Lesson 7

Scalability

The scalability trilemma

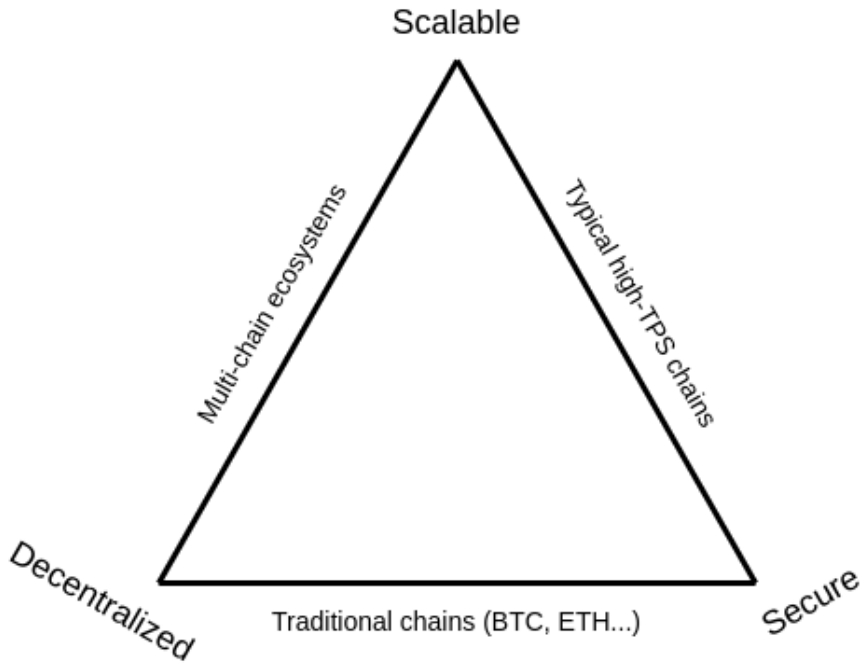


FIGURE 2. Taxonomy and comparison of blockchain scalability solutions.

From Scaling Blockchains: A Comprehensive Survey by Hafid et al.

"The decentralization of a system is determined by the ability of the weakest node in the network to verify the rules of the system." - Georgios Konstantopoulos

In Ethereum there is a goal to keep the hardware requirements low.
There is a useful guide to scalability in their documentation

Solutions

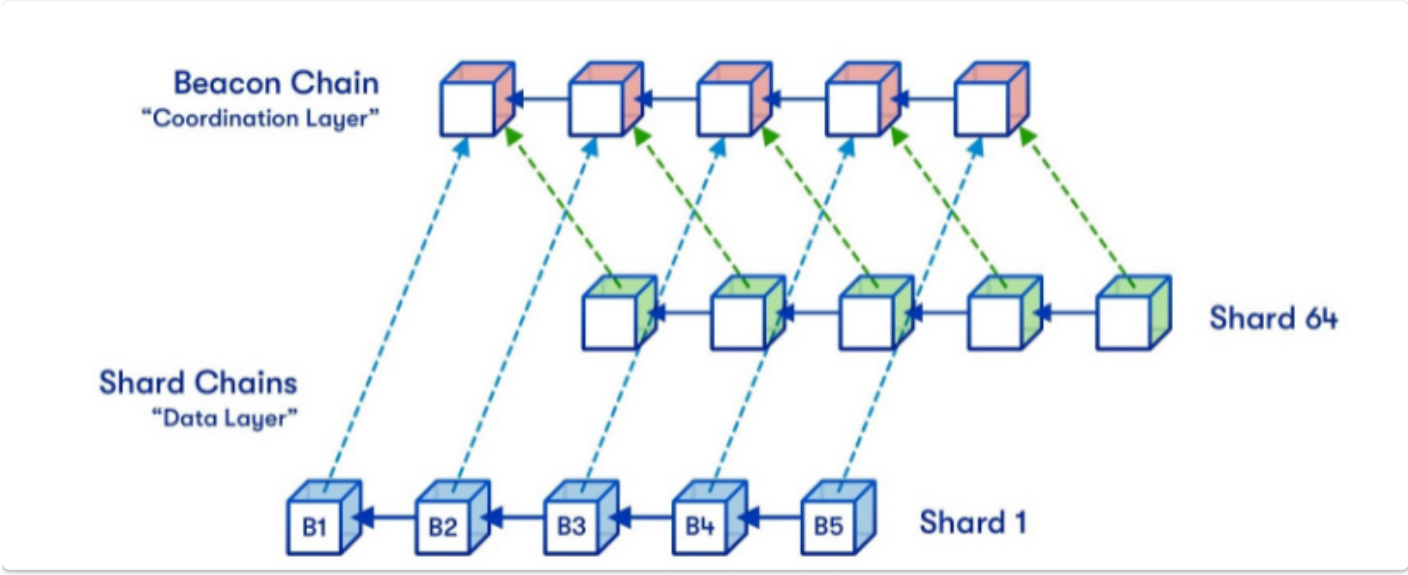
On chain Scaling (Layer 1)

Changing the Consensus Mechanism

Using DPoS - EOS
For example moving from Proof of Work to Proof of Stake - Ethereum
The Beacon chain is already live, in 2022 there will be the merge of main net and the beacon chain.

Sharding

Ethereum plans to introduce 64 new shard chains, to spread the network load.
Vitalik's overview
Introduction



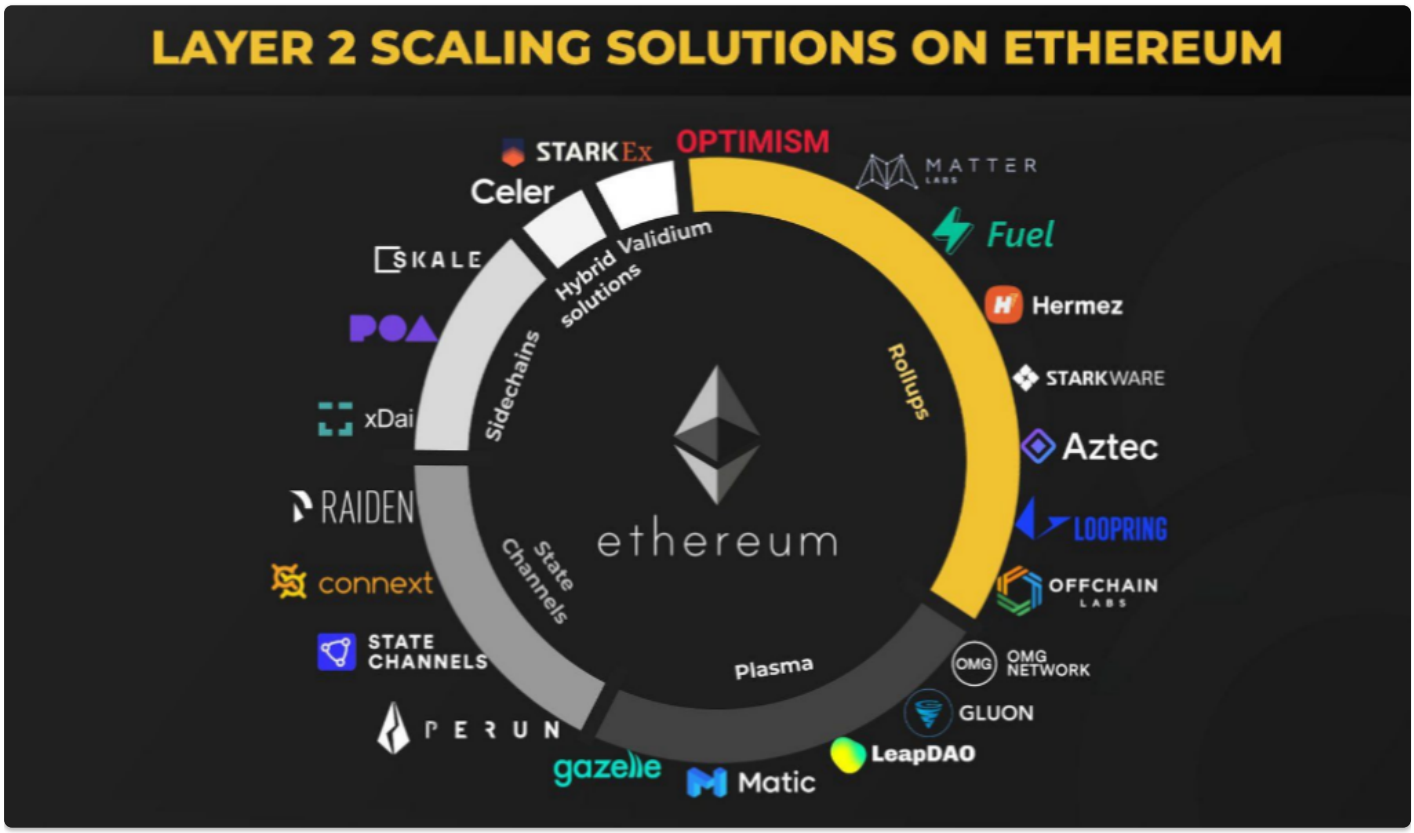
Introduction of Sharding

After the merge there will be increased network participation because of the reduced hardware requirements.
Vitalik sees 3 options

- **Shards remain as data depots**
- **A subset of the 64 shards will allow smart contracts**
- **Wait until increased use of ZKPs allows private transactions**

Off chain Scaling (Layer 2)

Generally speaking, transactions are submitted to these layer 2 nodes instead of being submitted directly to layer 1 (Mainnet). For some solutions the layer 2 instance then batches them into groups before anchoring them to layer 1, after which they are secured by layer 1 and cannot be altered.
A specific layer 2 instance may be open and shared by many applications, or may be deployed by one project and dedicated to supporting only their application.



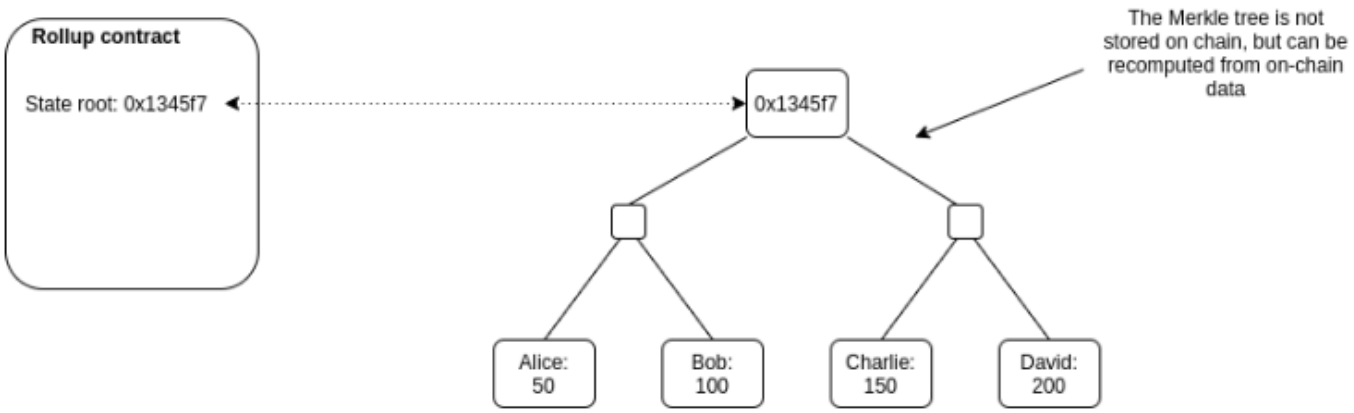
Rollups

Rollups are solutions that have

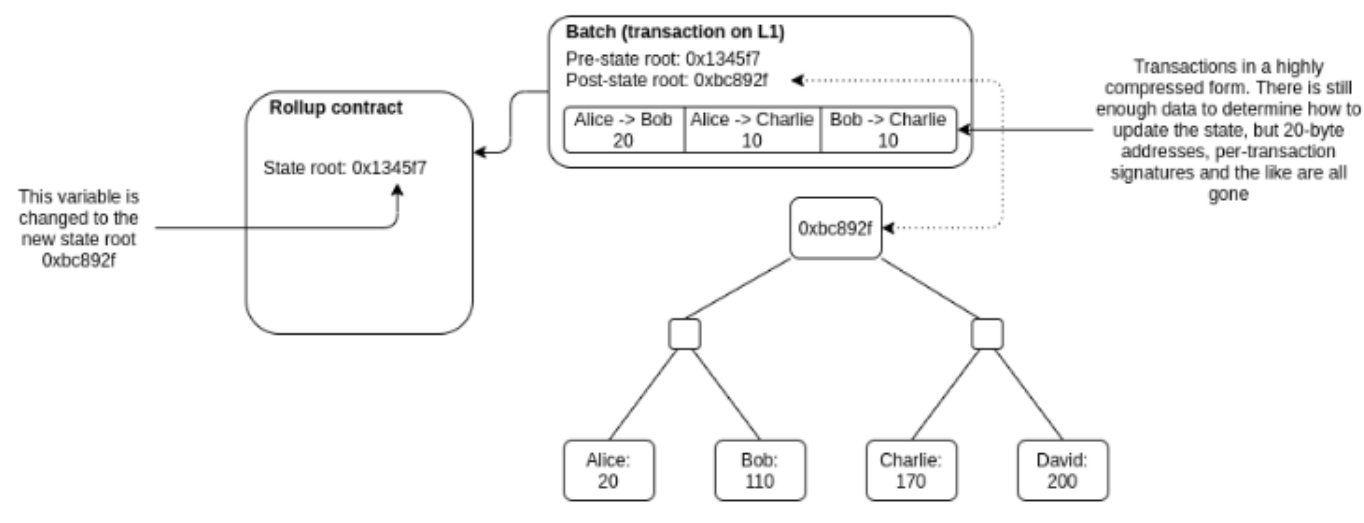
- transaction execution outside layer 1
- data or proof of transactions is on layer 1
- a rollup smart contract in layer 1 that can enforce correct transaction execution on layer 2 by using the transaction data on layer 1

The main chain holds funds and commitments to the side chains
The side chain holds state and performs execution
There needs to be some proof, either a fraud proof (Optimistic) or a validity proof (zk)

Rollups require "operators" to stake a bond in the rollup contract. This incentivises operators to verify and execute transactions correctly.



Anyone can publish a batch, a collection of transactions in a highly compressed form together with the previous state root and the new state root (the Merkle root after processing the transactions). The contract checks that the previous state root in the batch matches its current state root; if it does, it switches the state root to the new state root.

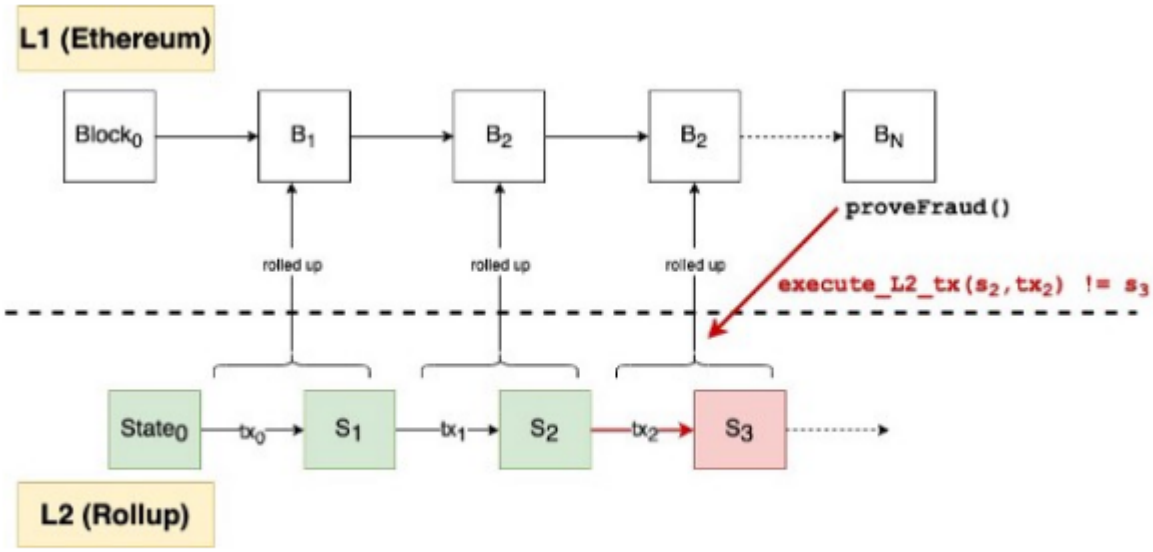


There are currently 2 types of rollups

- **Zero Knowledge Proof rollups**
- **Optimistic rollups**

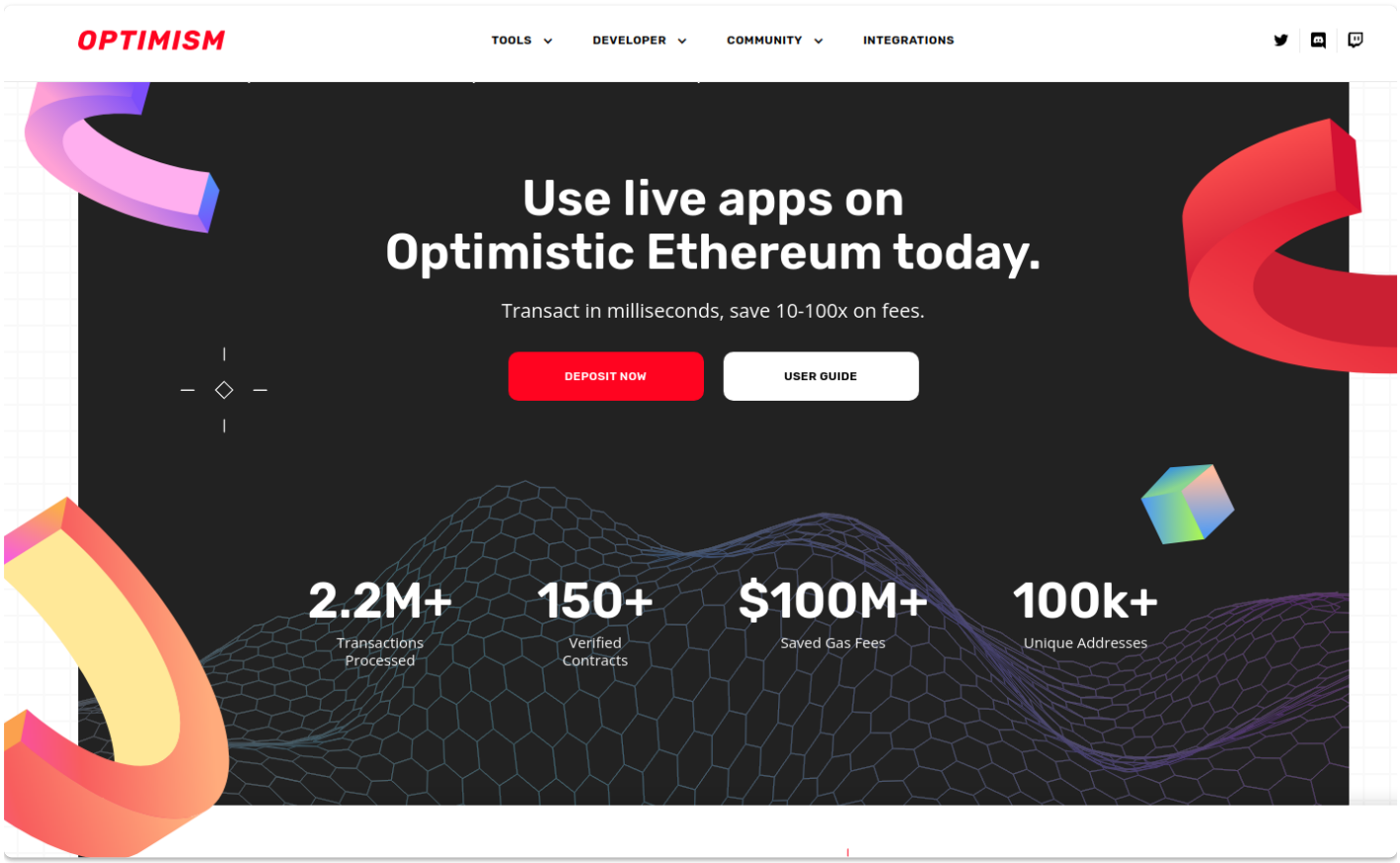
Optimistic Rollups

The name Optimistic Rollups originates from how the solution works. 'Optimistic' is used because aggregators publish only the bare minimum information needed with no proofs, assuming the aggregators run without commiting frauds, and only providing proofs in case of fraud. 'Rollups' is used because transactions are commited to main chain in bundles (that is, they are rolled-up).



Optimistic execution scales because L2 transactions can be replayed on L1 — but only when necessary!

Example Projects



Building Arbitrum for Secure Ethereum Dapps.

Experience economical efficiency of the blockchain without limits.

ARBITRUM

From Ethereum Docs

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.

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.

Process

- Developer sends transaction off-chain to a bonded aggregator
- Anyone with a bond may become an aggregator.
- There are multiple aggregators on the same chain.
- Fees are paid however the aggregator wants (account abstraction / meta transactions).
- Developer gets an instant guarantee that the transaction will be included or else the aggregator loses their bond.
- Aggregator locally applies the transaction & computes the new state root.
- Aggregator submits an Ethereum transaction (paying gas) which contains the transaction & state root (an optimistic rollup block).
- If anyone downloads the block & finds that it is invalid, they may prove the invalidity with verify_state_transition(prev_state, block, witness) which:
  - Slashes the malicious aggregator & any aggregator who built on top of the invalid block.
  - Rewards the prover with a portion of the aggregator's bond.

Types of Fraud Proof Systems

From explanation by Kelvin Fichter

A level 1 fault proof system is a system has an admin that can upgrade the system within the challenge window and can only be used by the admin.

A level 2 fault proof system still has an admin that can upgrade within the challenge window but is permissioned to allow a few others (besides the admin/team itself) to run the fault proof.

A level 3 fault proof is permissionless but still has an admin that can execute an upgrade within the challenge window. At level 3, you only need to trust that the admin won't do anything malicious and won't be compromised.

Level 4 proofs are completely permissionless and cannot be upgraded before users have a chance to withdraw their funds.
Level 4 fault proofs are the holy grail for an Optimistic Rollup.

Zero Knowledge Proof Rollups

See [Ethworks Report]

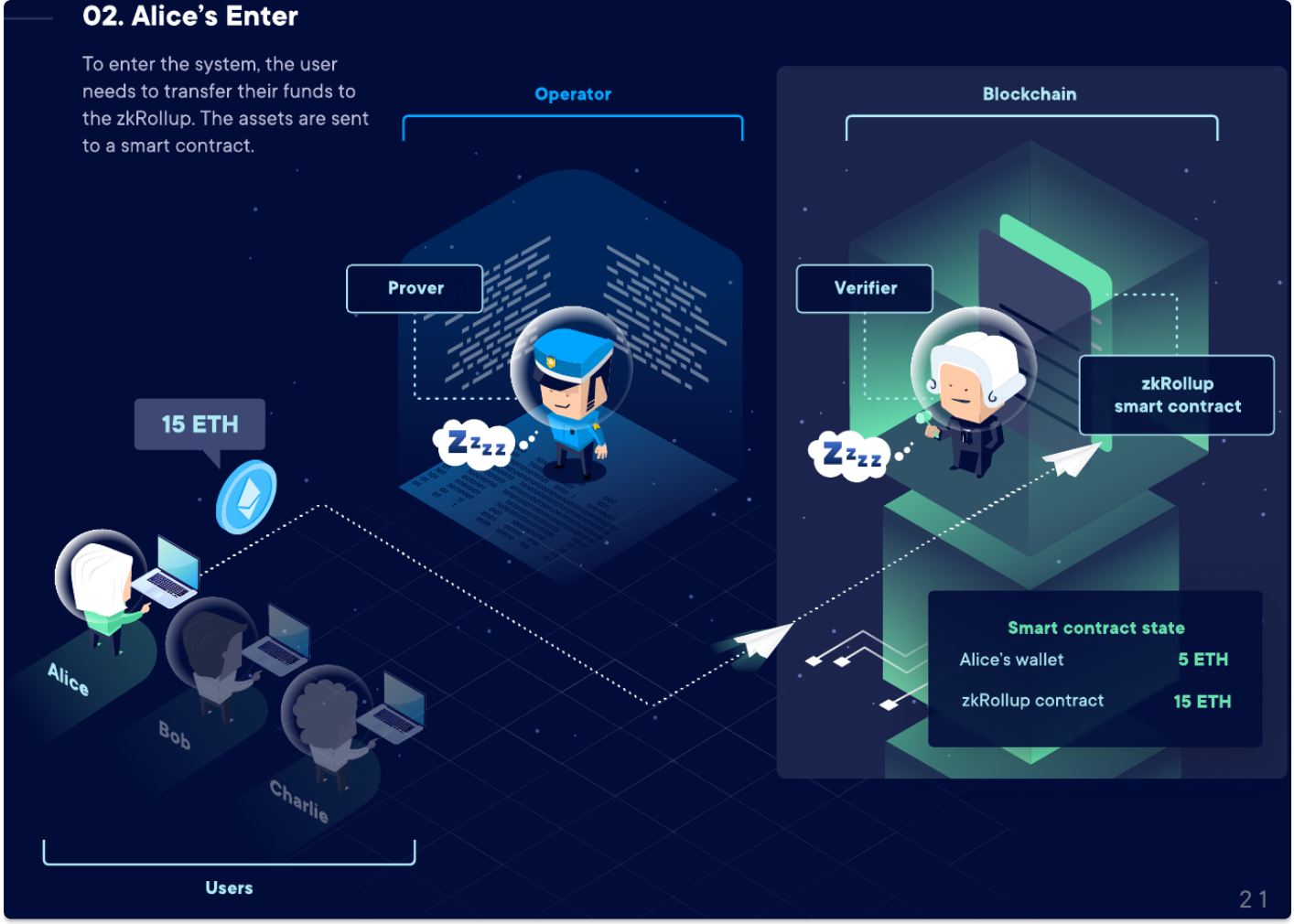
Note that in this context the proofs produced are often referred to as validity proofs since the zero knowledge aspect is not required.

ZK Rollup Process

From [Ethworks]

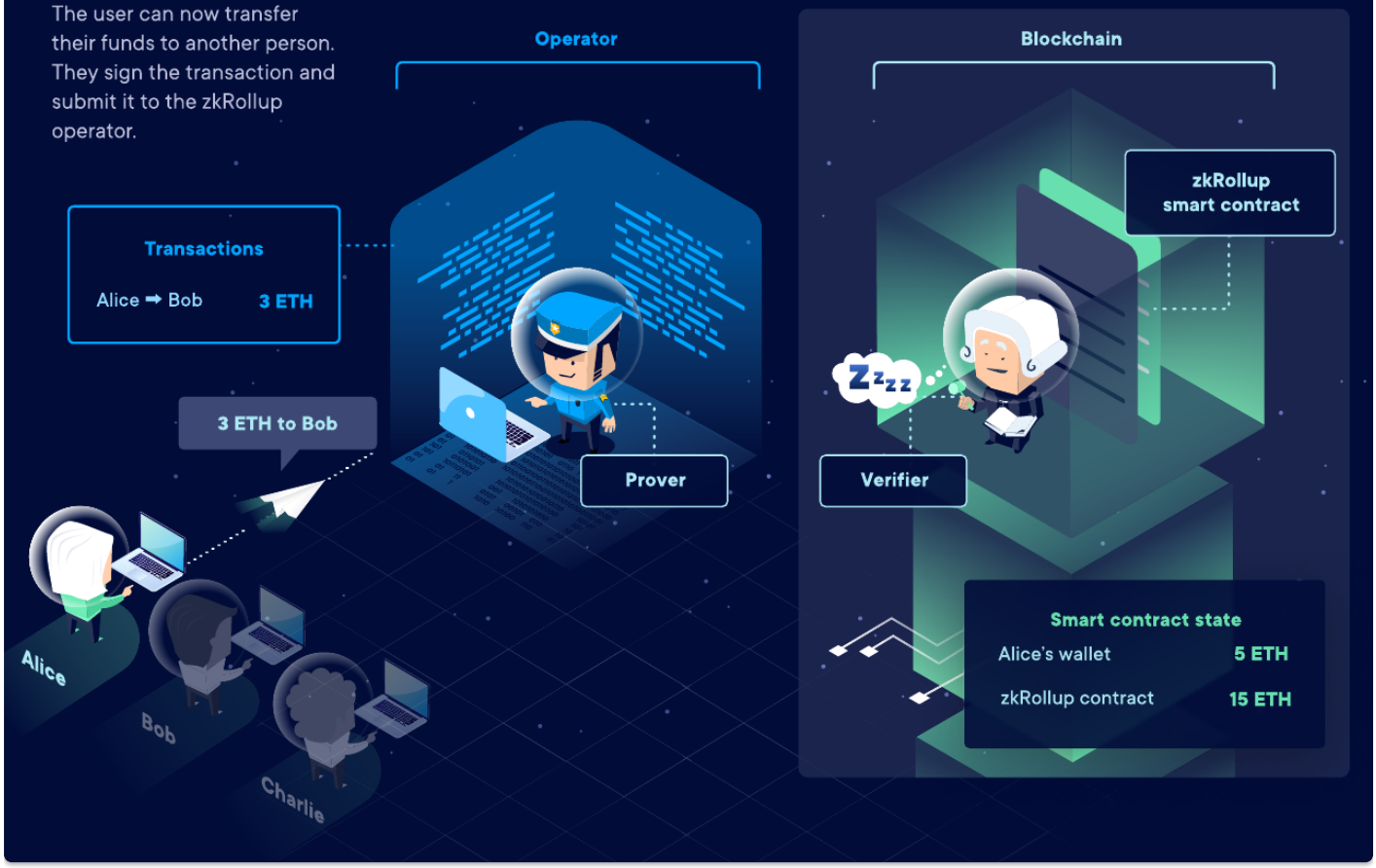
02. Alice's Enter

To enter the system, the user needs to transfer their funds to the zkRollup. The assets are sent to a smart contract.

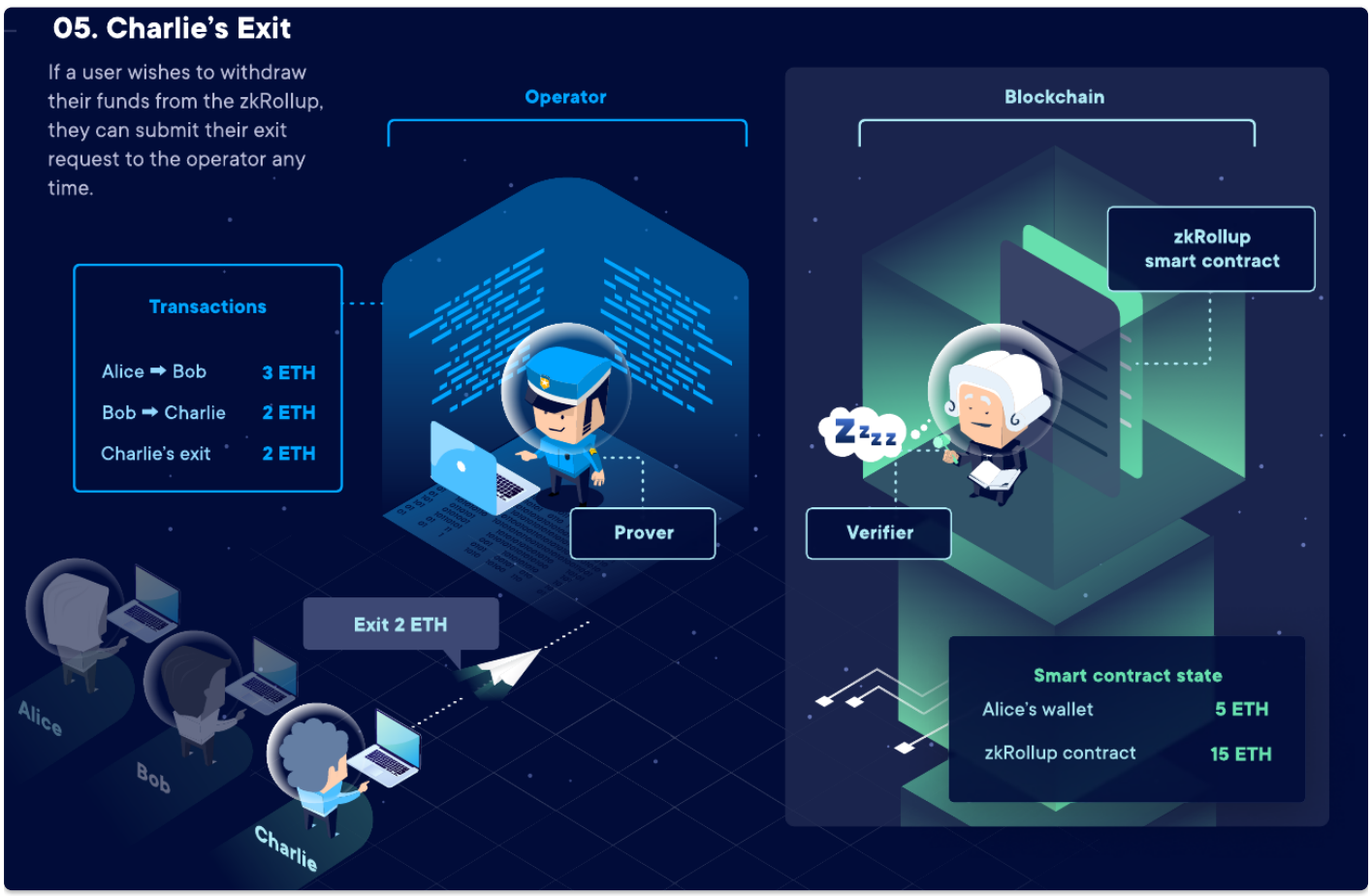


03. Alice's Transfer

The user can now transfer their funds to another person. They sign the transaction and submit it to the zkRollup operator.




04. Bob's Transfer
Operator
Blockchain
Transactions
Alice ➡ Bob 3 ETH
Bob ➡ Charlie 2 ETH
zkRollup smart contract
Prover
Verifier
2 ETH to Charlie
Alice
Bob
Charlie
Smart contract state
Alice's wallet 5 ETH
zkRollup contract 15 ETH


05. Charlie's Exit
If a user wishes to withdraw their funds from the zkRollup, they can submit their exit request to the operator any time.
Operator
Blockchain
Transactions
Alice ➡ Bob 3 ETH
Bob ➡ Charlie 2 ETH
Charlie's exit 2 ETH
zkRollup smart contract
Prover
Verifier
Exit 2 ETH
Alice
Bob
Charlie
Smart contract state
Alice's wallet 5 ETH
zkRollup contract 15 ETH

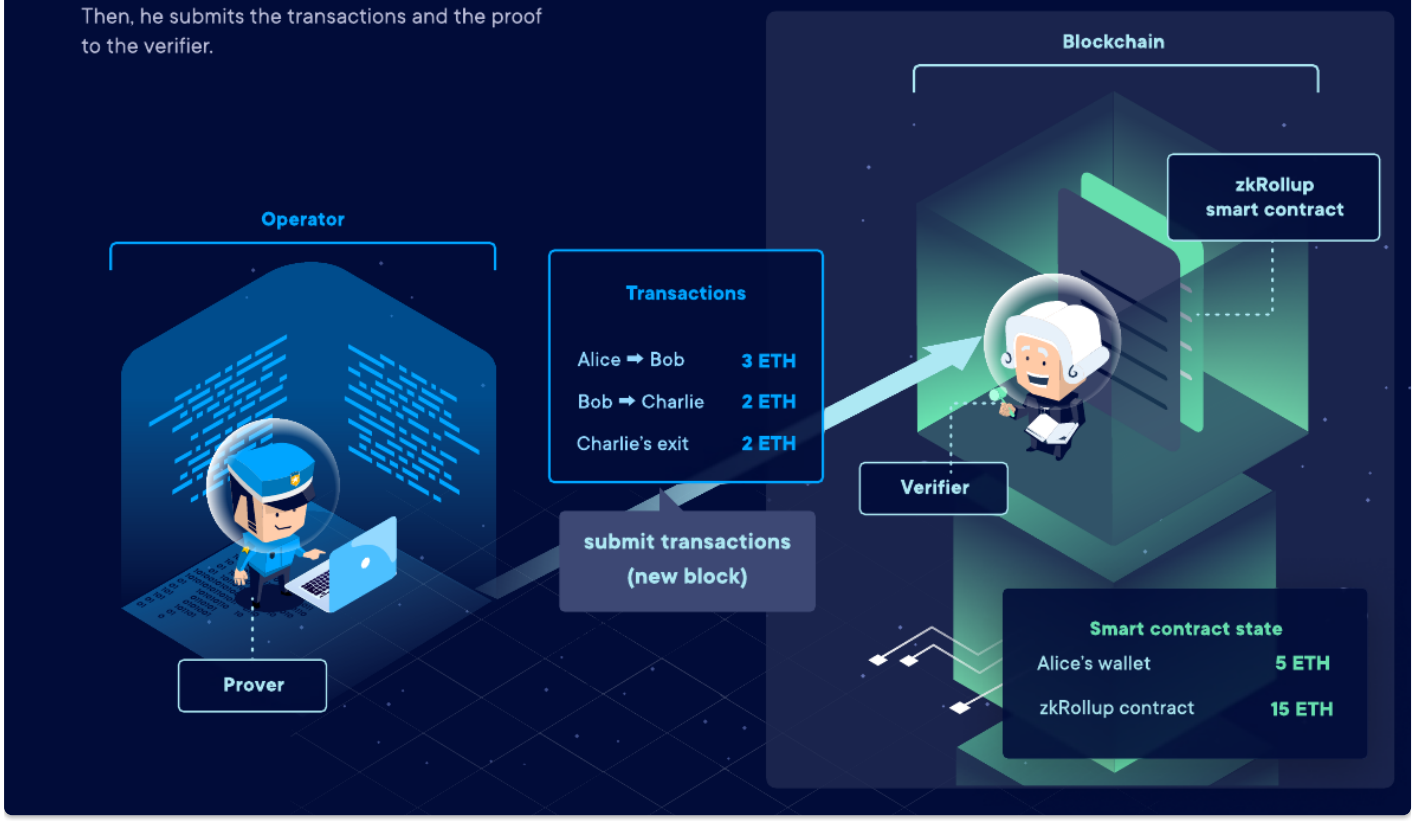
06. Collecting Transactions

In the meantime, the operator collects transactions and exit requests from many users.

* Note that even if Bob and Charlie didn't have any funds on the zkRollup, they could still receive transfers from other users.

07. Submitting Transactions

Once in a while, the operator bundles the collected transactions together and generates a ZK proof. Then, he submits the transactions and the proof to the verifier.




08. Submitting ZK Proof
The smart contract verifies the transactions and the proof. Once it's done, the transactions are finalized.
Operator
Blockchain
zkRollup smart contract
submit ZK proof (new block)
Verifier
Prover
Calldata
Alice ➡ Bob 3 ETH
Bob ➡ Charlie 2 ETH
Charlie's exit 2 ETH
Smart contract state
Alice's wallet 5 ETH
Charlie's wallet 2 ETH
zkSync contract 13 ETH

Comparison of the rollup types

Property	Optimistic rollups	ZK rollups
Fixed gas cost per batch	~40,000 (a lightweight transaction that mainly just changes the value of the state root)	~500,000 (verification of a ZK-SNARK is quite computationally intensive)
Withdrawal period	~1 week (withdrawals need to be delayed to give time for someone to publish a fraud proof and cancel the withdrawal if it is fraudulent)	Very fast (just wait for the next batch)
Complexity of technology	Low	High (ZK-SNARKs are very new and mathematically complex technology)
Generalizability	Easier (general-purpose EVM rollups are already close to mainnet)	Harder (ZK-SNARK proving general-purpose EVM execution is much harder than proving simple computations, though there are efforts (eg. Cairo) working to improve on this)
Per-transaction on-chain gas costs	Higher	Lower (if data in a transaction is only used to verify, and not to cause state changes, then this data can be left out, whereas in an optimistic rollup it would need to be published in case it needs to be checked in a fraud proof)
Off-chain computation costs	Lower (though there is more need for many full nodes to redo the computation)	Higher (ZK-SNARK proving especially for general-purpose computation can be expensive, potentially many thousands of times more expensive than running the computation directly)

See this article from Starkware comparing the types of proofs

Transaction Compression

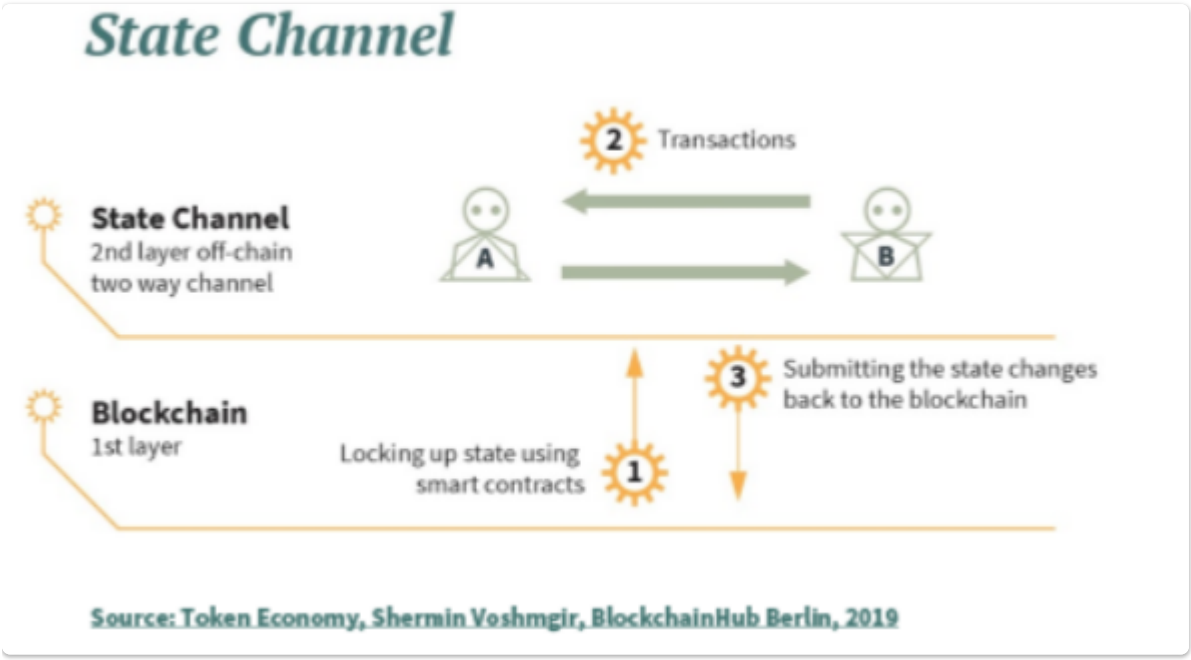
How does compression work?

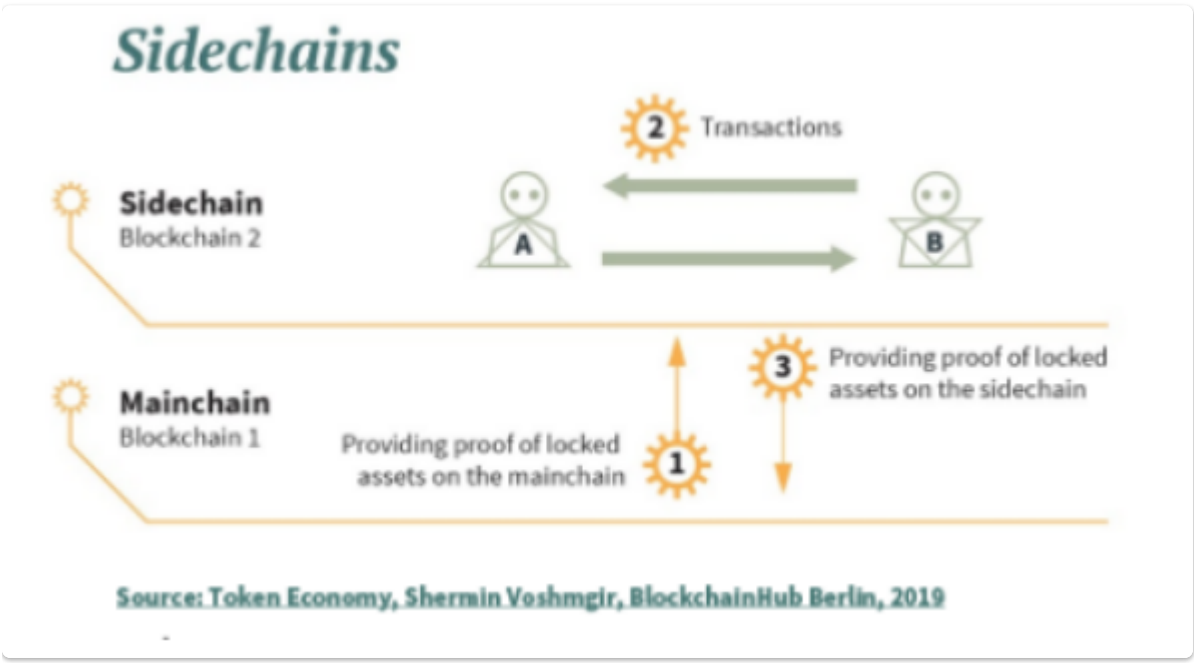
A simple Ethereum transaction (to send ETH) takes ~110 bytes. An ETH transfer on a rollup, however, takes only ~12 bytes:

Parameter	Ethereum	Rollup
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	~12

Part of this is simply superior encoding: Ethereum's RLP wastes 1 byte per value on the length of each value. But there are also some very clever compression tricks that are going on:

State Channels and Side Chains





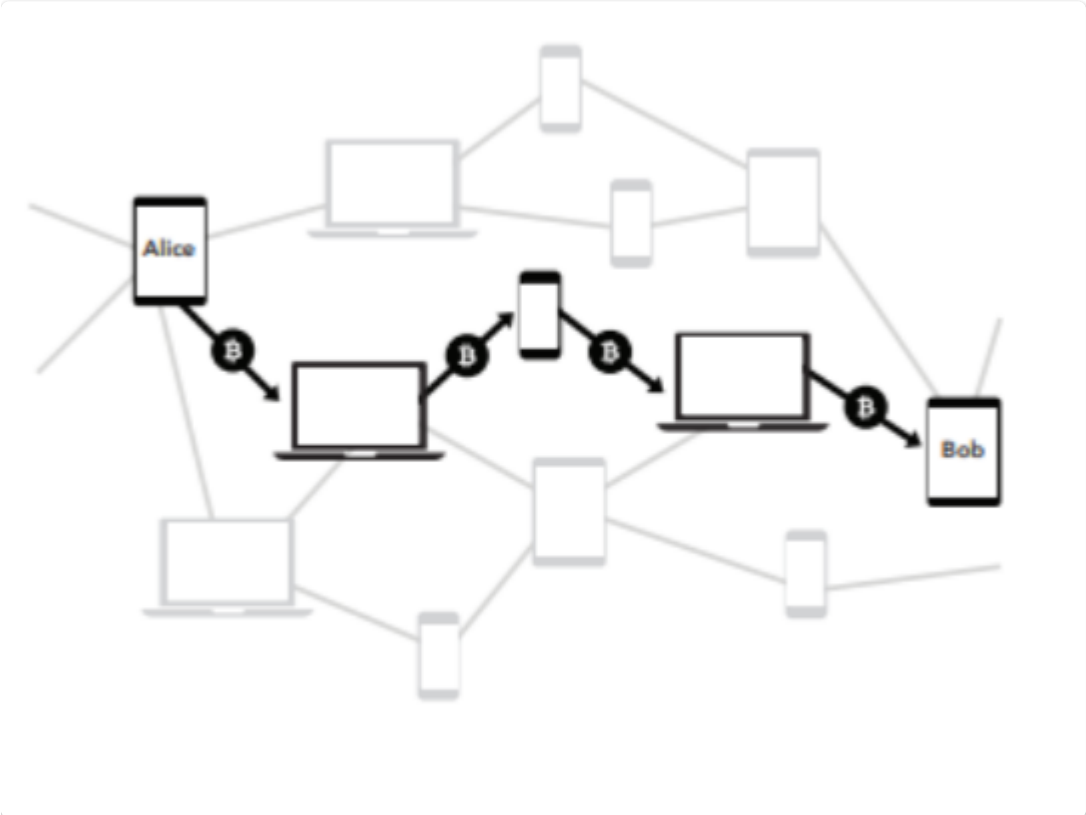
State channels

Payment channels are a specialised form of state channel

State channels allow participants to transact many off-chain while but only require 2 transactions on the L1 blockchain, one at the start and one at the end. An ideal use case for this is micropayments.
Participants must lock a portion of Ethereum's state, like an ETH deposit, into a multisig contract.
Locking the state in this way is the first transaction and opens up the channel. The participants can then transact quickly and freely off-chain. When the interaction is finished, a final on-chain transaction is submitted, unlocking the state.

Examples

[Lightning network]



Funds are placed into a two-party, multisignature "channel" bitcoin address. This channel is represented as an entry on the bitcoin public ledger. In order to spend funds from the channel, both parties must agree on the new balance. The current balance is stored as the most recent transaction signed by both parties, spending from the channel address. To make a payment, both parties sign a new exit transaction spending from the channel address. All old exit transactions are invalidated by doing so. The Lightning Network does not require cooperation from the counterparty to exit the channel. Both parties have the option to unilaterally close the channel, ending their relationship. Since all parties have multiple multisignature channels with many different users on this network, one can send a payment to any other party across this network.

Advantages

- **Instant Payments.**
  Bitcoin aggregates transactions into blocks spaced ten minutes apart. Payments are widely regarded as secure on bitcoin after confirmation of six blocks, or about one hour. On the Lightning Network, payments don't need block confirmations, and are instant and atomic. Lightning can be used at retail point-of-sale terminals, with user device-to-device transactions, or anywhere instant payments are needed

- **Micropayments.**
  New markets can be opened with the possibility of micropayments. Lightning enables one to send funds down to 0.00000001 bitcoin without custodial risk. The bitcoin blockchain currently enforces a minimum output size many hundreds of times higher, and a fixed per-transaction fee which makes micropayments impractical. Lightning allows minimal payments denominated in bitcoin, using actual bitcoin transactions.

Sidechains

A sidechain is an independent EVM-compatible blockchain which runs in parallel to Mainnet. These are compatible with Ethereum via two-way bridges, and run under their own chosen rules of consensus, and block parameters.

Examples

- Skale
- POA Network - now part of Gnosis
- [Gnosis chain](https://docs.gnosischain.com/

Advantages

- Easy to implement with existing technology
- EVM compatible

Disadvantages

- Consensus mechanism may not be better
- Not secured by layer 1, so more susceptible to fraud
- Probably less decentralised

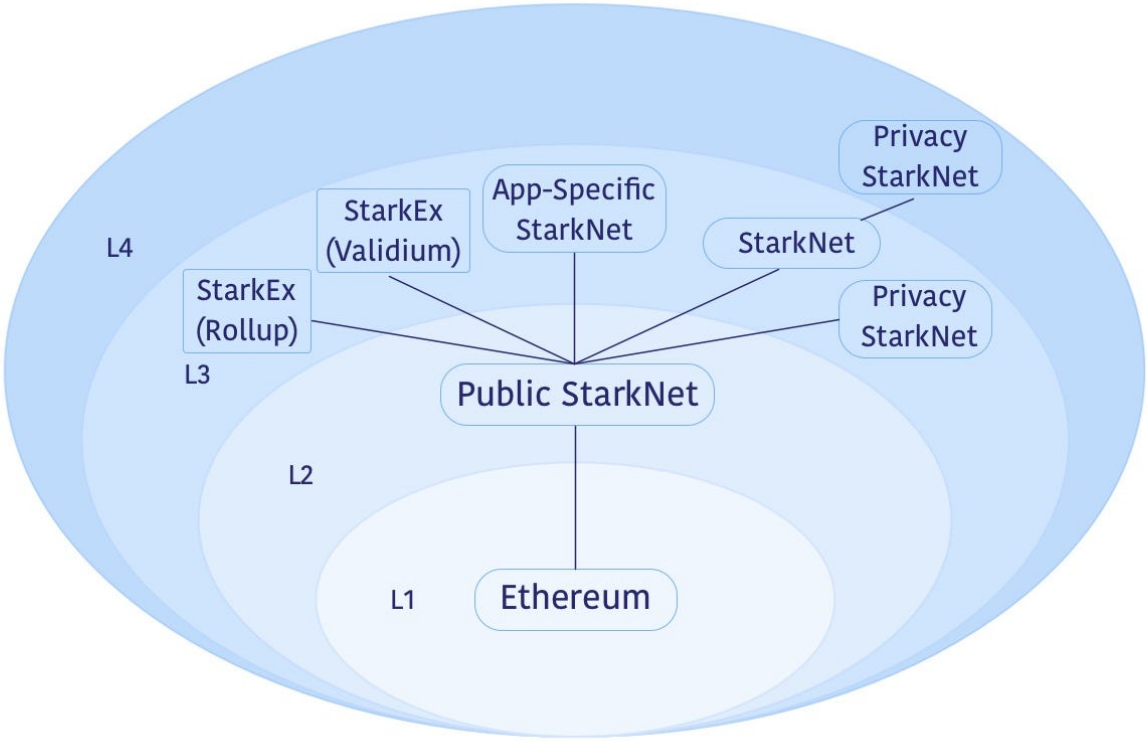
Plasma Chains

A plasma chain is a separate blockchain that is anchored to the main Ethereum chain, and uses fraud proofs (like Optimistic rollups) to arbitrate disputes.
These chains are sometimes referred to as "child" chains as they are essentially smaller copies of the Ethereum Mainnet.
Merkle trees enable creation of a limitless stack of these chains that can work to offload bandwidth from the parent chains (including Mainnet). These derive their security through fraud proofs, and each child chain has its own mechanism for block validation.

Fractal Scaling

The rollup process can be extended further by the use of recursive proofs, essentially you can create a proof that proves a secondary proof (that proves another proof ....)

There is now talk about L3 for specific applications, where L3 a bundle of L3 proofs can be sent as a proof to L2, which will be part of the bundle of proofs sent to L1.
This gives a further boost to scalability.



zkEVM Solutions

Rollup Recap

Rollups are solutions that have

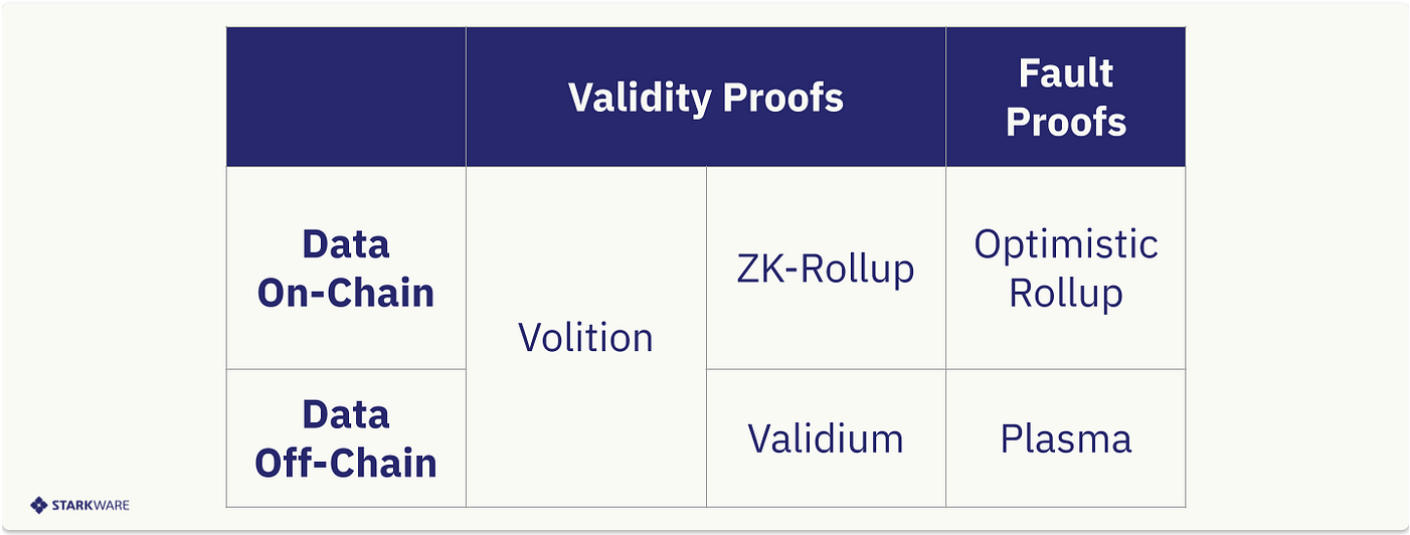
- transaction execution outside layer 1
- transaction data and proof of transactions is on layer 1
- a rollup smart contract in layer 1 that can enforce correct transaction execution on layer 2 by using the transaction data on layer 1

The main chain holds funds and commitments to the side chains
The side chain holds additional state and performs execution
There needs to be some proof, either a fraud proof (Optimistic) or a validity proof (zk)

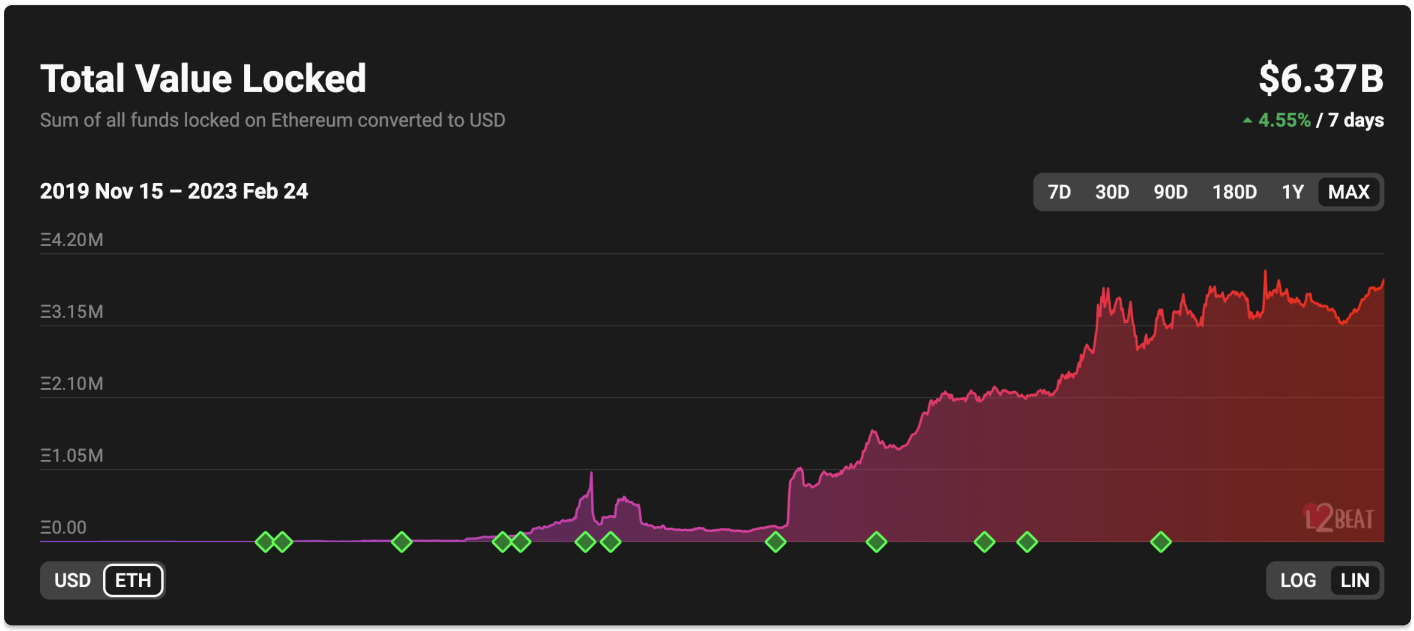
Rollups require “operators” to stake a bond in the rollup contract. This incentivises operators to verify and execute transactions correctly.

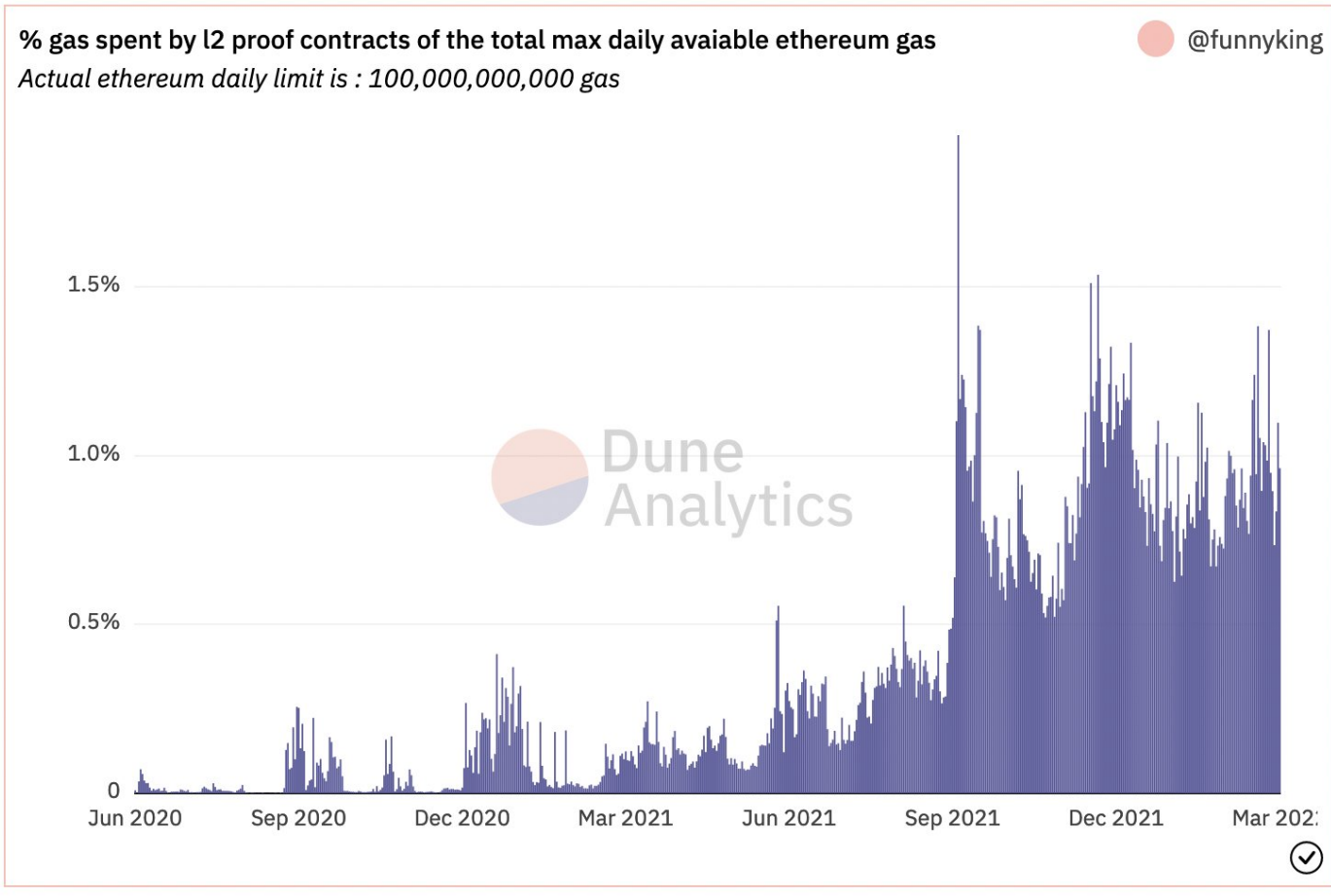
Data Availability in general

In order to re create the state, transaction data is needed, the data availability question is where this data is stored and how to make sure it is available to the participants in the system.



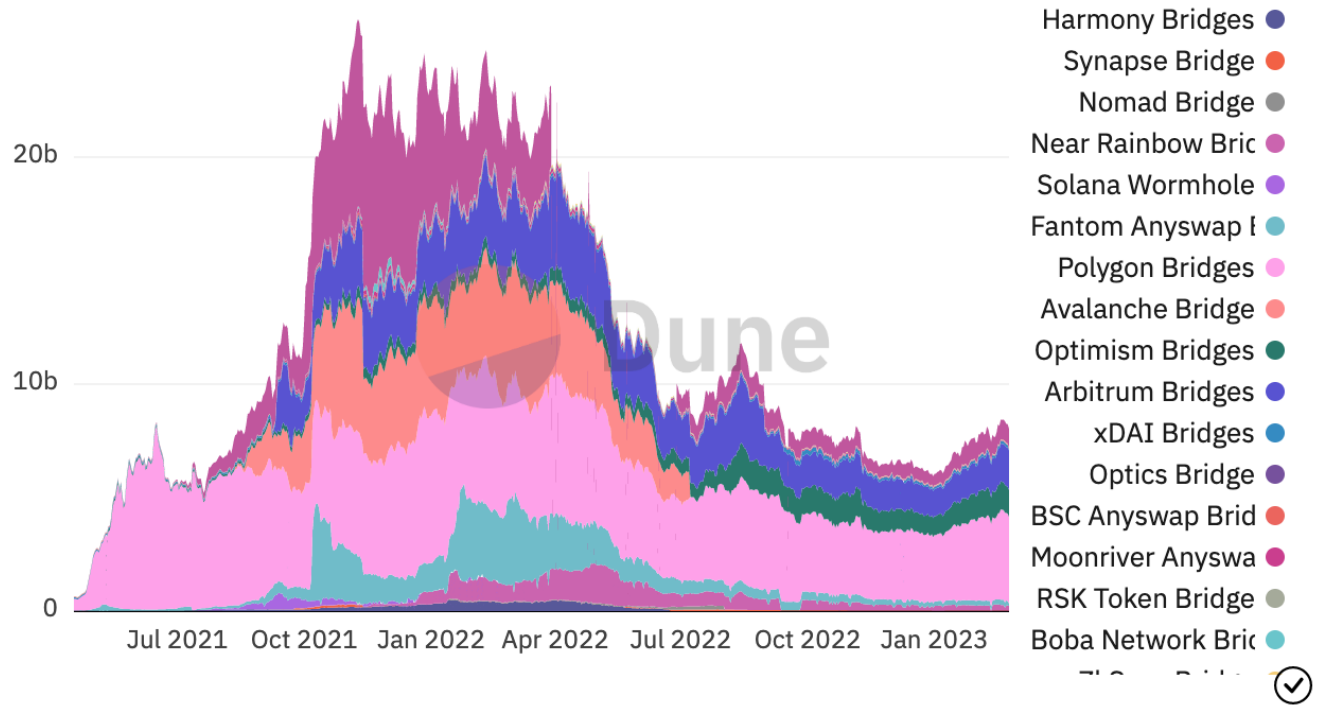
L2 Statistics





Ethereum bridges TVL over time 📈

@eliasimos



Transaction compression and costs

For zk rollups it is expensive to verify the validity proof.
For STARKs, this costs ~5 million gas, which when aggregated is about 384 gas cost per transaction.
Due to compression techniques, the calldata cost is actually only 86 gas.
This is further reduced by the calldata EIP448 to 16.1 gas.

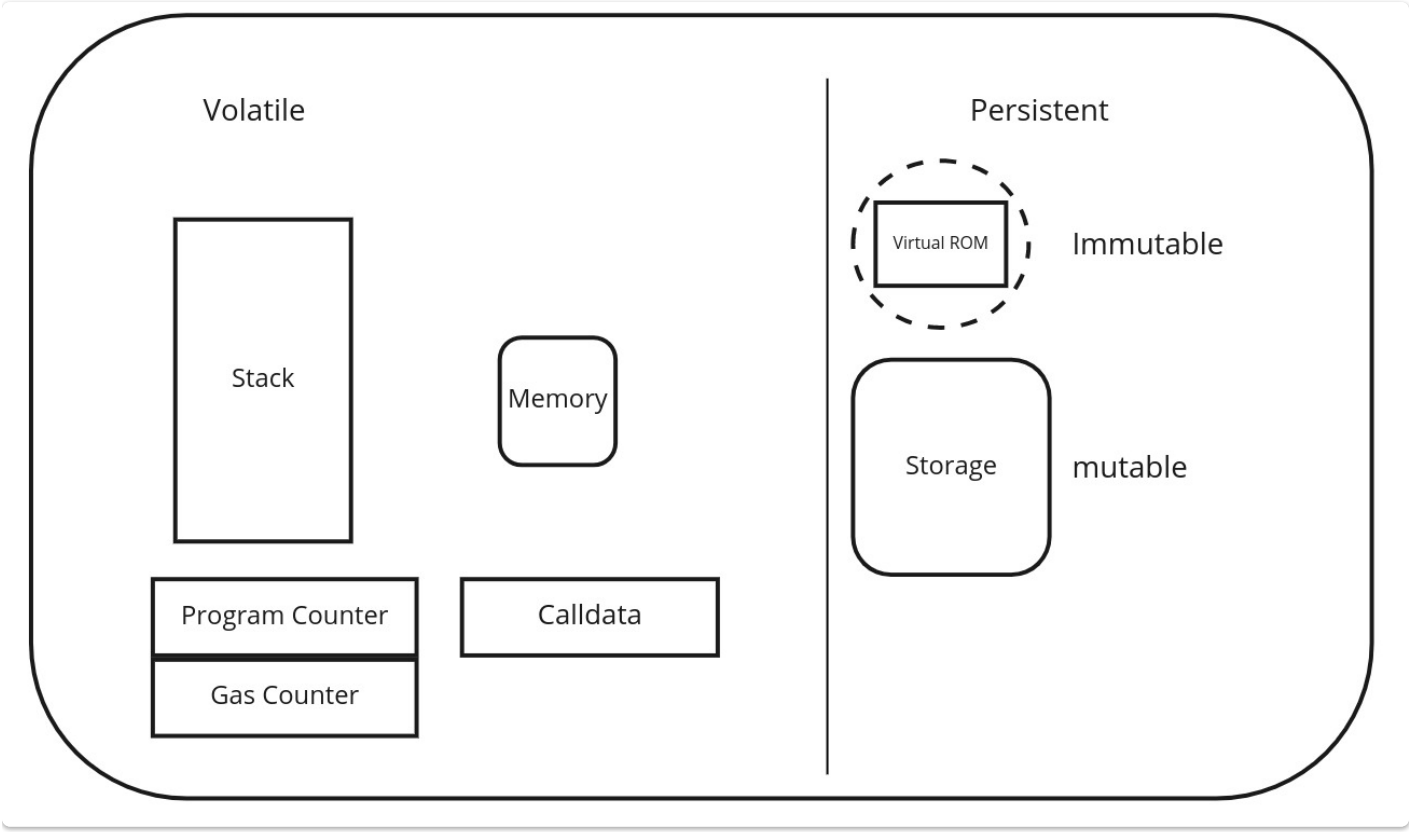
The batch cost is poly-log, so if activity increases 100x, the batch costs will decrease to only 4–5 gas per transaction, in which case the calldata reduction would have a huge impact.

At 100x the TPS today on dYdX, the total on-chain cost will reduce to only 21 gas
At this point, the bottleneck becomes prover costs for the rollup as much as on-chain gas fees, see Polygon Zero and recursive proofs

Approaches to zkRollups on Ethereum

1. Building application-specific circuit (although this can be fairly generic as in Starknet)
2. Building a universal “EVM” circuit for smart contract execution

zkEVM Solutions in general



The opcode of the EVM needs to interact with Stack, Memory, and Storage during execution. There should also be some contexts, such as gas/program counter, etc. Stack is only used for Stack access, and Memory and Storage can be accessed randomly.

AppliedZKP zkEVM

AppliedZKP divides proofs into two types:

1. State proof, used to check the correctness of the state transition in Stack/Memory/Storage.
2. EVM proof, used to check that the correct opcode is used at the correct time, the correctness of the opcode itself, the validity of the opcode, and all the abnormal conditions (such as out_of_gas) that may be encountered during the execution of the opcode.

EVM processing

In general the EVM will

1. Read elements from stack, memory or storage
2. Perform some computation on those elements
3. Write back results to stack, memory or storage

So our circuit has to model / prove this process, in particular

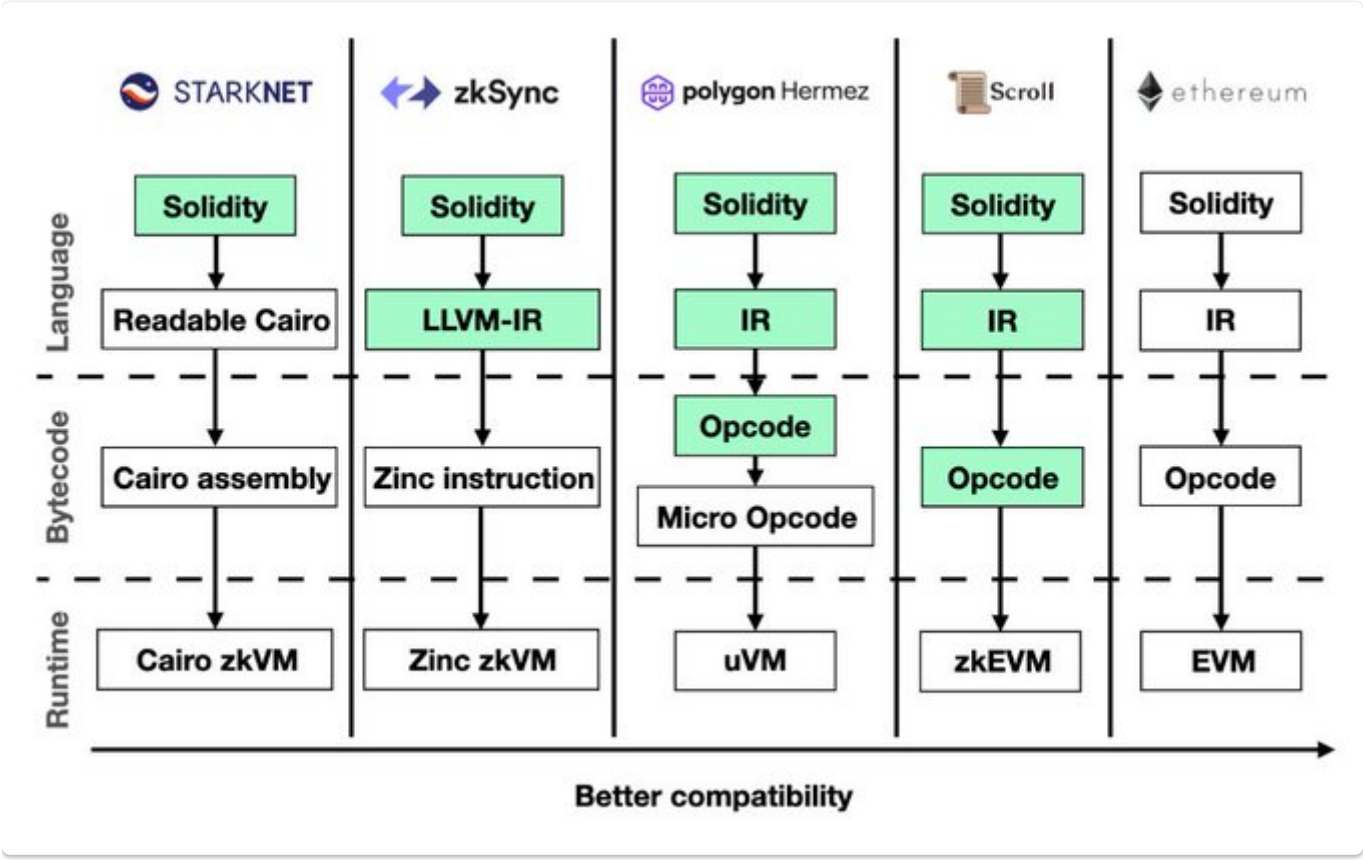
- The bytecode is correctly loaded from persistent storage
- The opcodes in the bytecode are executed in sequence

- Each opcode is executed correctly (following the above 3 steps)

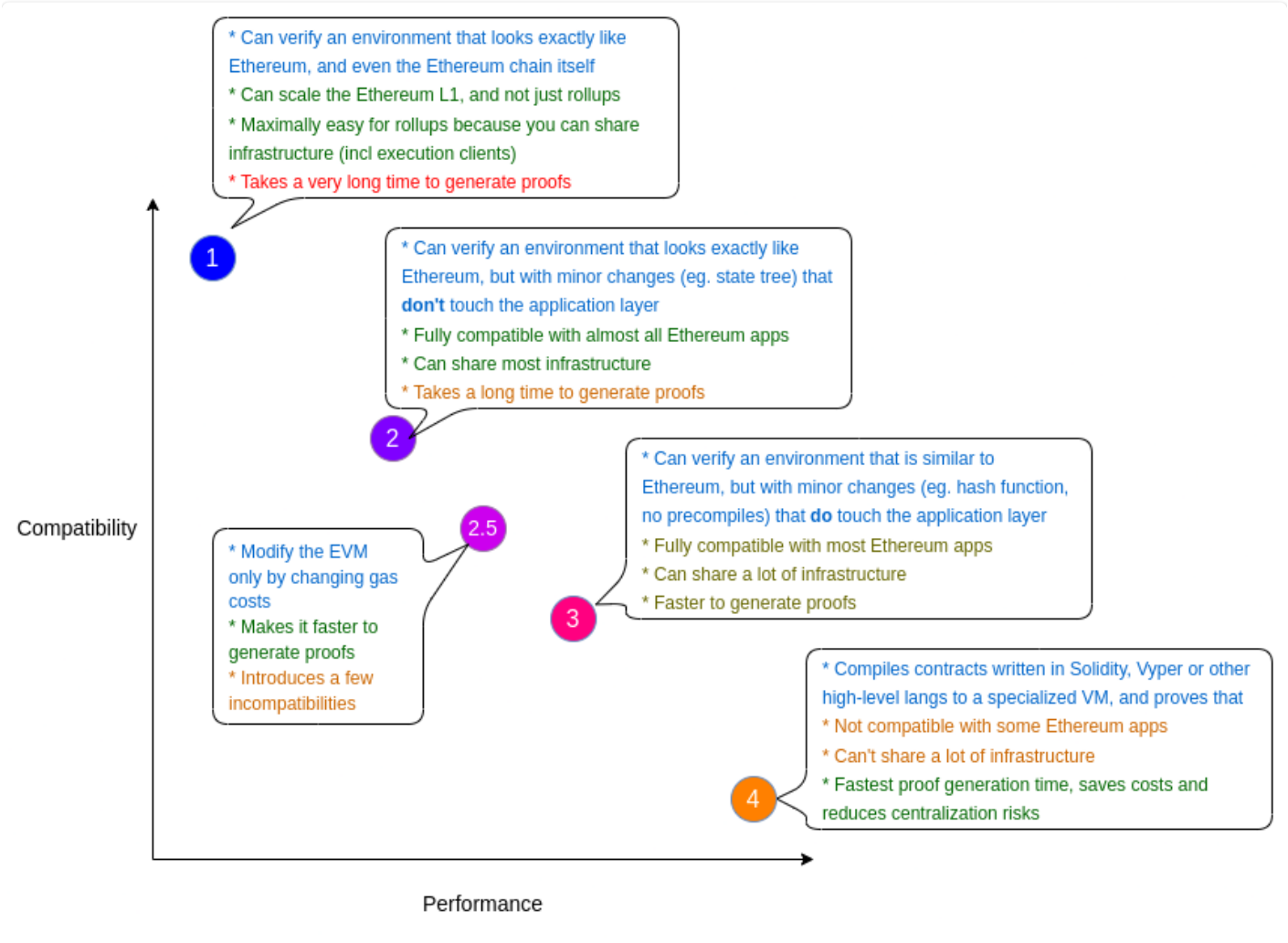
Design challenges in designing a zkEVM

1. We are constrained by the cryptography (curves, hash functions) available on Ethereum.
2. The EVM is stack based rather than register based
3. The EVM has a 256 bit word (not a natural field element size)
4. EVM storage uses keccak and Merkle Patricia trees, which are not zkp friendly
5. We need to model the whole EVM to do a simple op code.

zkEVM taxonomy



See Vitalik [article]



Type 1 (fully Ethereum-equivalent)

See zkEVM research team

Type 2 (fully EVM-equivalent)

(not quite Ethereum-equivalent)
Scroll
Hermez

Type 2.5 (EVM-equivalent, except for gas costs)

Type 3 (almost EVM-equivalent)

Scroll and Hermez in their current form

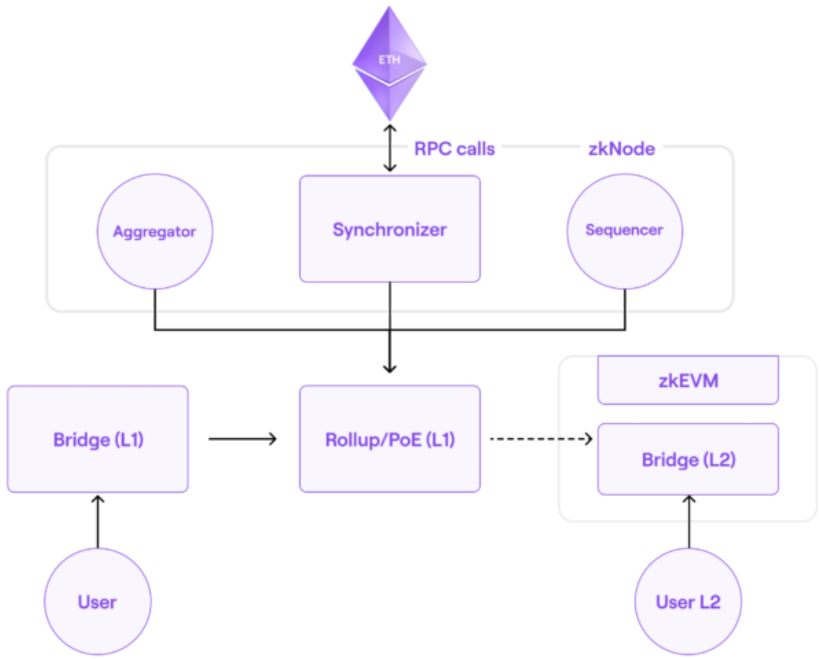
Type 4 (high-level-language equivalent)

ZKSync
Starknet + Warp

Also see
The ZK-EVM Community Edition (bootstrapped by community contributors including Privacy and Scaling Explorations, the Scroll team, Taiko and others) is a Tier 1 ZK-EVM.

Polygon zkEVM

See Repo



- `polygon zkEVM` is a new zk-rollup that provides Ethereum Virtual Machine (EVM) equivalence (opcode-level compatibility) for a transparent user experience and existing Ethereum ecosystem and tooling compatibility.
- It consists on a decentralized Ethereum Layer 2 scalability solution utilising cryptographic zero-knowledge technology to provide validation and fast finality of off-chain transaction computations.
- This approach required the recreation of all EVM opcodes for transparent deployment and transactions with existing Ethereum smart contracts. For this purpose a new set of tools and technologies were created and engineered and are contained in this organization.

Scroll

Scroll Features

- Users will be able to play with a few key demo applications such as a Uniswap fork with familiar web interfaces such as Metamask.
- Users will be able to view the state of the Scroll testnet via block explorers.
- Scroll will run a node that supports unlimited read operations (e.g. getting the state of accounts) and user-initiated transactions involving interactions with the pre-deployed demo applications (e.g. transfers of ERC-20 tokens or swaps of tokens).
- Rollers will generate and aggregate validity proofs for part of the zkEVM circuits to ensure a stable release. In the next testnet phase, we will ramp up this set of zkEVM circuits.

- Bridging assets between these testnet L1 and L2s will be enabled through a smart contract bridge, though arbitrary message passing will not be supported in this release.

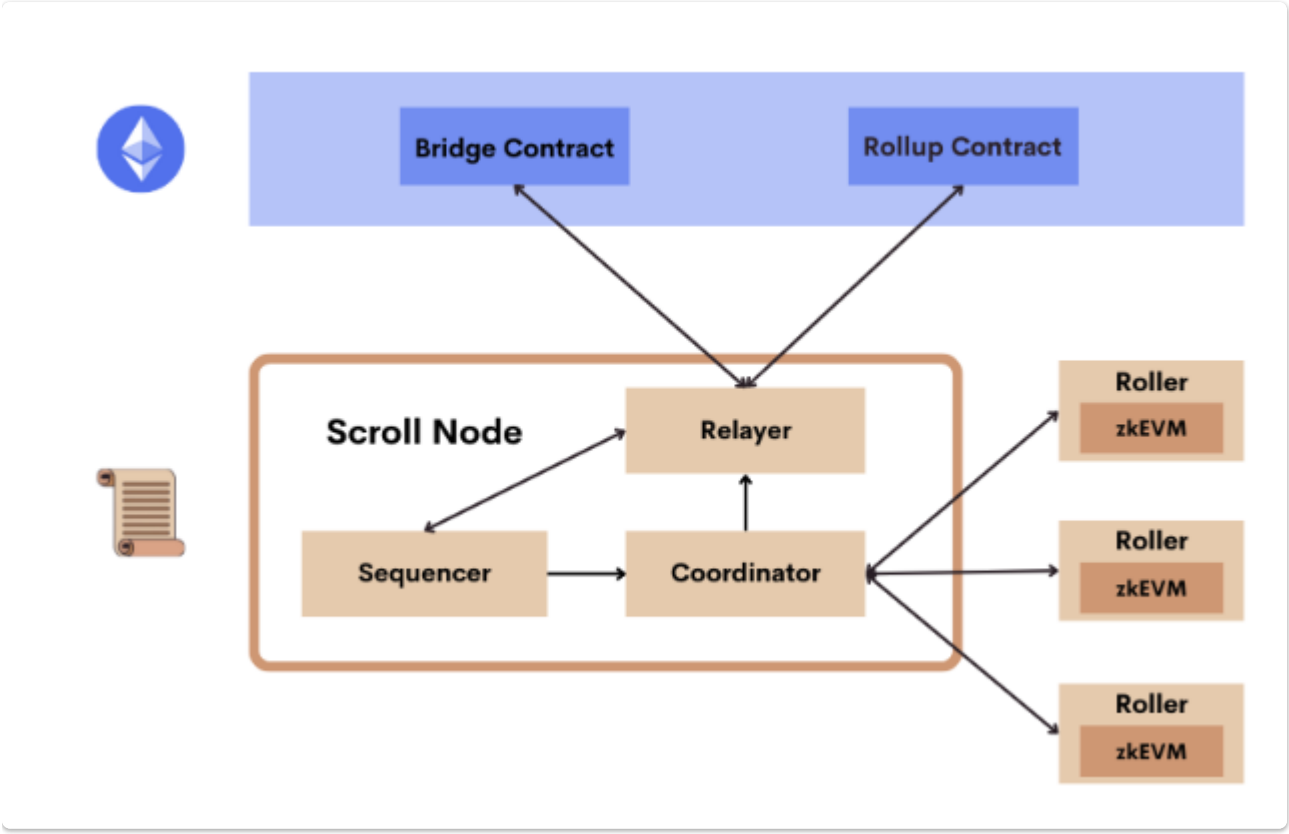
From article and article

Approaches to zkRollups on Ethereum

1. Building application-specific circuit (although this can be fairly generic as in Starknet)
2. Building a universal “EVM” circuit for smart contract execution

---

Scroll Architecture



Components

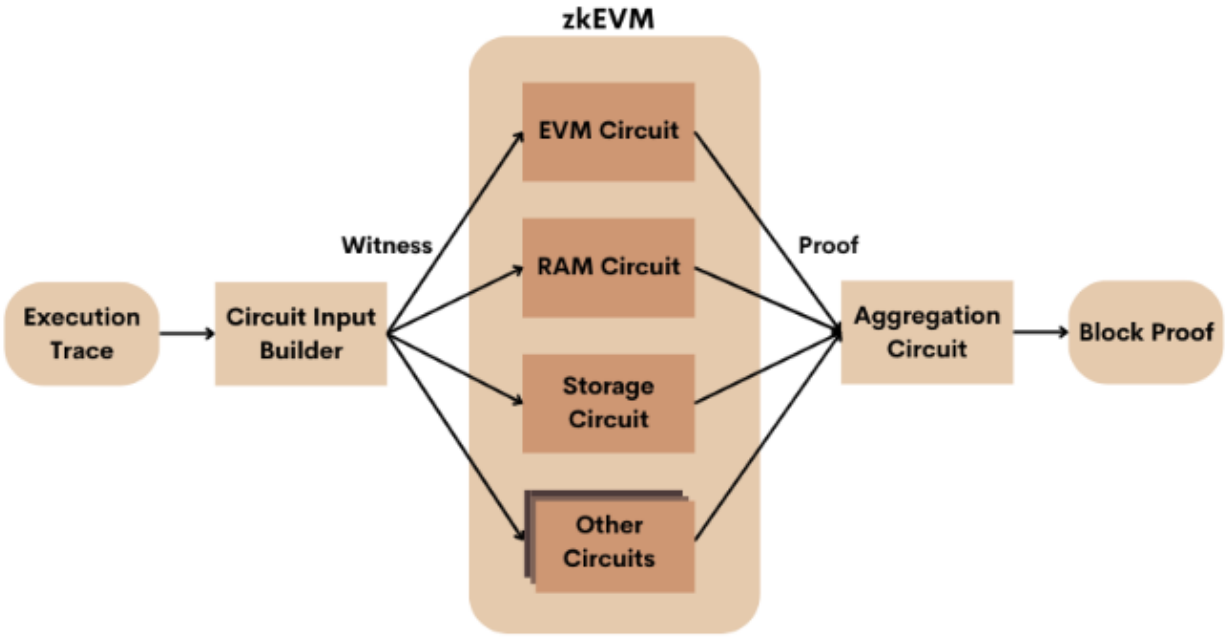
The Sequencer provides a JSON-RPC interface and accepts L2 transactions. Every few seconds, it retrieves a batch of transactions from the L2 mempool and executes them to generate a new L2 block and a new state root.

Once a new block is generated, the Coordinator is notified and receives the execution trace of this block from the Sequencer.
It then dispatches the execution trace to a randomly-selected Roller from the roller pool for proof generation.

The Relayer watches the bridge and rollup contracts deployed on both Ethereum and Scroll. It has two main responsibilities.

1. It monitors the rollup contract to keep track of the status of L2 blocks including their data availability and validity proof.
2. It watches the deposit and withdraw events from the bridge contracts deployed on both Ethereum and Scroll and relays the messages from one side to the other.

Rollers - creating proofs

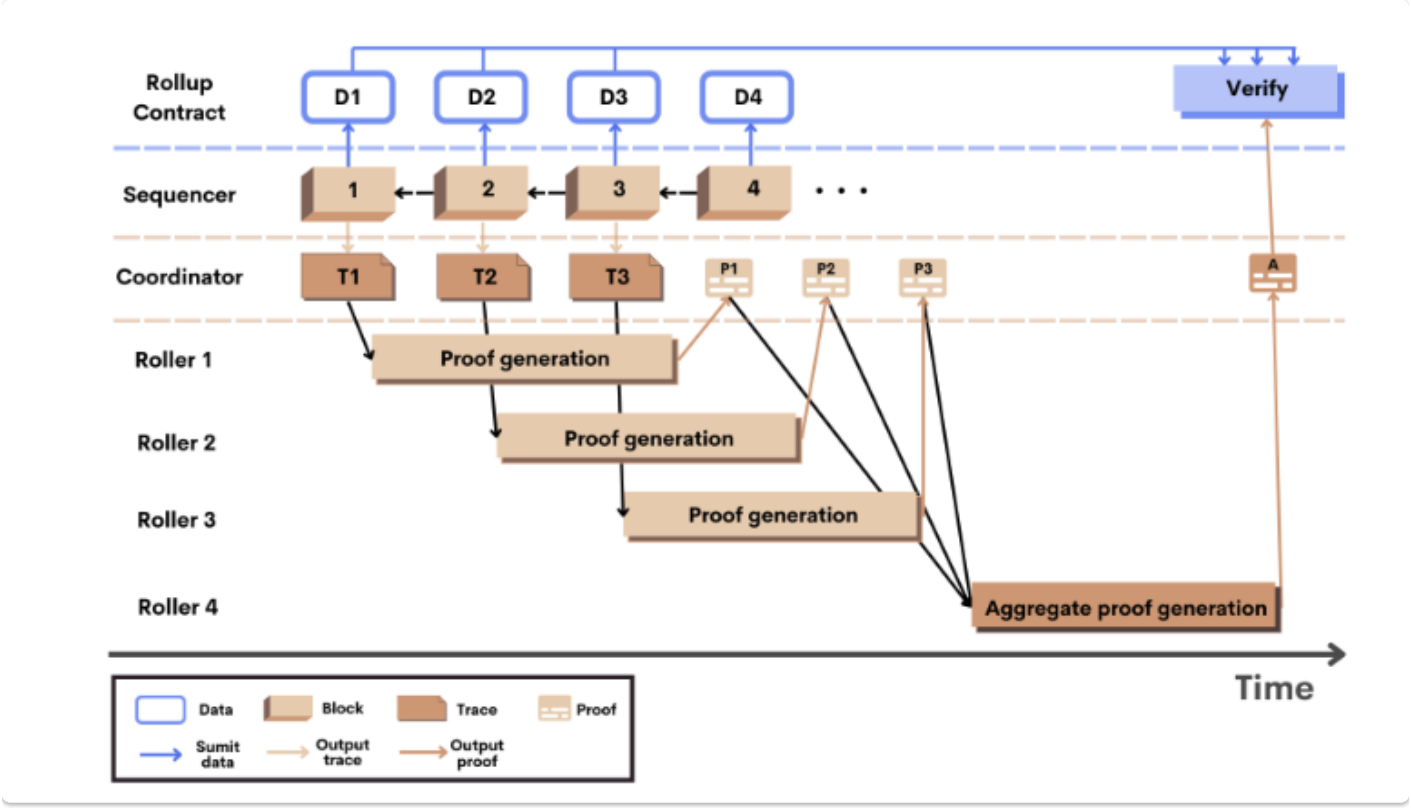


The Rollers serve as provers in the network that are responsible for generating validity proofs for the zkRollup

- A Roller first converts the execution trace received from the Coordinator to circuit witnesses.
- It generates proofs for each of the zkEVM circuits.
- Finally, it uses proof aggregation to combine proofs from multiple zkEVM circuits into a single block proof.

The Rollup contract on L1 receives L2 state roots and blocks from the Sequencer.
It stores state roots in the Ethereum state and L2 block data as Ethereum calldata.
This provides data availability for Scroll blocks and leverages the security of Ethereum to ensure that indexers including the Scroll Relayer can reconstruct L2 blocks.
Once a block proof establishing the validity of an L2 block has been verified by the Rollup contract, the corresponding block is considered finalized on Scroll.

A useful sequence diagram from the Scroll Documentation



L2 blocks in Scroll are generated, committed to base layer Ethereum, and finalized in the following sequence of steps:

1. The Sequencer generates a sequence of blocks. For the *i*-th block, the Sequencer generates an execution trace ***T*** and sends it to the Coordinator. Meanwhile, it also submits the transaction data ***D*** as calldata to the Rollup contract on Ethereum for data availability and the resulting state roots and commitments to the transaction data to the Rollup contract as state.
2. The Coordinator randomly selects a Roller to generate a validity proof for each block trace. To speed up the proof generation process, proofs for different blocks can be generated in parallel on different Rollers.
3. After generating the block proof ***P*** for the *i*-th block, the Roller sends it back to the Coordinator. Every *k* blocks, the Coordinator dispatches an aggregation task to another Roller to aggregate *k* block proofs into a single aggregate proof ***A***.
4. Finally, the Coordinator submits the aggregate proof ***A*** to the Rollup contract to finalize L2 blocks *i+1* to *i+k* by verifying the aggregate proof against the state roots and transaction data commitments previously submitted to the rollup contract.