2 (EVM-Equivalent):** These make minor modifications to the EVM (e.g., changing hash functions) to make proof generation easier, while keeping the application layer 100% compatible.
- **Type 2.5 (EVM-Equivalent, Gas-optimized):** A practical optimization of Type 2. It remains EVM-equivalent but significantly increases the gas cost of specific operations that are computationally expensive to prove in ZK (like Keccak hashes). This discourages their use without breaking compatibility, offering a better performance-compatibility balance.
- **Type 3 (Almost EVM-Equivalent):** These remove features that are exceptionally hard to prove (like certain precompiles). Most applications work, but some may require rewriting.
- **Type 4 (High-Level Language Equivalent):** These projects (like zkSync Era and Starknet) take a source code (Solidity) and compile it into a completely different, ZK-friendly virtual machine bytecode. This abandons EVM equivalence for drastic performance gains. Proofs are fast and cheap, but some edge-case Solidity features might not be supported.

4.5 User Experience Implications

(1) **Fast Finality:** The defining advantage of zk-Rollups is speed of settlement. Once the proof is submitted and verified on L1 (which might take 15 minutes to an hour depending on batch frequency), the transaction is final. There is no 7-day challenge window.
(2) **Fast Withdrawals:** Because the L1 contract verifies the validity proof immediately, it knows the withdrawal request is legitimate. Alice can withdraw her funds from a zk-Rollup to Ethereum in roughly the time it takes to generate and verify a proof (typically less than an hour), rather than waiting a week. This capital efficiency is critical for institutional traders.
(3) **High Computational Cost:** Generating validity proofs requires immense computational power. Prover nodes often require server farms with high-end GPUs or FPGAs. This can make the "fixed cost" of operating a zk-Rollup higher than an Optimistic Rollup, although this cost is amortized as volume grows.
(4) **Data Compression:** zk-Rollups can actually be cheaper than Optimistic Rollups in terms of *L1 data costs*. Optimistic rollups must publish signatures and more data to allow verifiers to replay transactions. zk-Rollups can omit signatures and intermediate data because the Validity Proof already certifies that the signatures were checked off-chain. This allows for higher compression ratios.

5 OPTIMISTIC VS ZK ROLLUPS: TRADE-OFFS AND COMPARISON

The Layer 2 landscape in 2025 is defined by the competition between these two approaches. While Optimistic Rollups currently enjoy a "first-mover advantage" and dominate in Total Value Locked (TVL), zk-Rollups are rapidly maturing and are often viewed as the superior long-term solution due to their cryptographic guarantees.

5.1 Comparative Analysis

The following table (Table 1) synthesizes the key technical and economic differences between the two architectures:

5.2 Market Adoption and TVL (November 2025 Status)

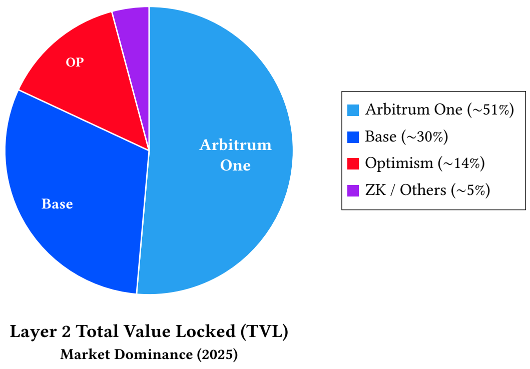

Figure 7. Dominance of Optimistic Rollups in the 2025 Market

As of November 2025, the market data reflects the maturity of Optimistic Rollups (see Figure 7) [6]:

- **Arbitrum One** leads the sector with approximately **$16.63 billion** in Total Value Locked (TVL), holding roughly 51% of the market share. Its first-mover advantage and deep liquidity have made it the default home for DeFi applications.

Feature	**Optimistic Rollups** (Arbitrum, Base, Optimism)	**zk Rollups** (zkSync Era, Starknet, Linea)
Security Mechanism	**Fraud Proofs:** Reactive security. Relies on game theory and the assumption that at least one honest watcher exists to challenge fraud.	**Validity Proofs:** Proactive security. Relies on mathematics (cryptography). The L1 rejects invalid states instantly.
Trust Assumption	**1-of-N Trust:** Honest verifier assumption. If all verifiers are censored or offline, fraud is possible.	**Trustless:** No human monitoring required for safety. Only requires the L1 to verify the math.
L1 Finality Time	**Slow (~7 days):** The challenge window must elapse before the state is irreversible.	**Fast (Minutes/Hours):** Final as soon as the proof is verified on L1.
Withdrawal Time	**~7 days:** Unless using a third-party bridge for fast exits (which incurs fees).	**~1 hour:** Funds can be moved back to L1 rapidly, improving capital efficiency.
Implementation	**Simpler:** Easier to achieve full EVM equivalence. Codebase is mature.	**Complex:** Requires sophisticated cryptography and massive engineering effort to build zkEVMs.
Operating Costs	**Low Compute, High Data:** Cheap to run nodes, but posts more data to L1.	**High Compute, Low Data:** Expensive proof generation, but posts less data (higher compression).
Market Status (2025)	**Dominant:** Highest TVL and transaction volume.	**Growing:** Gaining market share as technology matures.

Table 1. Optimistic vs. zk Rollup Comparison

- **Base**, the optimistic rollup incubated by Coinbase, has surged to roughly **$10-12 billion** TVL, driven by strong retail integration.
- **Optimism (OP Mainnet)** secures approximately **$6 billion**.

In contrast, zk-Rollups are growing but trailing in total capital:

- **Starknet** secures roughly **$1 billion**.
- **zkSync Era** and **Linea** hold between **$500 million and $1 billion** each.

This distribution highlights a key industry trend: **Technology is not the only driver of adoption.** Optimistic rollups launched earlier with better developer tooling (full EVM compatibility), allowing them to capture network effects. zk-Rollups are playing catch-up, but their superior security properties make them the likely endgame for the network.

6 SUMMARY OF KEY CONCEPTS

(1) **Scaling is an Existential Necessity:** Ethereum's L1 throughput of 15-30 TPS is insufficient for a global financial system. High fees exclude users and stifle innovation.
(2) **Rollups Decouple Execution from Security:** By moving computation to L2 while anchoring data on L1, rollups achieve high throughput without sacrificing the security guarantees of the base layer.
(3) **Optimistic Rollups (The Current Leader):** Utilize a "1-of-N" trust model with Fraud Proofs. They are easier to build and currently dominant (Arbitrum, Base) but suffer from a 7-day withdrawal delay, which degrades capital efficiency.
(4) **zk Rollups (The Future Challenger):** Utilize Validity Proofs for immediate cryptographic certainty. They offer fast finality and better data compression but face higher computational costs and engineering complexity.

REFERENCES

[1] Vitalik Buterin. 2021. An Incomplete Guide to Rollups. https://vitalik.eth.limo/general/2021/01/05/rollup.html. Accessed: 2025-11-24.

[2] Vitalik Buterin. 2022. The different types of ZK-EVMs. https://vitalik.eth.limo/general/2022/08/04/zkevm.html. Accessed: 2025-11-24.

[3] Arbitrum Foundation. 2025. Arbitrum Docs: Customizable challenge period. https://docs.arbitrum.io/launch-arbitrum-chain/configure-your-chain/common-configurations/customizable-challenge-period. Accessed: 2025-11-24.

[4] Ethereum Foundation. 2024. Danksharding | ethereum.org. https://ethereum.org/en/roadmap/danksharding/. Accessed: 2025-11-24.

[5] Visa Inc. 2019. Visa Fact Sheet. https://usa.visa.com/dam/VCOM/global/about-visa/documents/visa-fact-sheet-july-2019.pdf. Accessed: 2025-11-24.

[6] L2Beat. 2025. L2Beat: Ethereum Layer 2 Total Value Locked. https://l2beat.com/scaling/tvl. Accessed: 2025-11-24.

[7] L2Fees. 2025. L2Fees: Ethereum Layer 2 Fee Tracker. https://l2fees.info/. Accessed: 2025-11-24.