Lesson 7

Topics this week

- Scalability / L2 solutions
- Security
- Foundry
- MEV
- Gas Optimisation

Scalability

The scalability trilemma

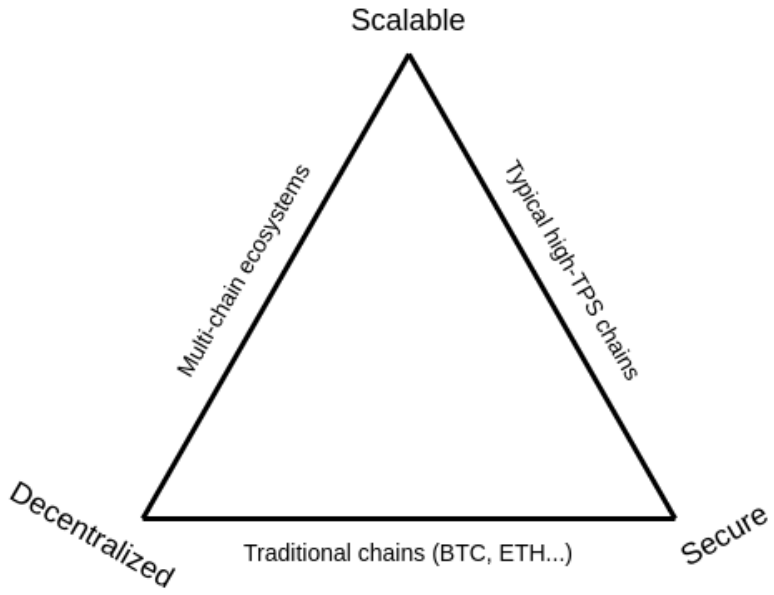


FIGURE 2. Taxonomy and comparison of blockchain scalability solutions.

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

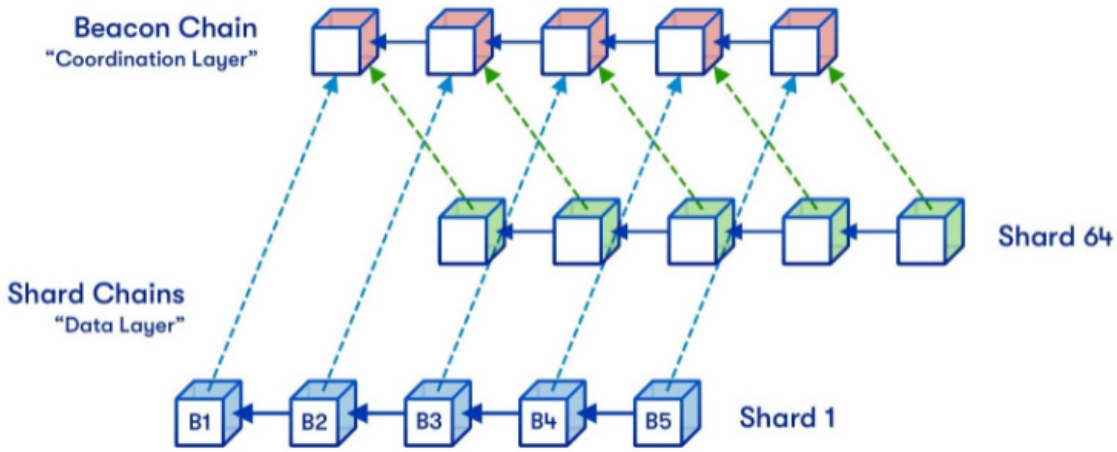
On chain Scaling (Layer 1)

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



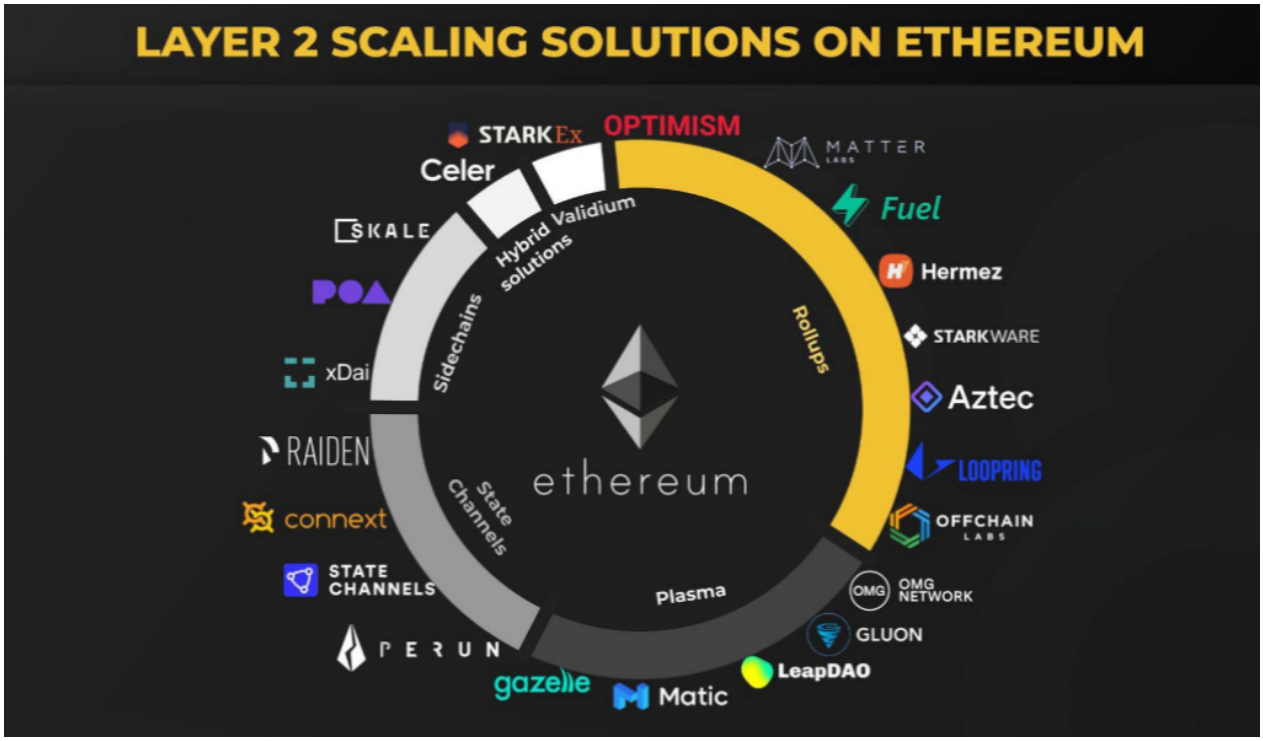
INTRODUCTION OF SHARDING

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.



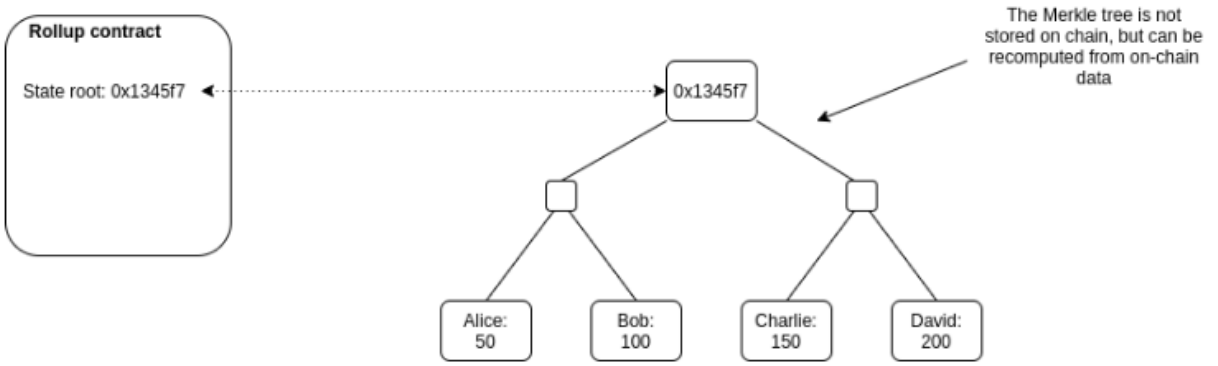
Rollups

Rollups are solutions that have

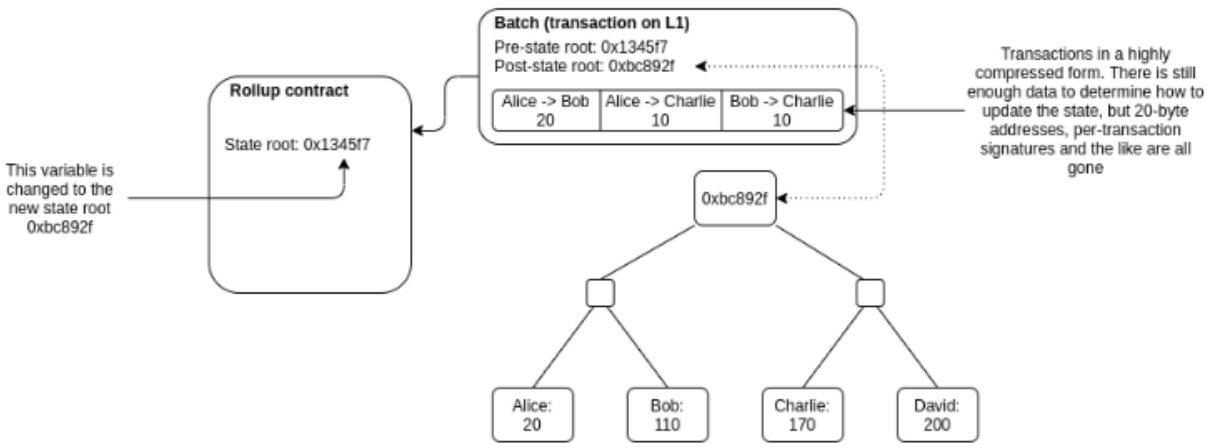
- transaction execution outside layer 1
- data or proof of transactions is on layer 1
- a rollup smart contract in layer 1 that can enforce correct transaction execution on layer 2 by using the transaction data on layer 1

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.

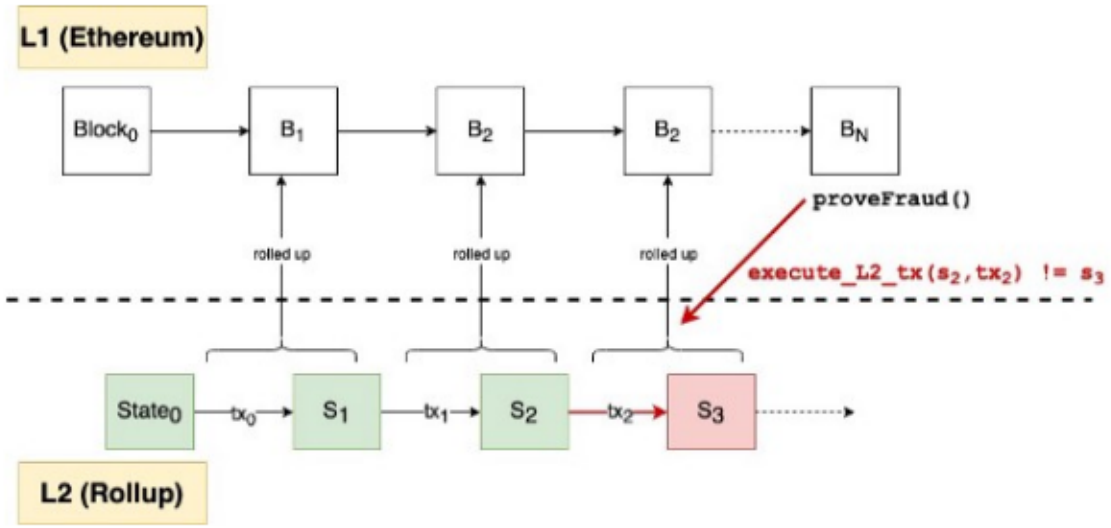


There are currently 2 types of rollups

- Zero Knowledge Proof rollups
- Optimistic rollups

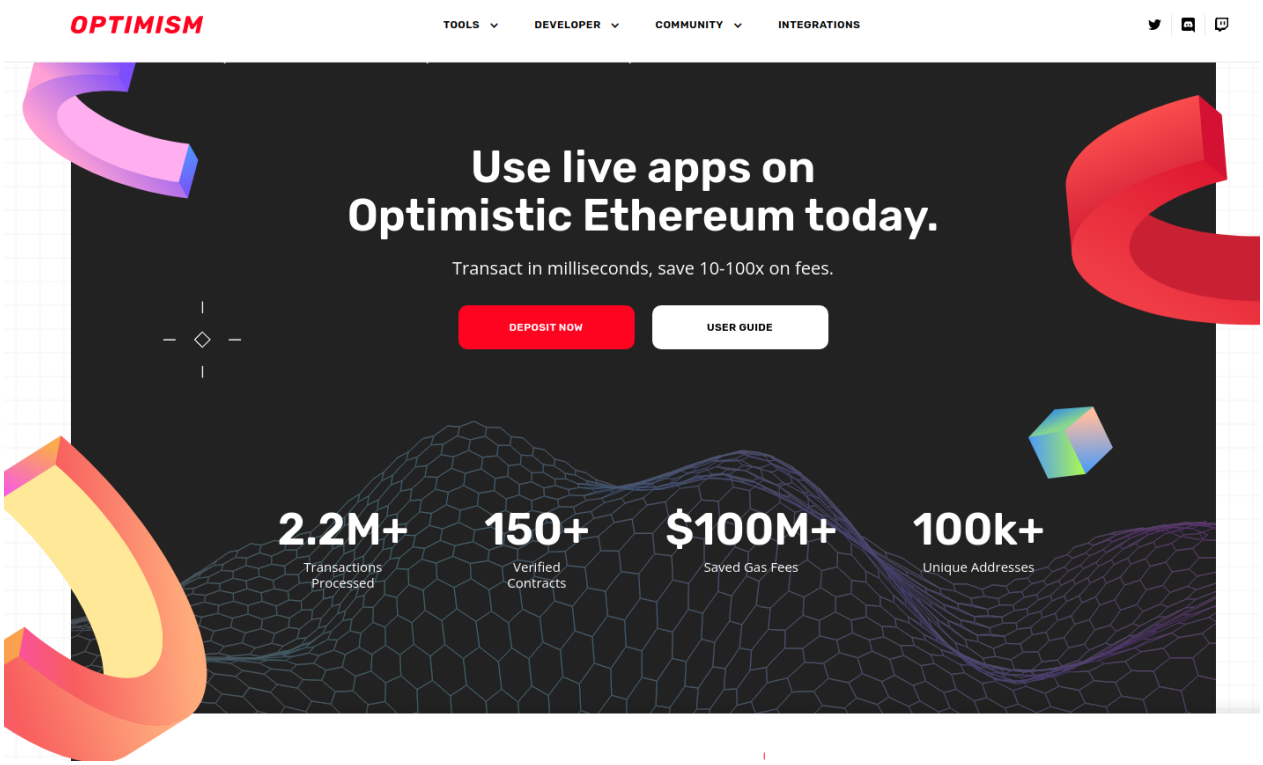
Optimistic Rollups

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.

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

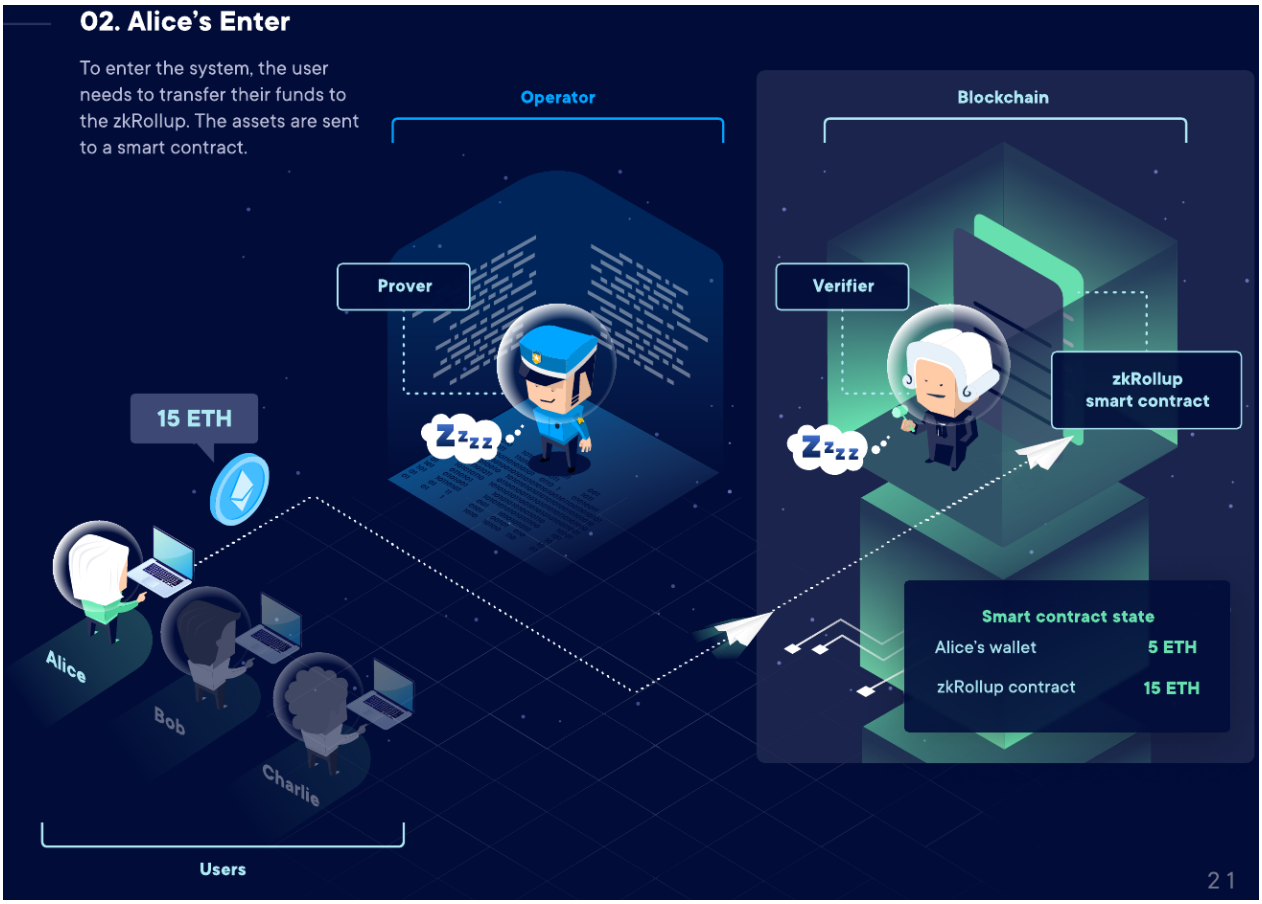
Note that in this context the proofs produced are often referred to as validity proofs since the zero knowledge aspect is not required.

ZK Rollup Process

From Ethworks

02. Alice's Enter

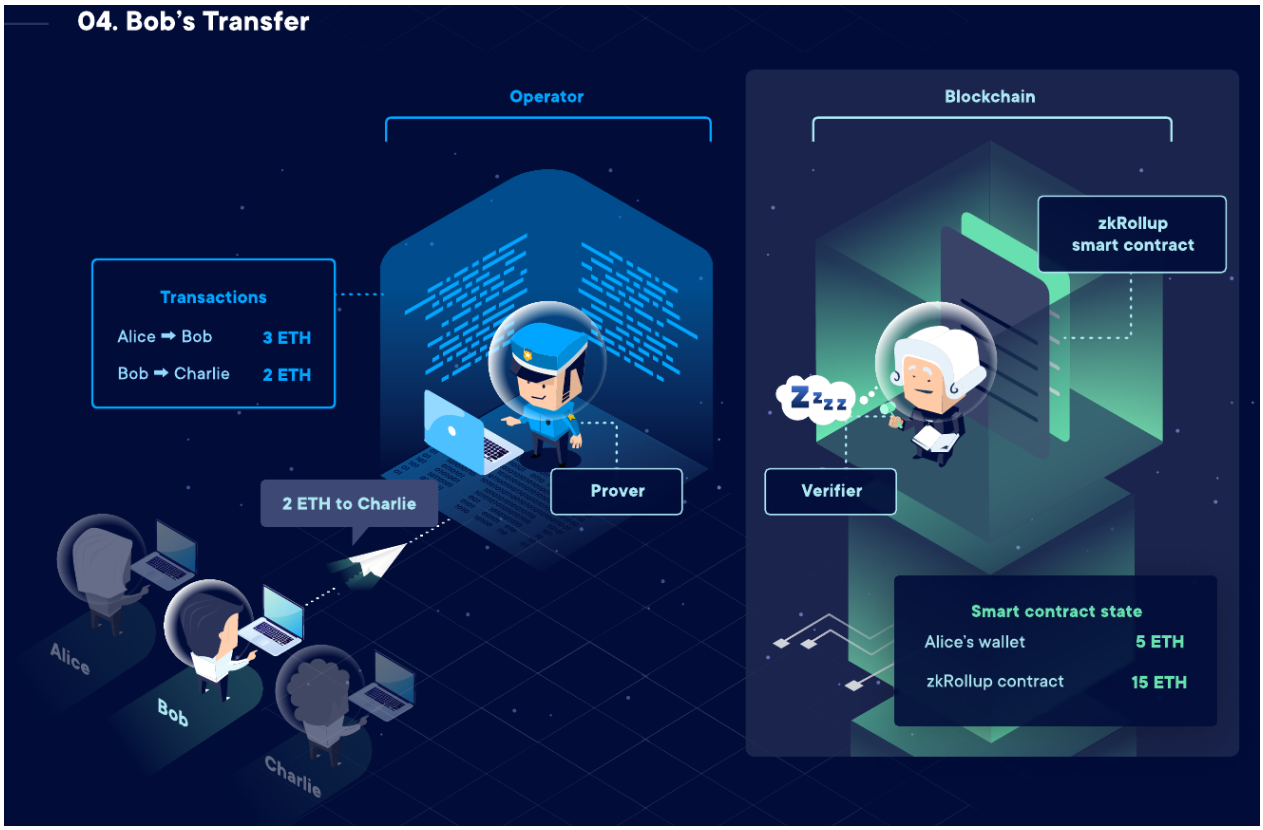
To enter the system, the user needs to transfer their funds to the zkRollup. The assets are sent to a smart contract.



03. Alice's Transfer

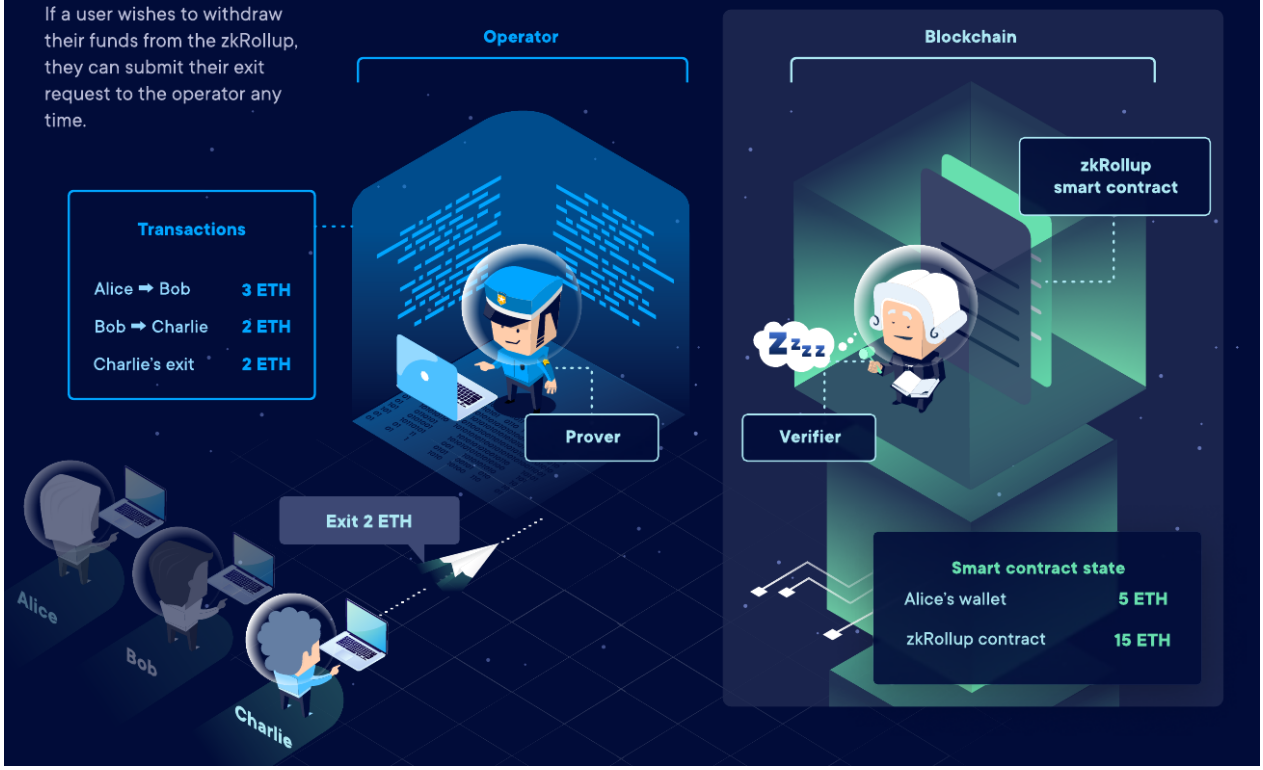
The user can now transfer their funds to another person. They sign the transaction and submit it to the zkRollup operator.

04. Bob's Transfer



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.



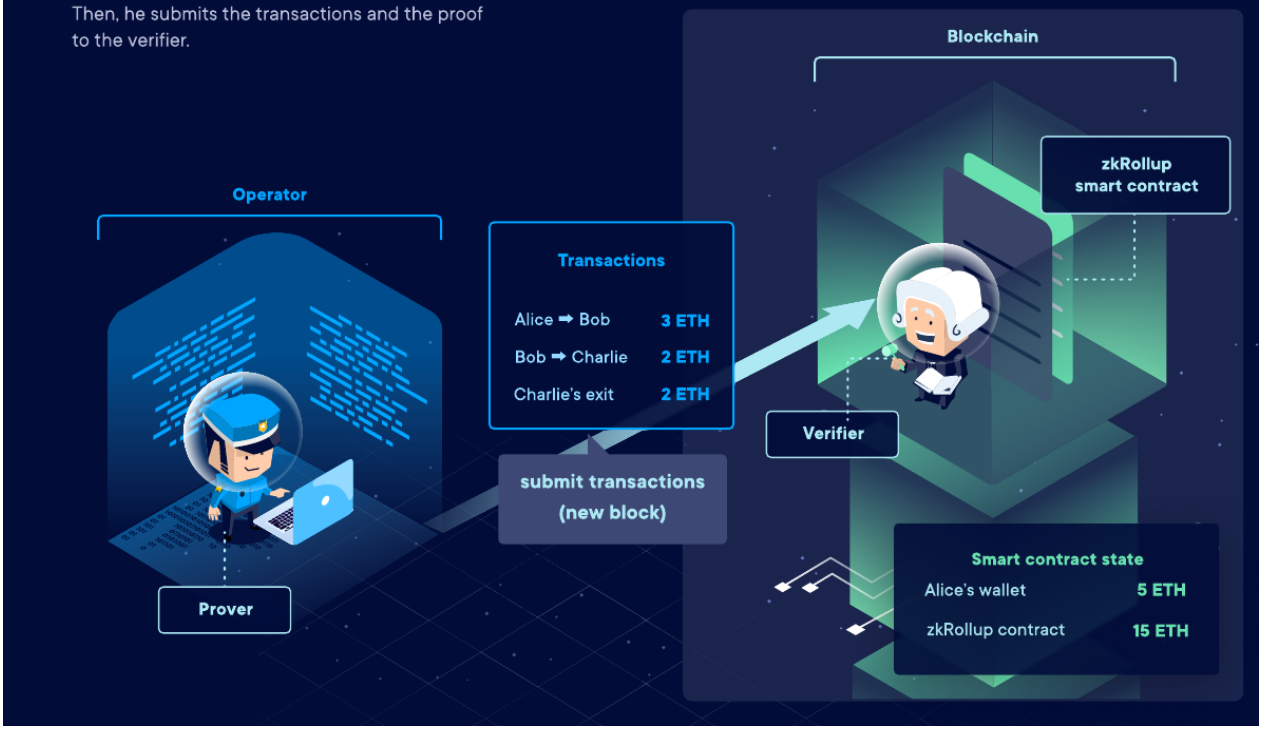
06. Collecting Transactions

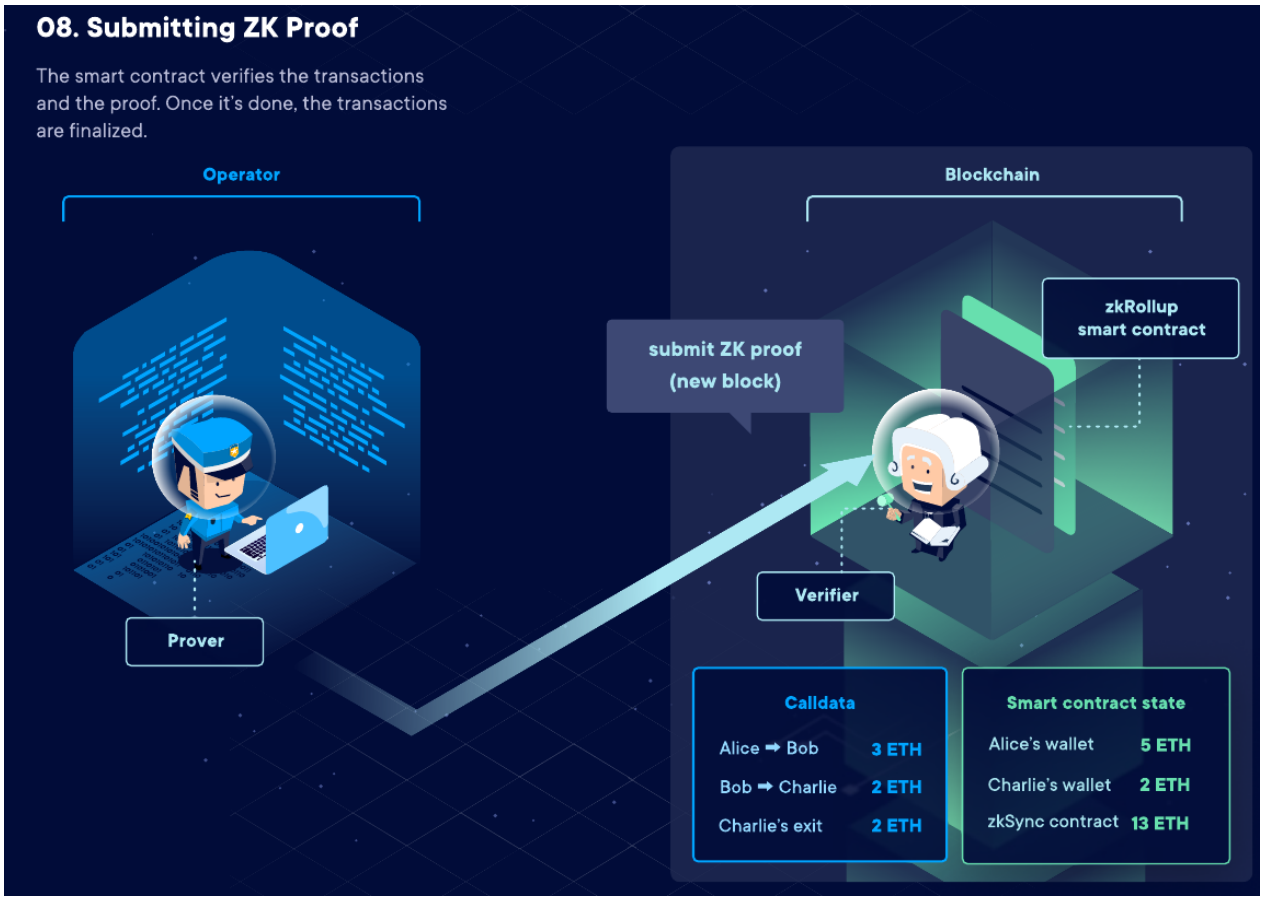
In the meantime, the operator collects transactions and exit requests from many users.

* 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)
Prover
Verifier
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

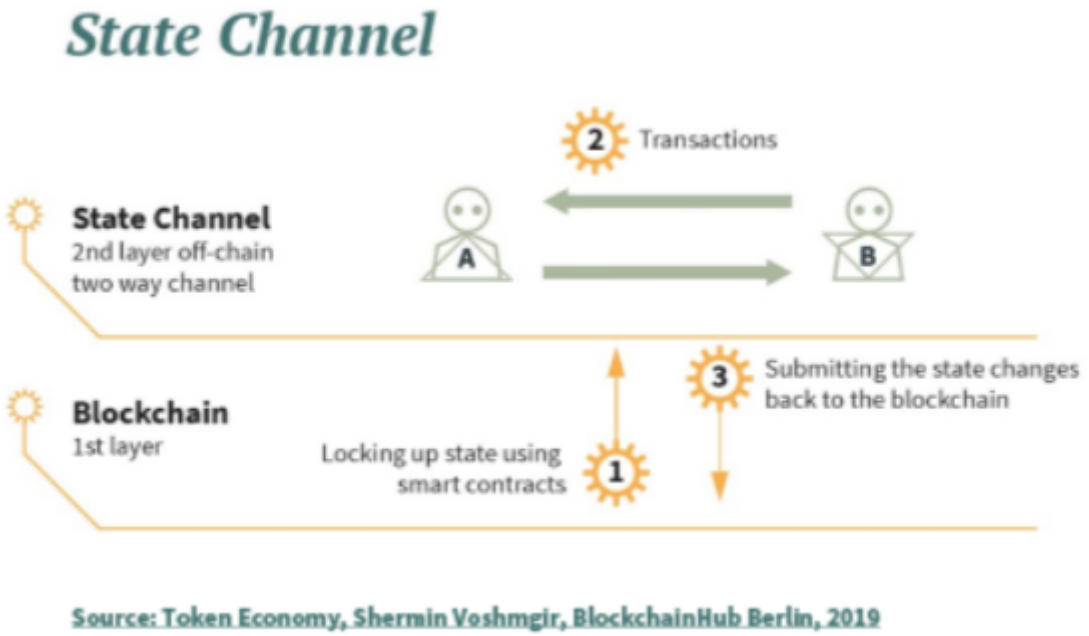
How does compression work?

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

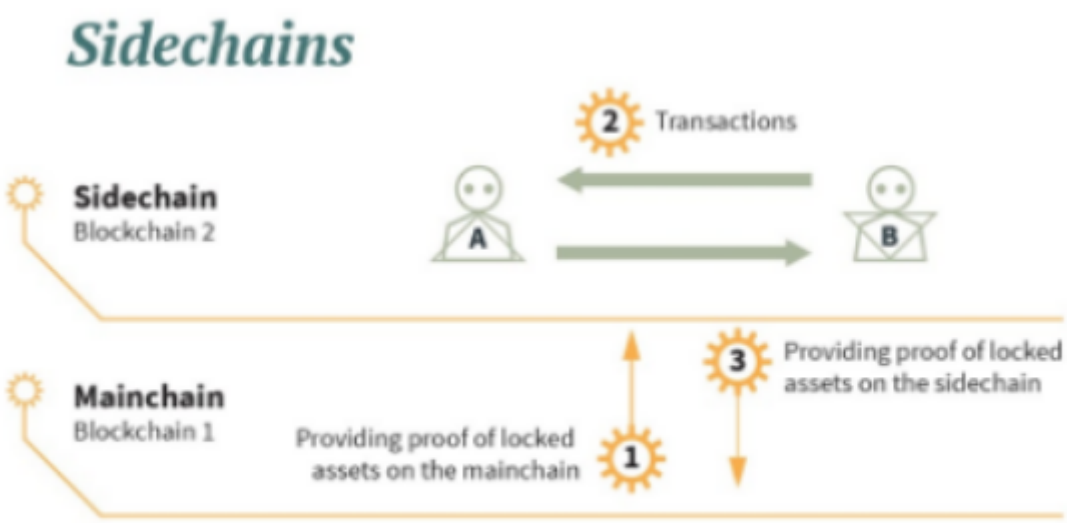
Parameter	Ethereum	Rollup
Nonce	~3	0
Gasprice	~8	0-0.5
Gas	3	0-0.5
To	21	4
Value	~9	~3
Signature	~68 (2 + 33 + 33)	~0.5
From	0 (recovered from sig)	4
Total	~112	~12

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



Source: Token Economy, Shermin Voshmgir, BlockchainHub Berlin, 2019



Source: Token Economy, Shermin Voshmgir, BlockchainHub Berlin, 2019

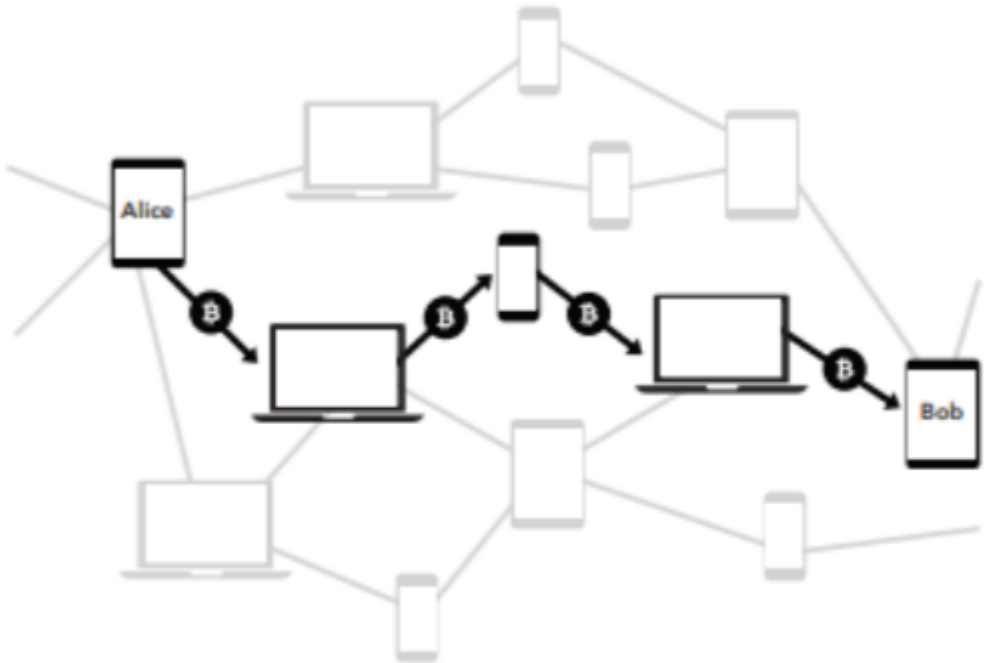
State channels

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

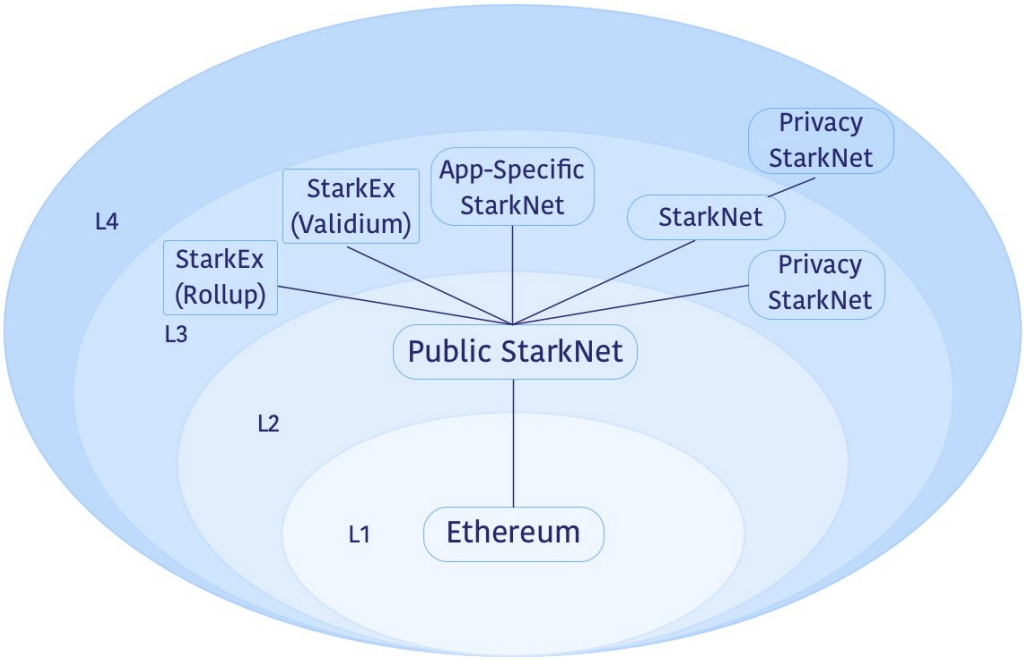
Plasma Chains

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

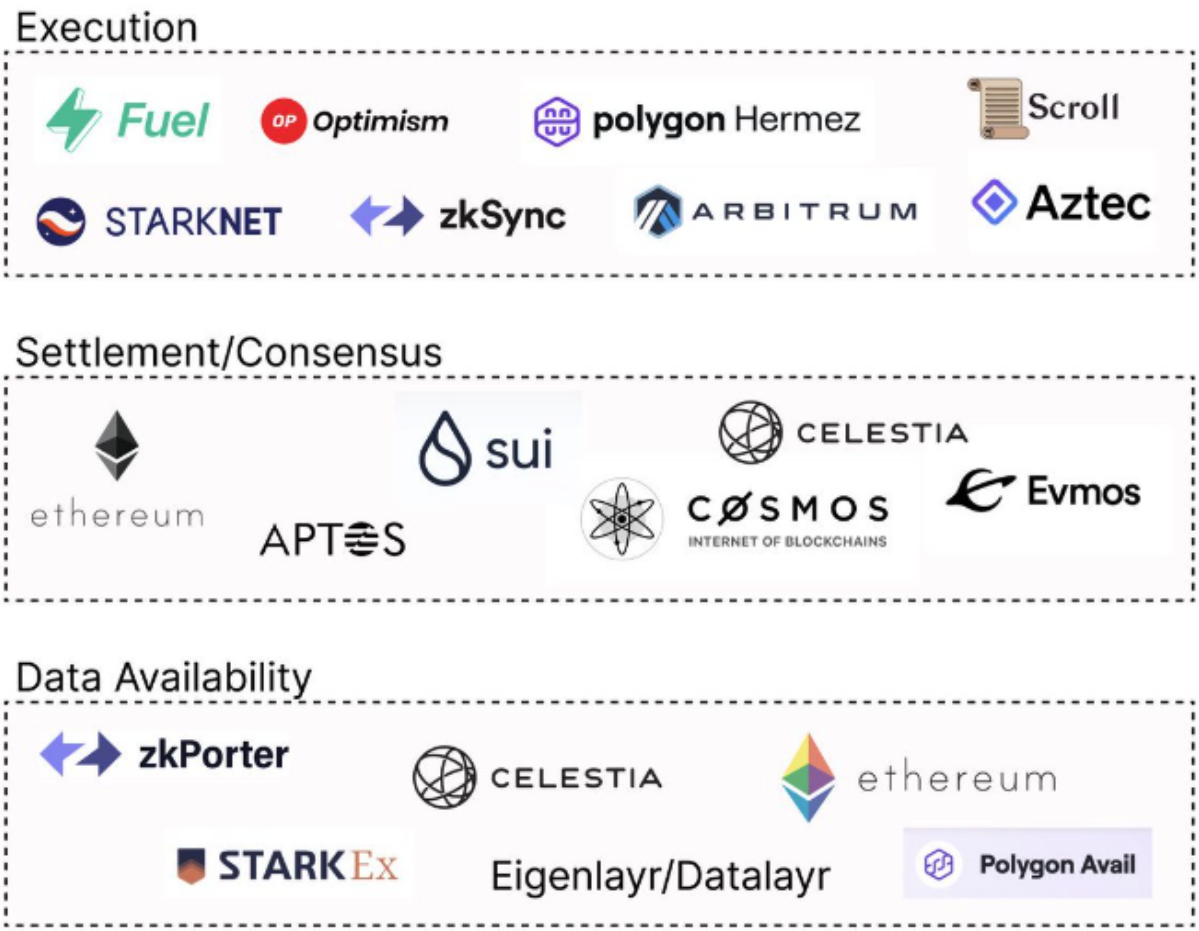
The rollup process can be extended further by the use of recursive proofs, essentially you can create a proof that proves a secondary proof (that proves another proof ....)

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.



Data Availability

Recall from Lesson 1 the concept of modular blockchains



If we follow the principle of separation of concerns, we can use a combination of blockchains to provide the functionality of a L1 and increase scalability.
For further details see Volt article
Also see the Ethereum documentation

"Data availability is the guarantee that the block proposer published all transaction data for a block and that the transaction data is available to other network participants."
This leads to the problem of ensuring that we have this guarantee for our particular chain (or L2 etc.)
In a later lesson we will look at the concept of stateless Ethereum and stateless clients.

For some zkRollups 3 modes are available :

- In ZK-Rollup mode data is published on-chain.
- In Validium mode data is stored off-chain.
- Volition is a hybrid data availability mode, where the user can choose whether to place data on-chain or off-chain.

	Validity Proofs		Fault Proofs
Data On-Chain	Volition	ZK-Rollup	Optimistic Rollup
Data Off-Chain		Validium	Plasma

STARKWARE

Data availabilty vs data retreivability

There is a distinction between data availabilty and data retreivability

Data availability refers to the ability of nodes to download transaction data for a block while it is being proposed for addition to the chain.

Data retrievability is the ability of nodes to retrieve historical information from the blockchain.

For Ethereum data availability is more of a concern.

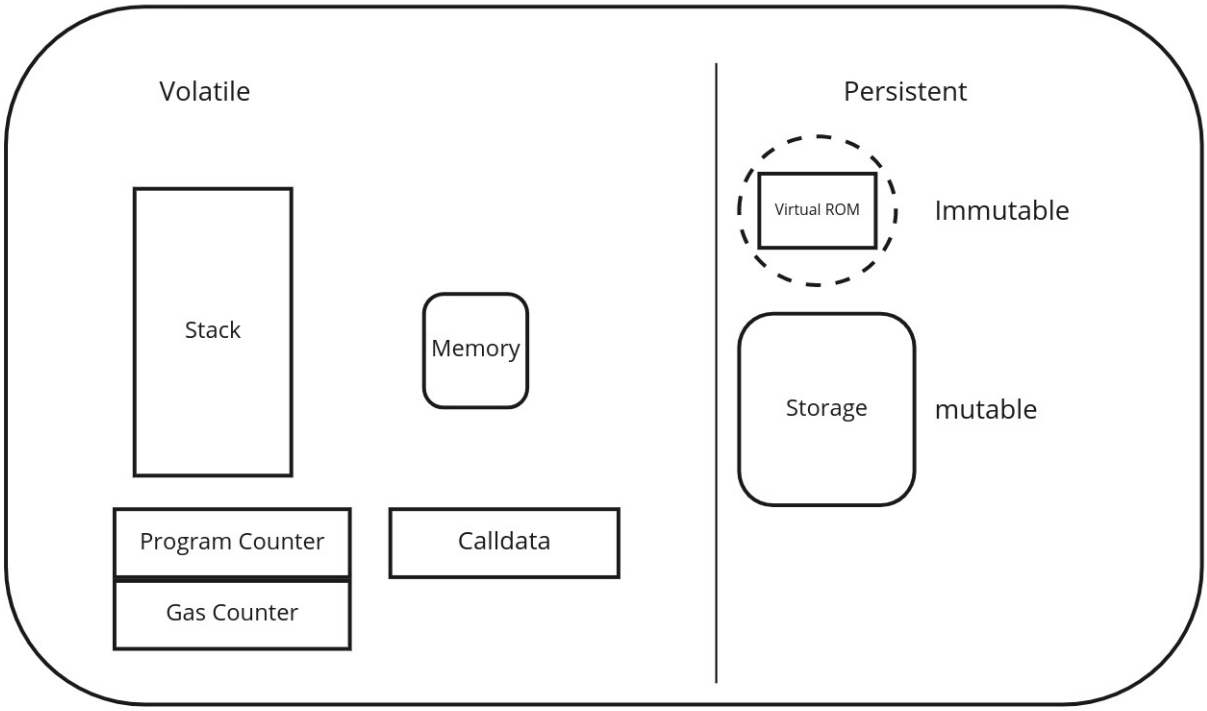
Suggestions for who would store historical data

So who will store this data? Some ideas:

- Individual and institutional volunteers
- Block explorers (etherchain.org, etherscan.io, amberdata.io...) would definitely store all of it, because providing the data to users is their business model.
- Rollup DAOs nominating and paying participants to store and provide the history relevant to their rollup
- History could be uploaded and shared through torrents
- Clients could voluntarily choose to each store a random 0.05% of the chain history (using erasure coding so you'd need many clients to drop offline at the same time to lose even a single piece).
- Clients in the Portal Network could store random portions of chain history, and the Portal Network would automatically direct requests for data to the nodes that have it.
- Historical data storage could be incentivized in-protocol.
- Protocols like TheGraph can create incentivized marketplaces where clients pay servers for historical data with Merkle proofs of its correctness. This creates an incentive for people and institutions to run servers that store historical data and provide it on demand.

Source: A step-by-step roadmap for scaling rollups with calldata expansion and sharding — HackMD (ethereum.org) (note: some of the information in that post is now outdated)

zkEVM Solutions



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

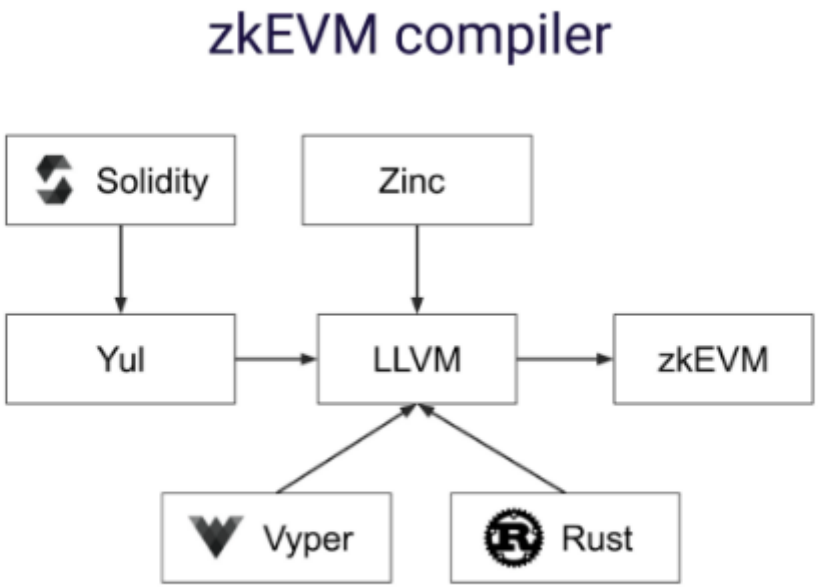
AppliedZKP divides proofs into two types:

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.

Matter Labs zkEVM (zkSync)

zkEVM is a virtual machine that executes smart contracts in a way that is compatible with zero-knowledge-proof computation.
zk-EVM keeps EVM semantics, but is also ZK-friendly and takes on traditional CPU architectures.
zkSync's zkEVM is not a replica of EVM but is newly designed to run 99% of Solidity contracts and ensure that it works properly under a variety of conditions (including

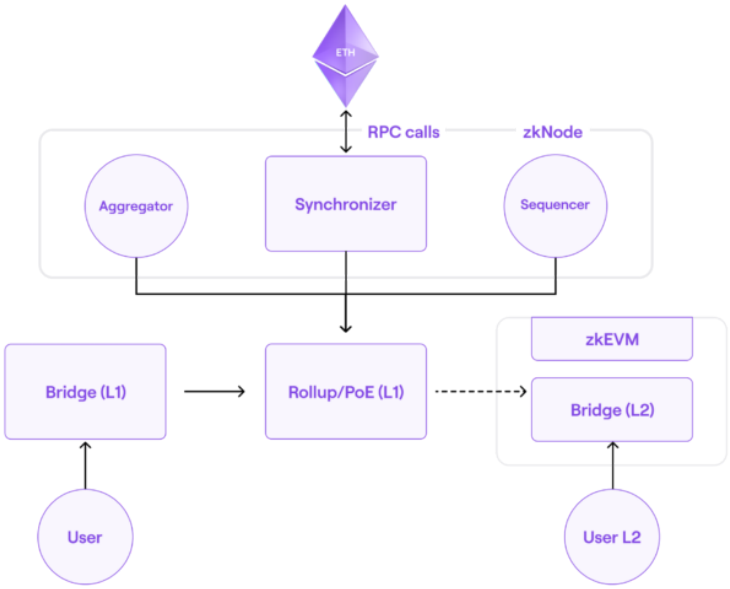
rollbacks and exceptions). At the same time, zkEVM can be used to efficiently generate zero-knowledge proofs in the circuit.



The circuit implementation of Matter Labs uses TinyRAM to implement ordinary opcodes, such as ADD, PUSH, etc.; opcodes that consume a lot of gas, such as SHA256/keccak, implement this circuit especially; finally, Matter Labs uses recursive aggregation technology to aggregate all proofs into one proof.

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.