Lesson 15 - Scalability

The scalability trilemma

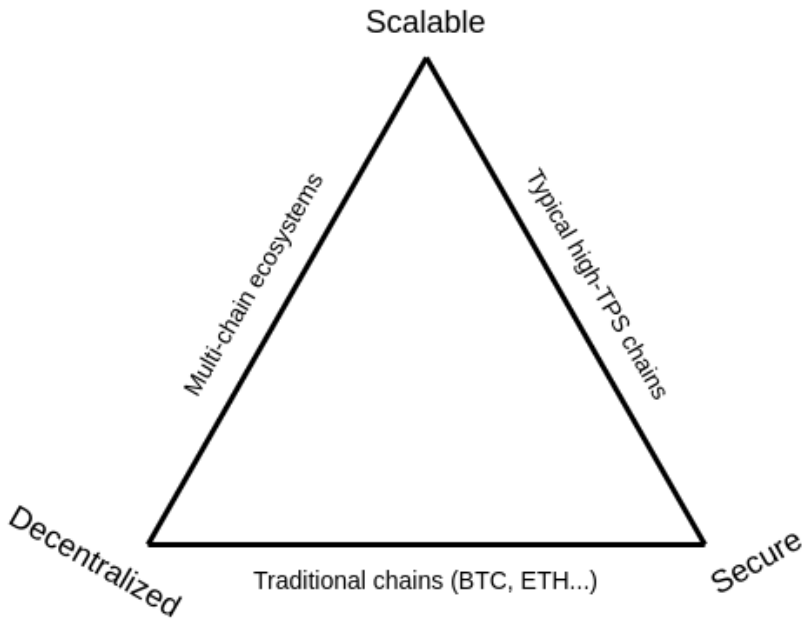


FIGURE 2. **Taxonomy and comparison of blockchain scalability solutions.**

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.

Solutions

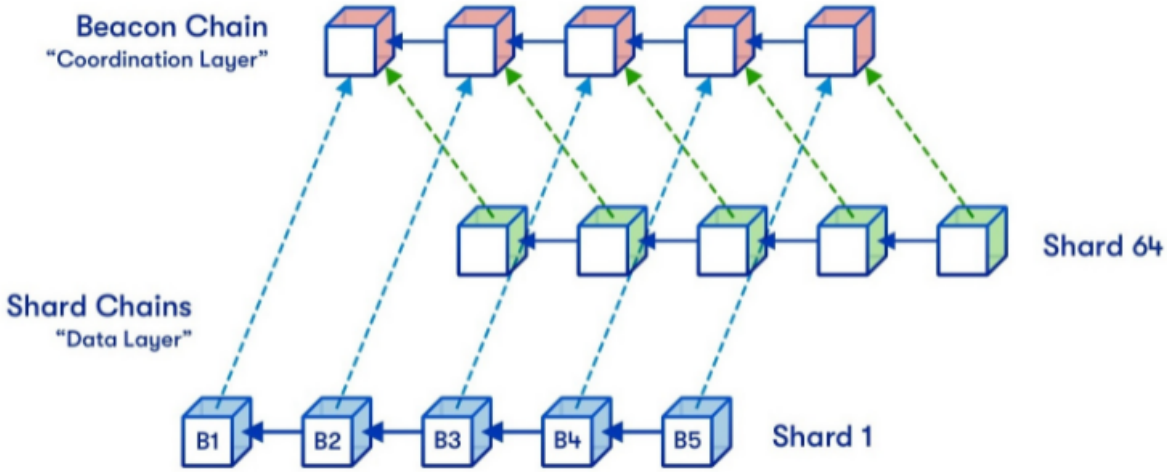
On chain Scaling (Layer 1)

Changing the Consensus Mechanism

Using DPoS - EOS
For example moving from Proof of Work to Proof of Stake - Ethereum

Sharding

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



This will follow the merge of Mainnet with the Beacon Chain, probably in 2022.

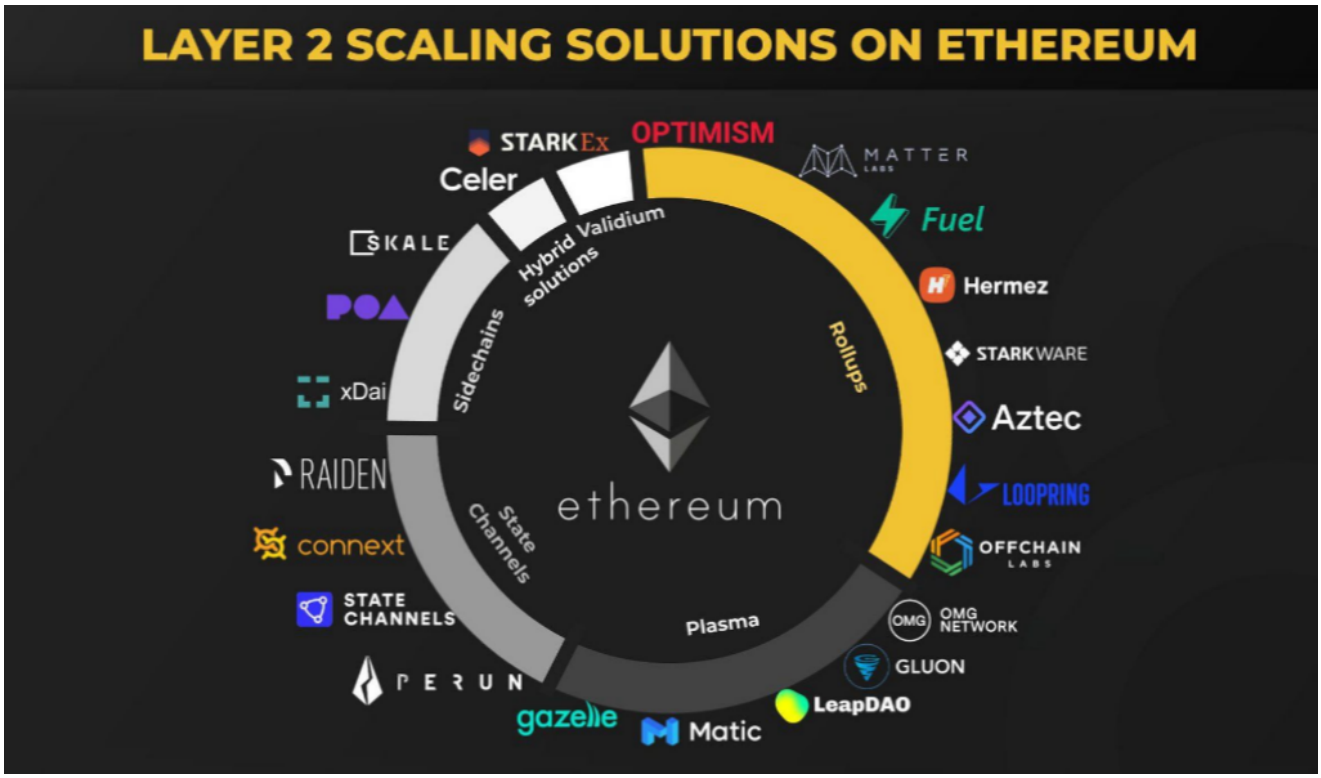
Introduction of Sharding

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.



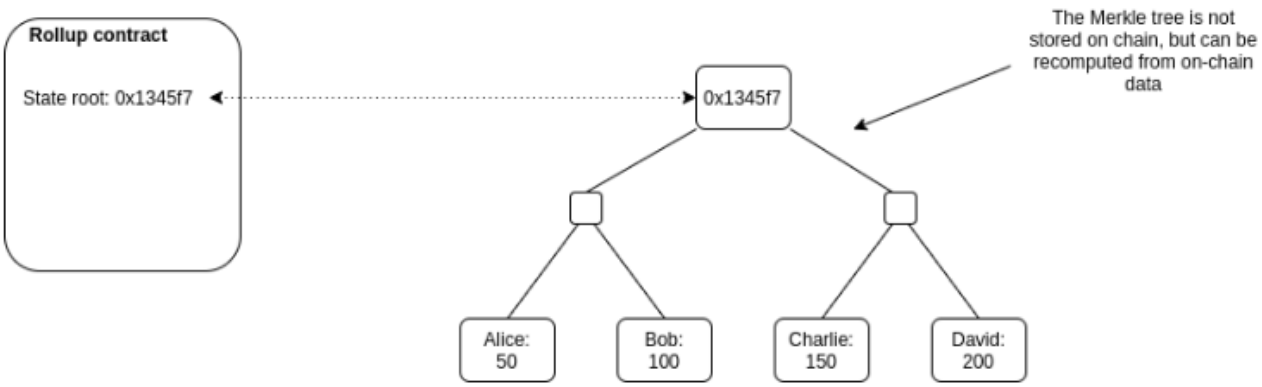
Rollups

Rollups are solutions that have

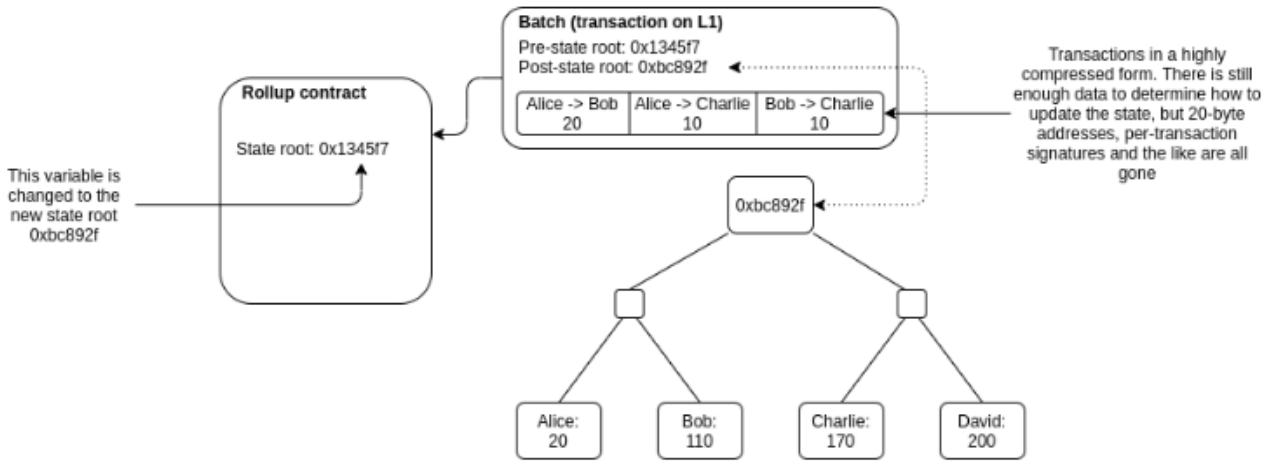
- transaction execution outside layer 1
- data or proof of transactions is on layer 1
- a rollup smart contract in layer 1 that can enforce correct transaction execution on layer 2 by using the transaction data on layer 1

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.

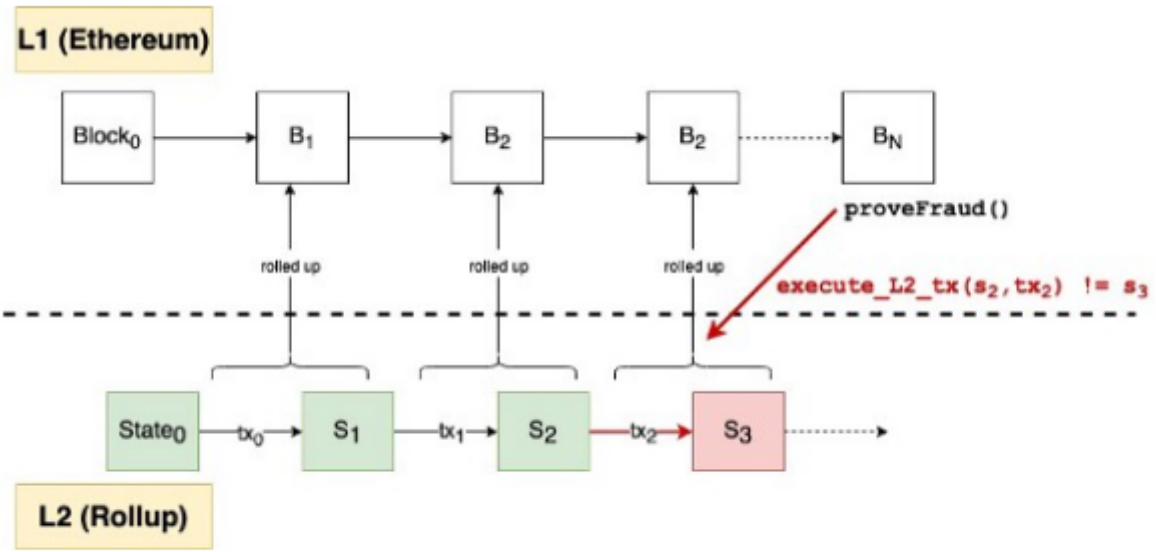


There are currently 2 types of rollups

- Zero Knowledge Proof rollups
- Optimistic rollups

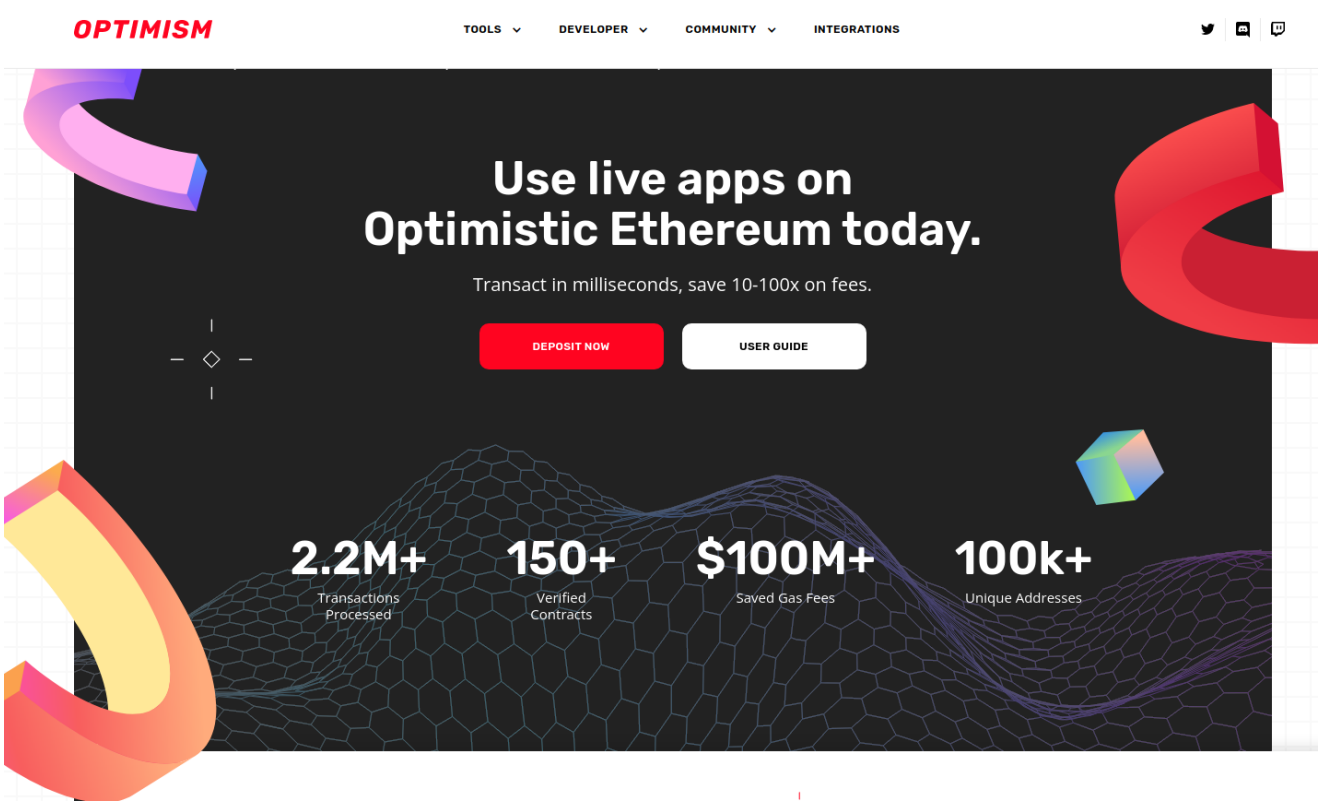
Optimistic Rollups

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



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.

Zero Knowledge Proof Rollups

See Ethworks Report

An overview from Ethereum
The ZK-Rollup scheme consists of two types of users: transactors and relayers.

- Transactors create their transfer and broadcast the transfer to the network. The transfer data consists of an indexed "to" and "from" address, a value to transact, the network fee, and nonce. A shortened 3 byte indexed version of the addresses reduces processing resource needs. The value of the transaction being greater than or less than zero creates a deposit or withdrawal respectively. The smart contract records the data in two Merkle Trees; addresses in one Merkle Tree and transfer amounts in another.
- Relayers collect a large amount of transfers to create a rollup. It is the relayers job to generate the SNARK proof. The SNARK proof is a hash that represents the delta of the blockchain state. State refers to "state of being." SNARK proof compares a snapshot of the blockchain before the transfers to a snapshot of the blockchain after the transfers (i.e. wallet values) and reports only the changes in a verifiable hash to the mainnet.

It is worth noting that anyone can become a relayer so long as they have staked the required bond in the smart contract. This incentivises the relayer not to tamper with or withhold a rollup.

ZK Rollup Process

From Ethworks

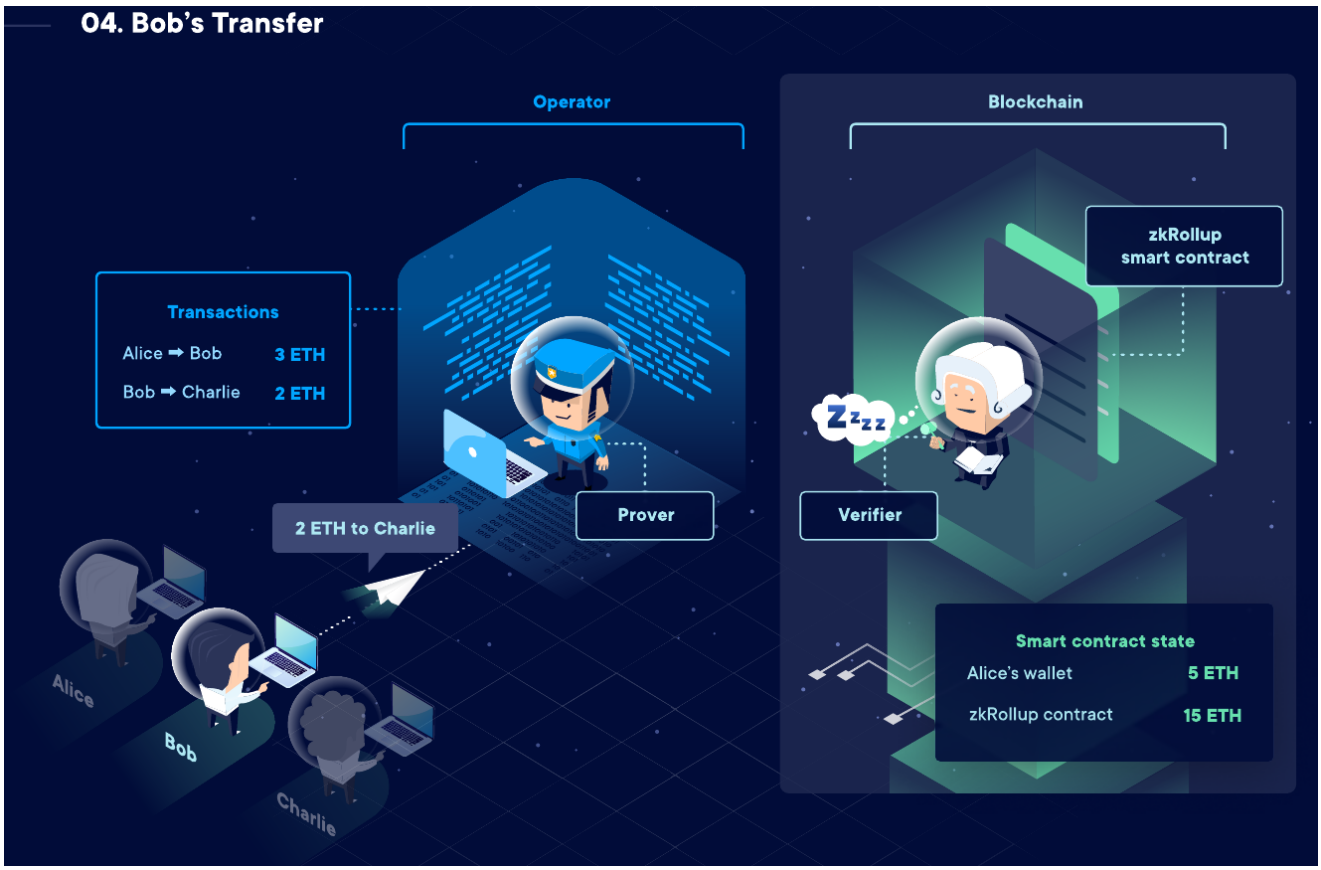
01. Start

While the operator server has an embedded prover, the smart contract is equipped with a pre-generated verifier.

02. Alice's Enter

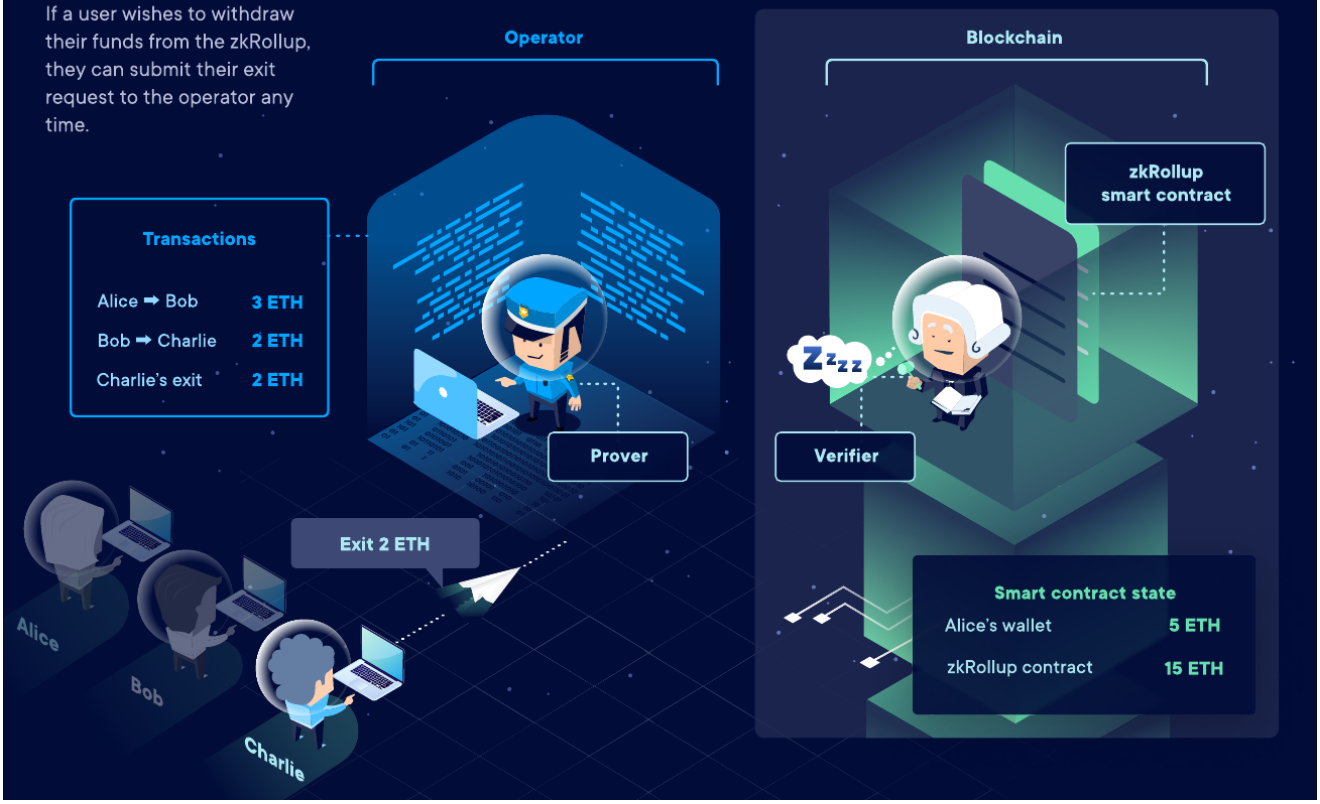
To enter the system, the user needs to transfer their funds to the zkRollup. The assets are sent to a smart contract.


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


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

05. Charlie's Exit

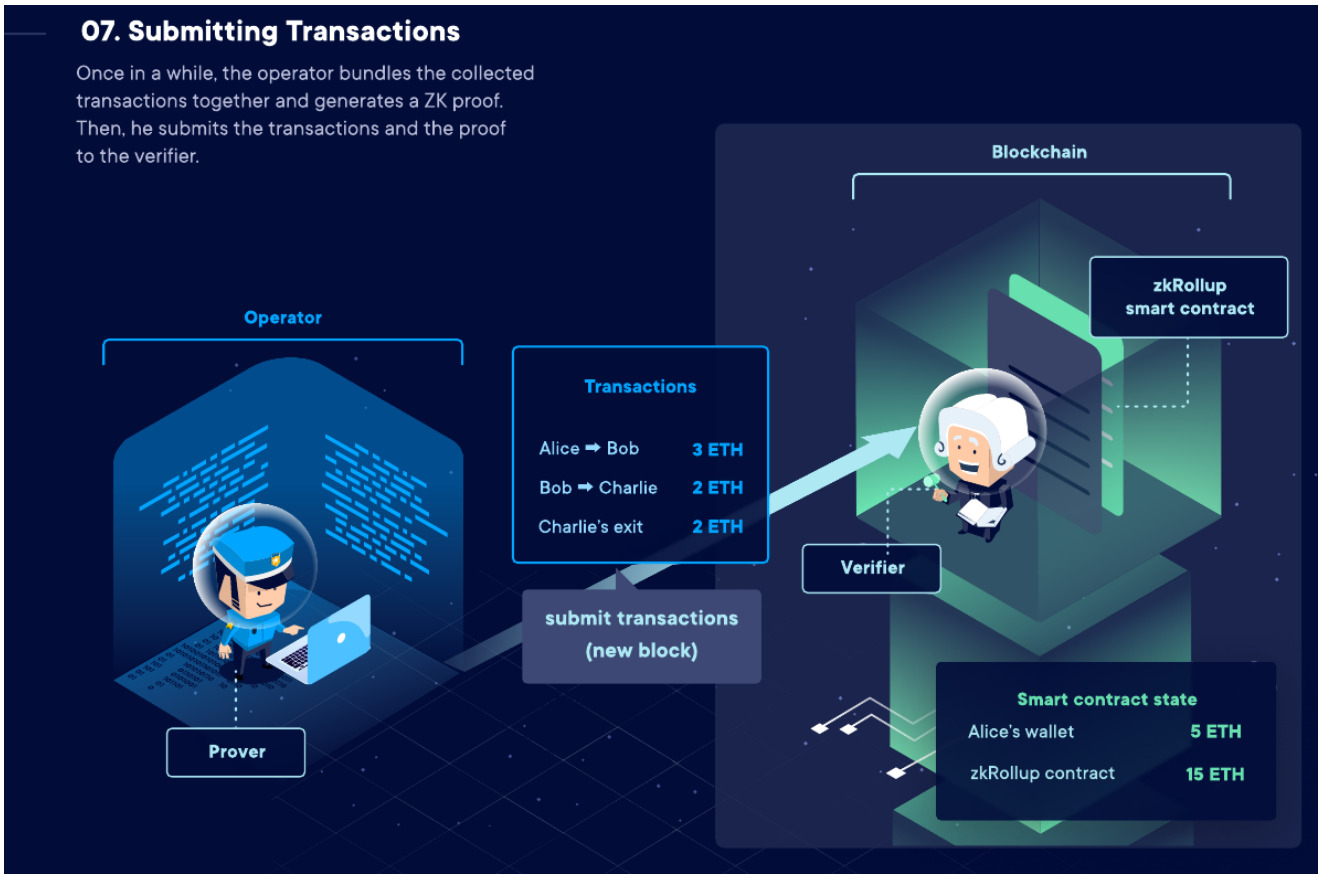
If a user wishes to withdraw their funds from the zkRollup, they can submit their exit request to the operator any time.

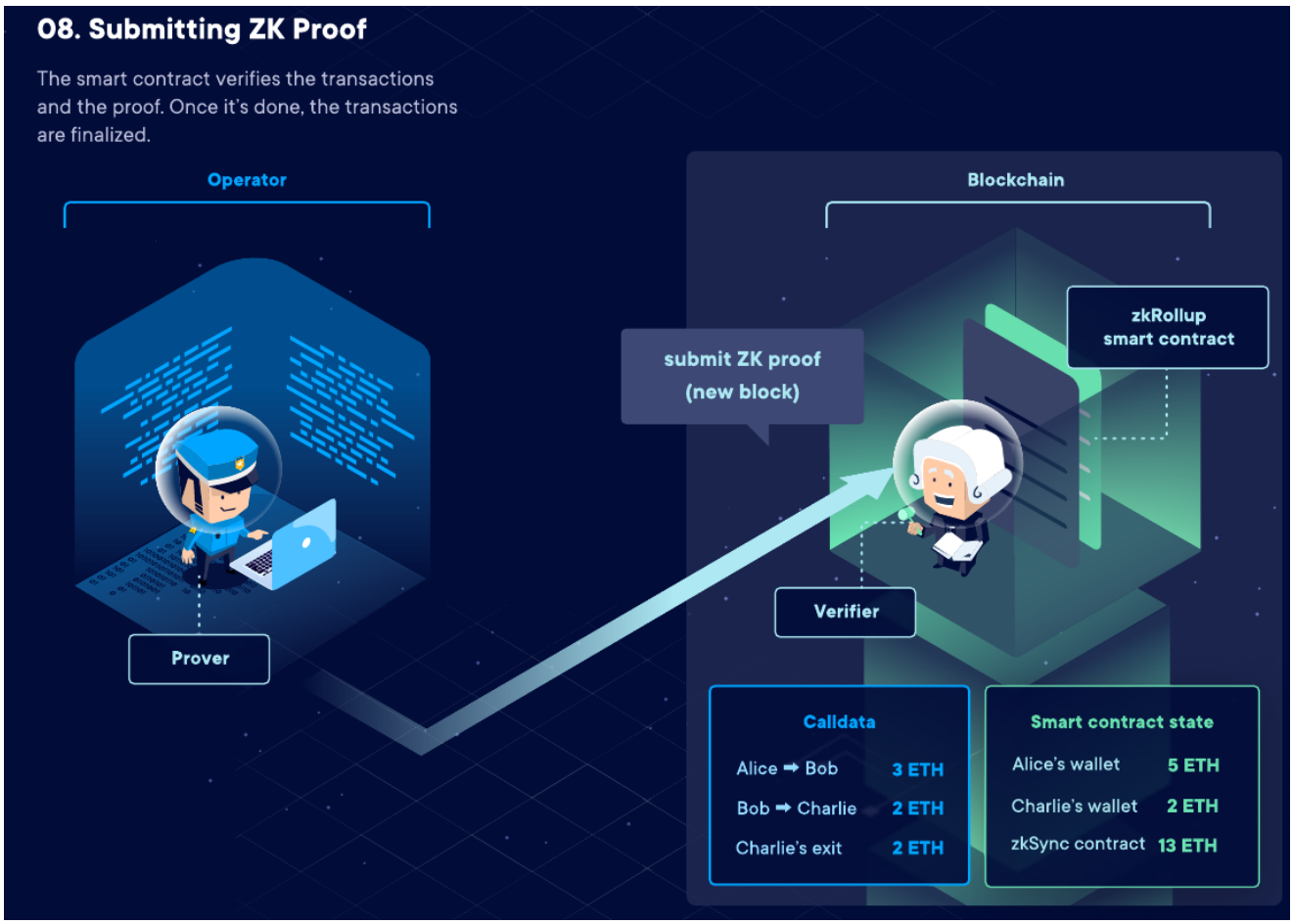


06. Collecting Transactions

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.





Comparison of the types

Property	Optimistic rollups	ZK rollups

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)

Proofs

Optimistic rollups use fraud proofs: the rollup contract keeps track of its entire history of state roots and the hash of each batch.
If anyone discovers that one batch had an incorrect post-state root, they can publish a proof to chain, proving that the batch was computed incorrectly. The contract verifies the proof, and reverts that batch and all batches after it.

ZK rollups use validity proofs: every batch includes a cryptographic proof called a ZK-SNARK (eg. using the PLONK protocol), which proves that the post-state root is the

correct result of executing the batch. No matter how large the computation, the proof can be very quickly verified on-chain.

Transaction Compression

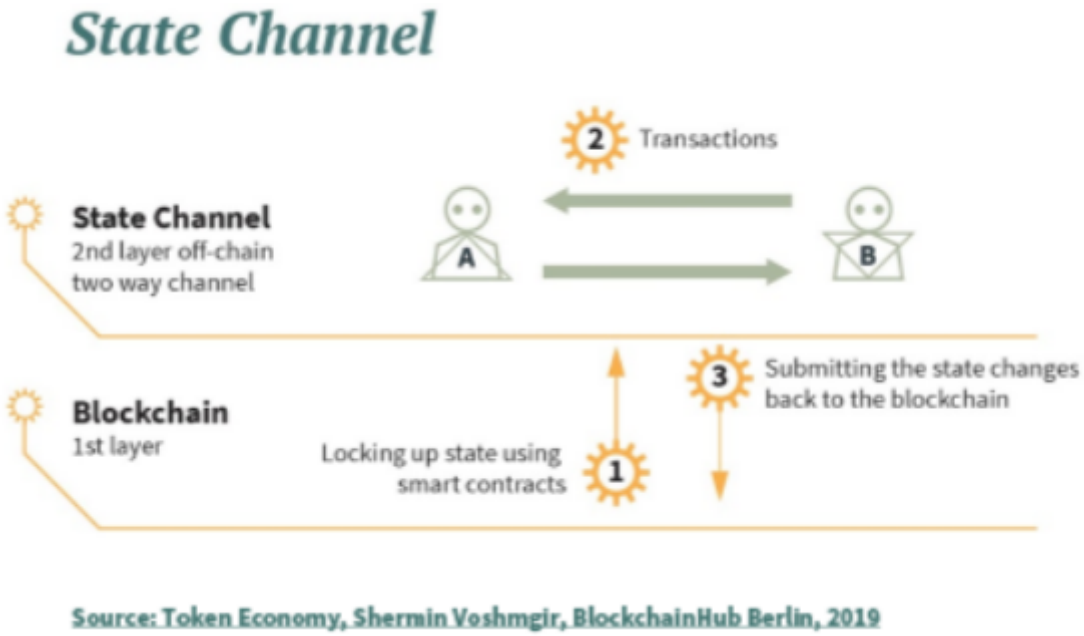
How does compression work?

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

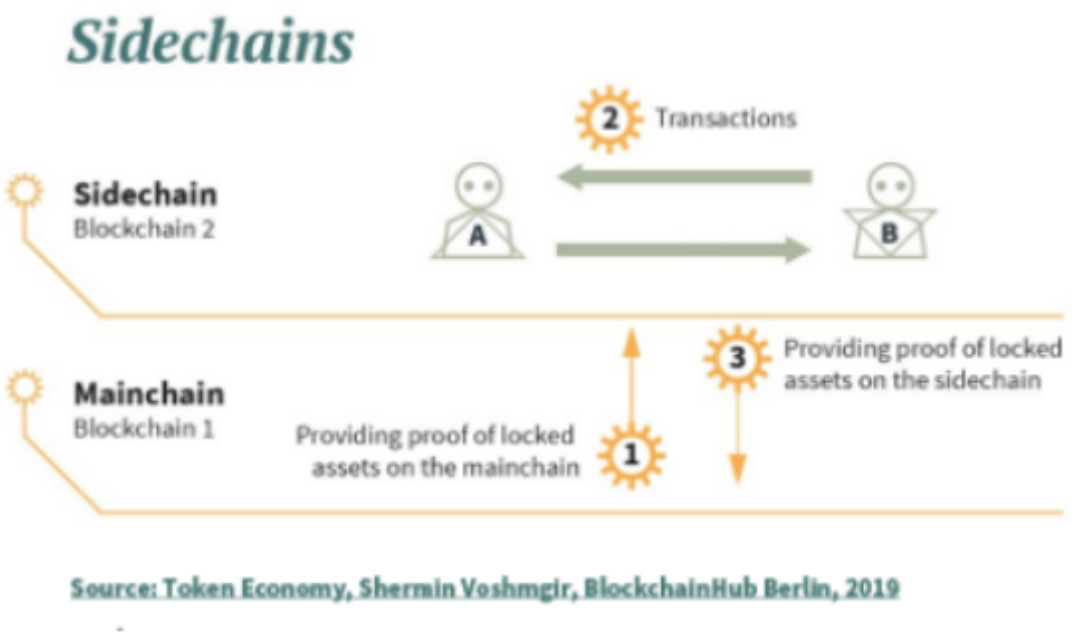
Parameter	Ethereum	Rollup
Nonce	~3	0
Gasprice	~8	0-0.5
Gas	3	0-0.5
To	21	4
Value	~9	~3
Signature	~68 (2 + 33 + 33)	~0.5
From	0 (recovered from sig)	4
Total	~112	~12

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



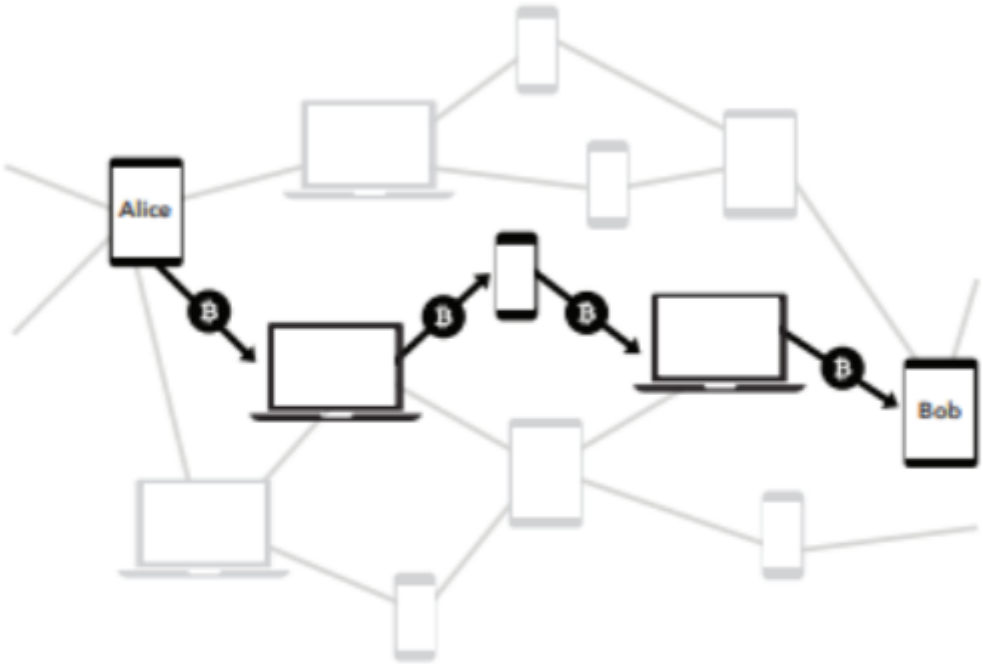
State channels

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.

- Raiden Network

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
- XDai
  - xDai bridge to move between DAI and xDAI
  - Omni bridge to move ERC20 between xDai and Ethereum or Binance snart chain

Advantages

- Easy to implement with existing technology
- EVM compatible

Disadvantages

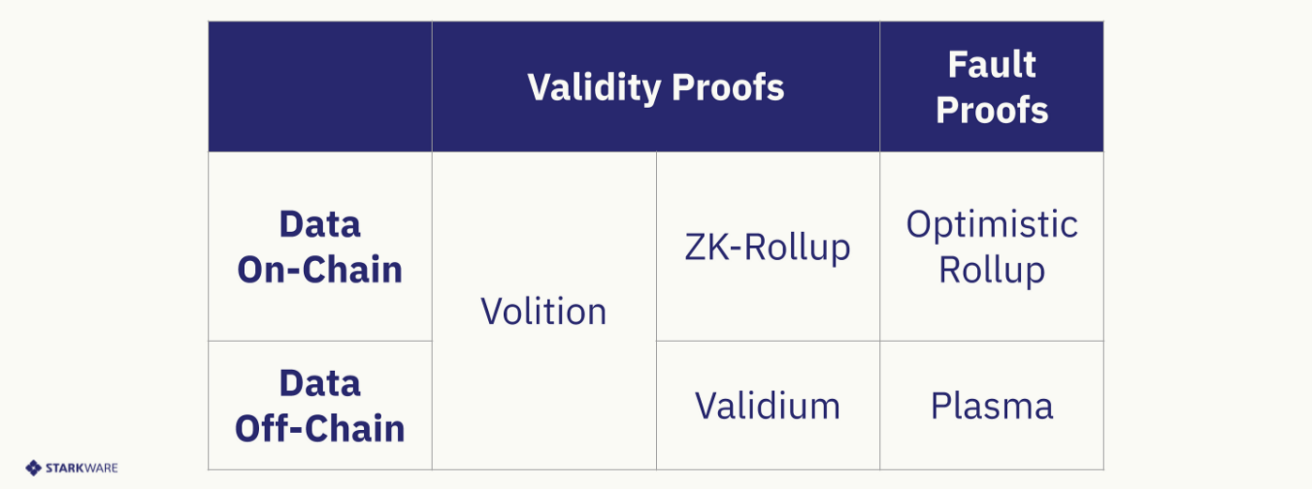
- Consensus mechanism may not be better
- Not secured by layer 1, so more susceptible to fraud
- Probably less decentralised

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.

Data Availability

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.



See Docs

StarkNet is currently in ZK-Rollup mode (see above). This means that upon the acceptance of a state update on-chain, the state diff between the previous and new state is sent as calldata to Ethereum.

This data allows anyone that observes Ethereum to reconstruct the current state of StarkNet. Note that to update the StarkNet state on L1, it suffices to send a valid proof — without information on the transactions or particular changes that this update caused. Consequently, more information must be provided in order to allow other parties to locally track StarkNet's state.

The state diffs contain information on every contract whose storage was updated and additional information on contract deployments. Those differences are sent as `uint256[]` array as part of the calldata, and are encoded as follows:

- Number of cells that encode contract deployments

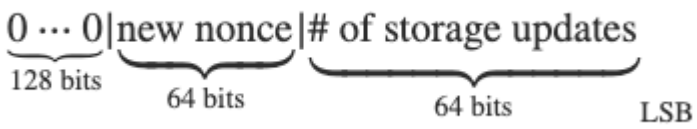
- For each deployed contract, we have:

    - `contract_address` - the address of the deployed contract
    - `contract_hash` - the hash of the contract
    - `len(constructor_call_data)` - the number of arguments to the contract constructor
    - `constructor_call_data` - the list of arguments to the constructor

- Number of contracts whose storage is updated

- For each such contract, we have:

    - `contract_address` - the address of the contract
    - `num_of_storage_updates` - number of storage updates
    - `nonce, num_of_storage_updates` - a uint256 value that encodes both the number of storage updates for that contract and the updated nonce:



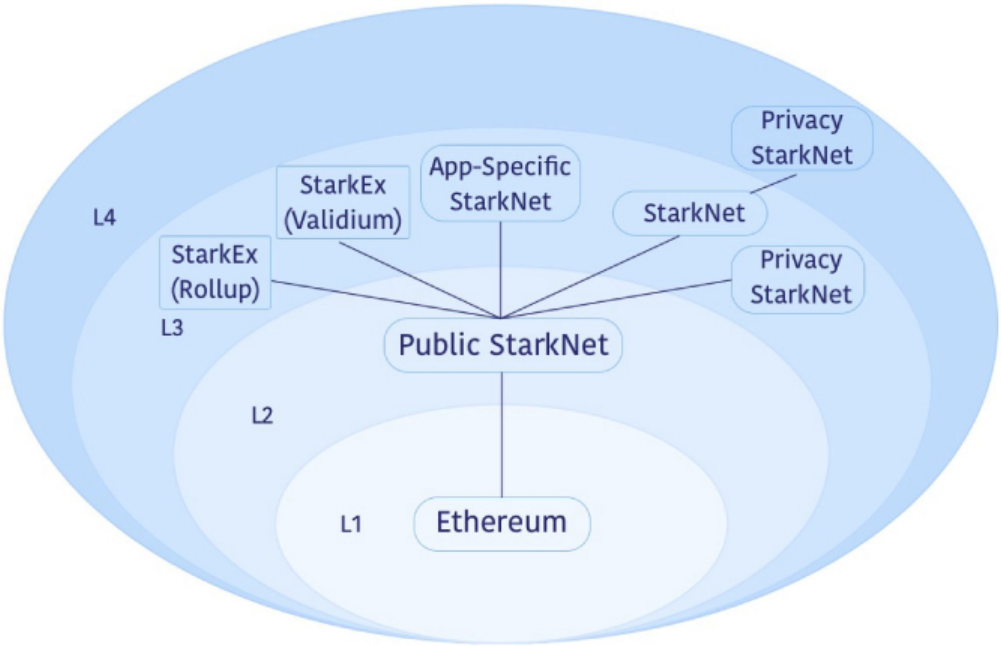
- For each storage update:

    - `key` - the address inside the contract's storage where the value is updated
    - `value` - the new value

The data described above is sent across several Ethereum transactions, each holding a part of this array as calldata. Each new StarkNet block has its associated state diff transactions.
You can find the code for extracting this data from Ethereum in Pathfinder's repo.
Pathfinder is the first StarkNet full node implementation.
You may also take a look at the python script which extracts the same information.

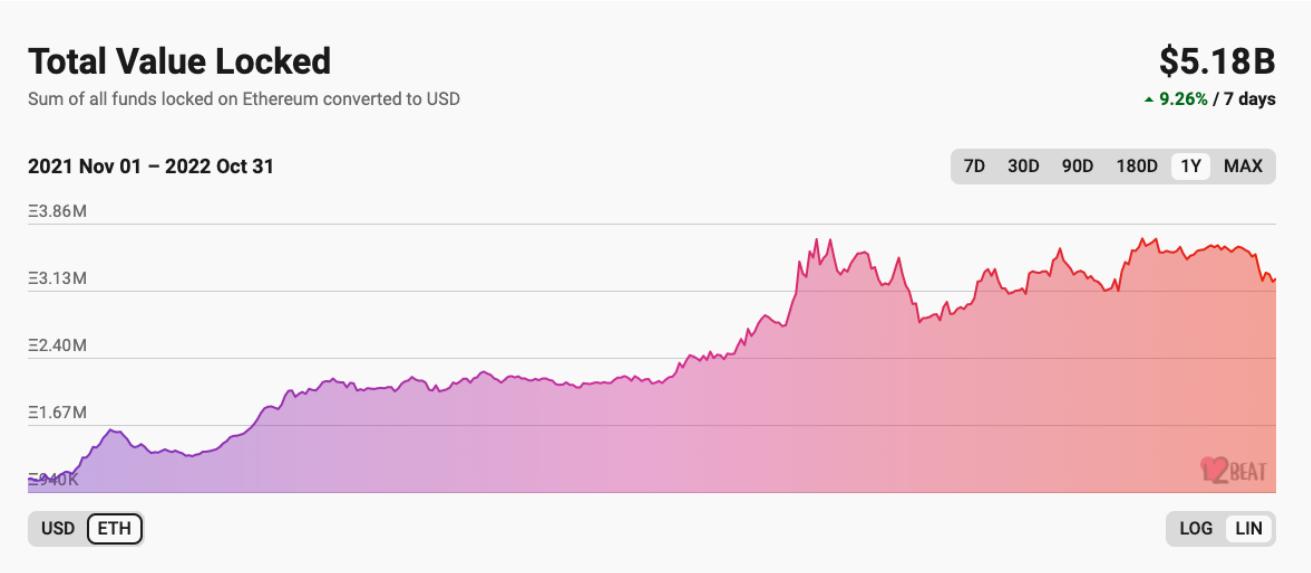
L3 and L4 ?

See Fractal scaling article



L2 Statistics

See L2 Beat

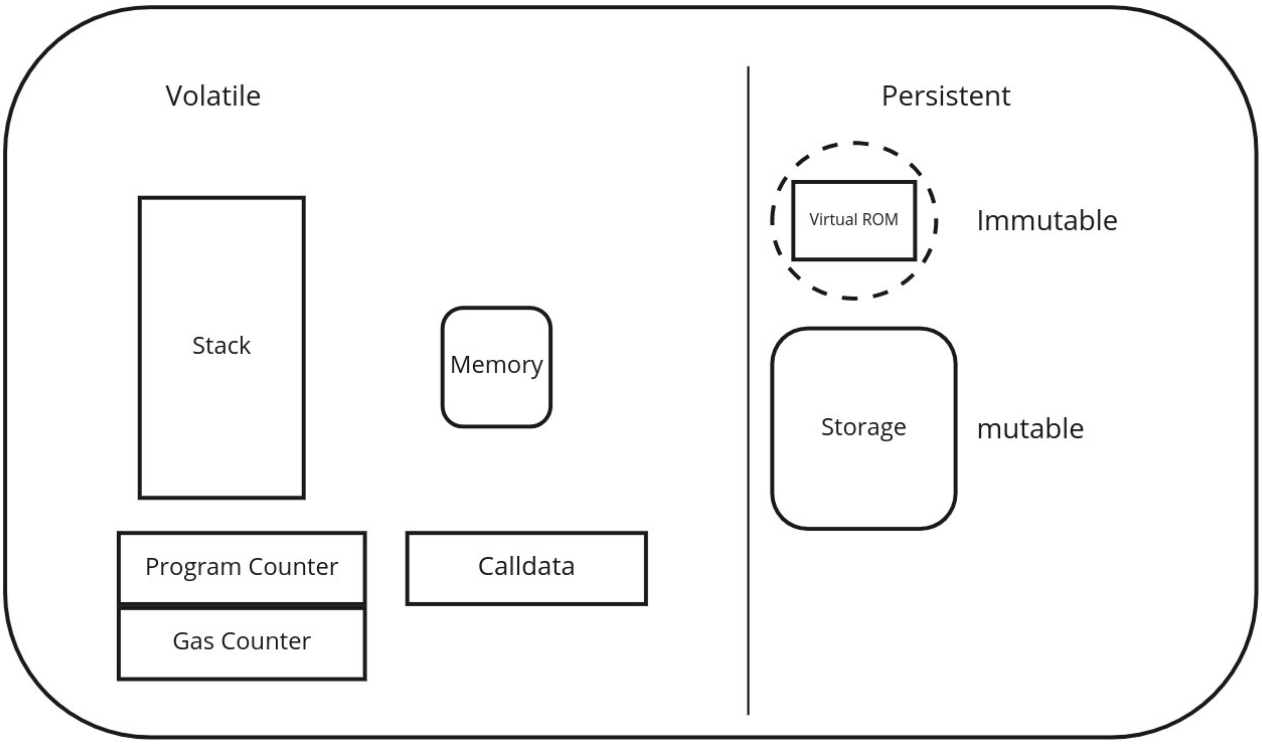


#	NAME	TVL	BREAKDOWN	7D CHANGE	MARKET SHARE	PURPOSE	TECHNOLOGY
1	Arbitrum One	$2.65B		▲ 9.93%	51.21%	Universal	Optimistic Rollup
2	Optimism	$1.61B		▲ 9.91%	31.09%	Universal	Optimistic Rollup
3	dYdX	$376M		▲ 2.81%	7.27%	Exchange	ZK Rollup
4	Metis Andromeda	$138M		▲ 2.29%	2.67%	Universal	Optimistic Chain
5	Loopring	$127M		▲ 10.94%	2.47%	Tokens, NFTs, AMM	ZK Rollup
6	zkSync	$63.27M		▲ 13.96%	1.22%	Tokens, NFTs	ZK Rollup
7	Immutable X	$61.23M		▲ 9.50%	1.18%	NFT, Exchange	Validium
8	ZKSpace	$41.94M		▲ 7.14%	0.81%	Tokens, NFTs, AMM	ZK Rollup
9	Boba Network	$30.65M		▲ 10.60%	0.59%	Universal	Optimistic Rollup
10	Sorare	$25.10M		▲ 25.77%	0.48%	NFT, Exchange	Validium
11	rhino.fi	$21.93M		▼ 0.78%	0.42%	Exchange	Validium
12	Aztec Connect	$8.65M		▲ 18.94%	0.17%	Private DeFi	ZK Rollup
13	ApeX	$5.10M		▲ 10.91%	0.10%	Exchange	Validium
14	Arbitrum Nova	$4.55M		▲ 25.66%	0.09%	Universal	Optimistic Chain
15	Aztec	$3.17M		▲ 14.85%	0.06%	Private payments	ZK Rollup
16	OMG Network	$3.03M		▲ 0.21%	0.06%	Payments	Plasma
17	StarkNet	$2.82M		▲ 56.05%	0.05%	Universal	ZK Rollup
18	ZKSwap 1.0	$1.88M		▲ 6.97%	0.04%	Payments, AMM	ZK Rollup
19	Polygon Hermez	$321K		▲ 4.10%	0.01%	Payments	ZK Rollup

Other 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



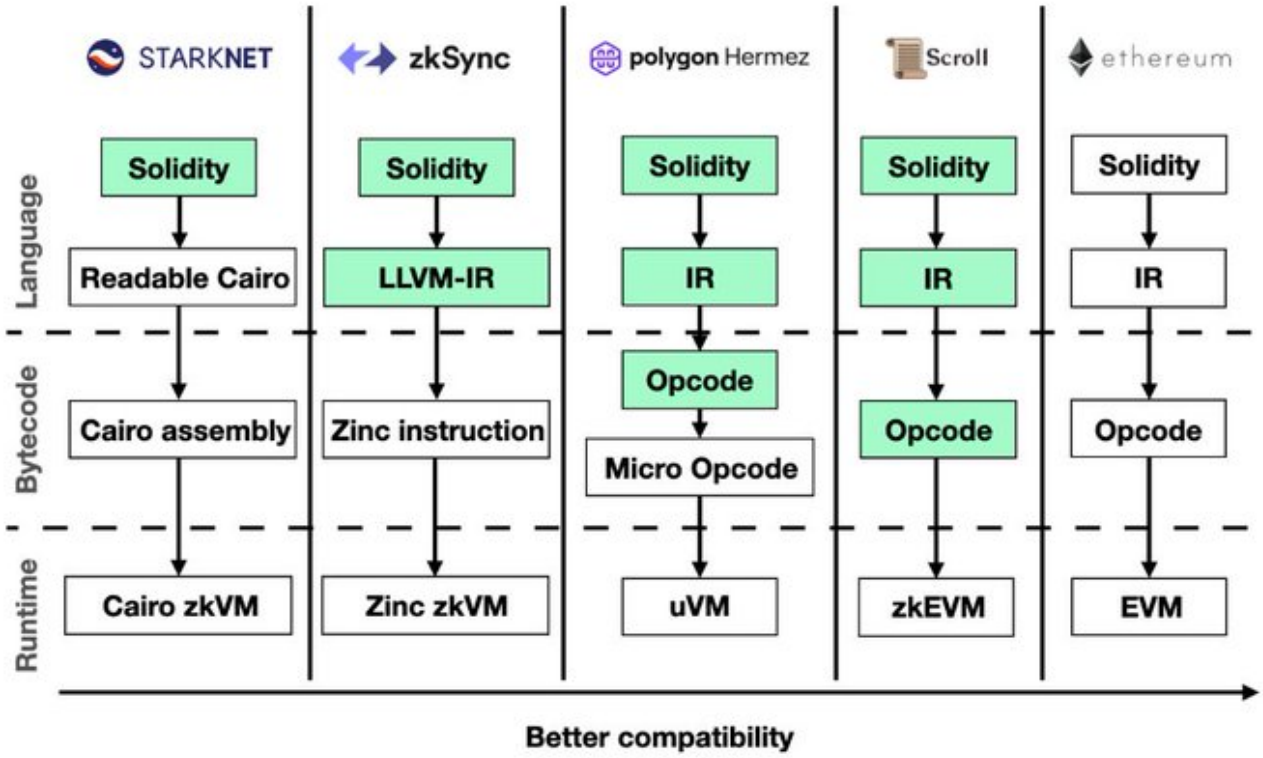
The opcode of the EVM needs to interact with Stack, Memory, and Storage during execution. There should also be some contexts, such as gas/program counter, etc. Stack is only used for Stack access, and Memory and Storage can be accessed randomly.

Design challenges in designing a zkEVM

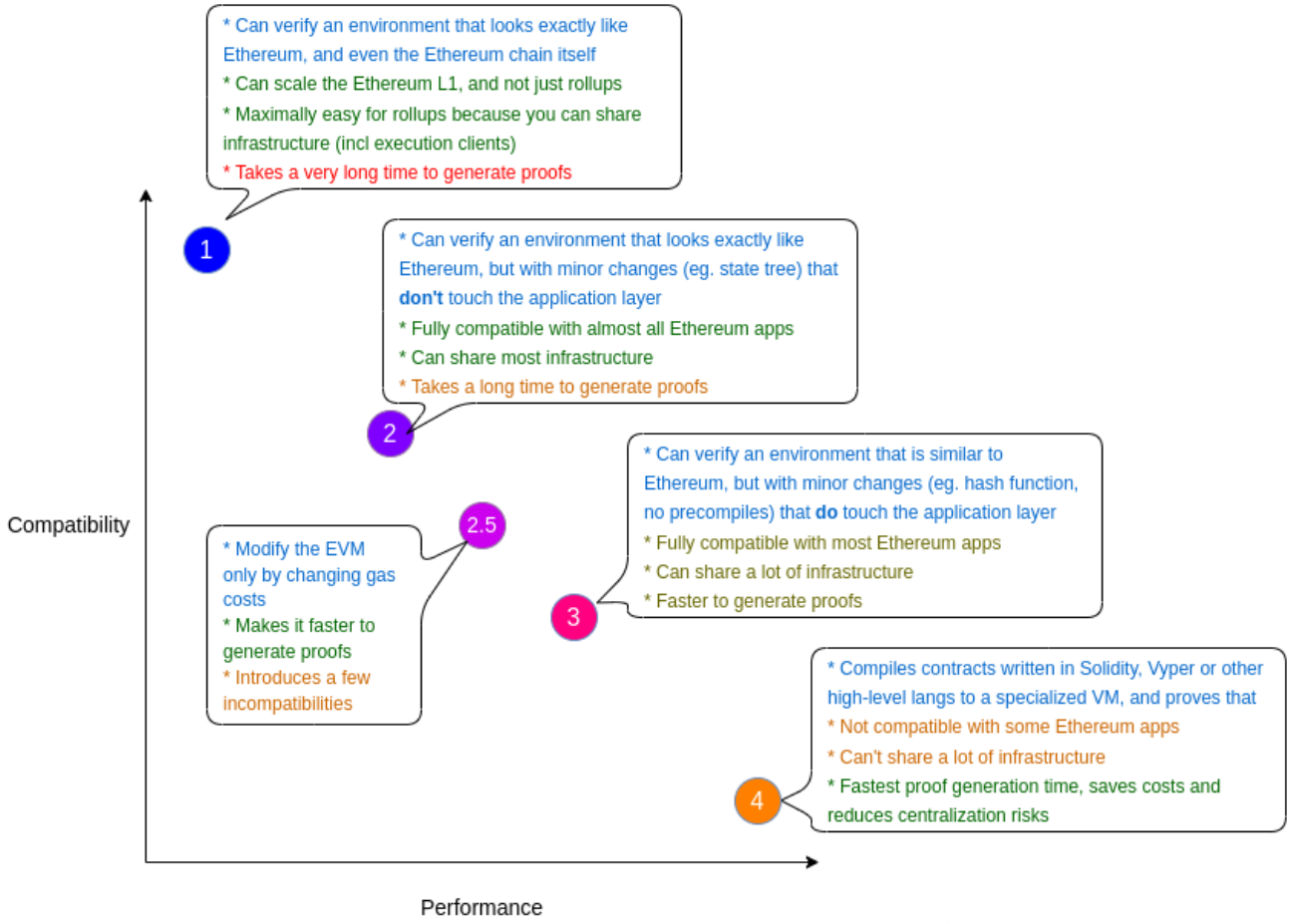
1. We are constrained by the cryptography (curves, hash functions) available on Ethereum.
2. The EVM is stack based rather than register based
3. The EVM has a 256 bit word (not a natural field element size)
4. EVM storage uses keccak and Merkle Patricia trees, which are not zkp friendly
5. We need to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

Summary of zkEVM Solutions

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

---

zkSync

zkSync is built on ZK Rollup architecture. ZK Rollup is an L2 scaling solution in which all funds are held by a smart contract on the mainchain, while computation and storage are performed off-chain.
For every Rollup block, a state transition zero-knowledge proof is generated and verified by the mainchain contract.
This SNARK includes the proof of the validity of every single transaction in the Rollup block. Additionally, the public data update for every block is published over the mainchain network in the cheap calldata.

Features of zkSync 2.0:

- zkSync is EVM and web3 compatible.
- We support Solidity and Vyper: no security re-audit required.
- Porting is effortless: 99% of tooling will work out of the box.
- With zkSync your project will inherit the full security of Ethereum.
- You will benefit from more transactions per second and lower gas fees.
- Build on zkSync 2.0 now and be permanently future-proof.

Scroll

This project is still at en early stage, there alpha testnet will be run on a private PoA fork of Ethereum (the testnet L1) operated by Scroll.
On top of this private chain, will run a testnet Scroll L2 supporting the following 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.

Polygon Products

See this guide
Strategy article
Recent paper on efficient zk proofs for Keccak

Polygon Zero

zkRollup solutions have a bottleneck in the time in takes to generate a proof.

Polygon Zero attempts to solve this with "recursive proofs", based on Plonky2.

Polygon Zero generates proofs simultaneously for every transaction in the batch. Theses are then aggregated into a single proof which is submitted on the Ethereum network.

This approach significantly reduces the effort it takes to generate reliable validity proofs. Polygon Zero's Plonky2 can generate a recursive proof in 0.17 seconds.

Hermez 2.0

Hermez current functionality is limited to token transfers and atomic swaps, so there are plans to introduce Hermez 2.0 which will have EVM compatibility.

Polygon Midden

Polygon Miden is a general-purpose, STARK-based ZK rollup with EVM compatibility. This will differ from Starknet in that it will be EVM compatible, so it should run Solidity contracts.
From their documentation
"Polygon Miden can process up to 5,000 transactions in a single block, with new blocks produced every five seconds. Although this ZK rollup exists as a prototype for now, it is expected to boost throughput to over 1,000 transactions per second (TPS) at launch."

Polygon Nightfall

From a collaboration with EY it is designed to allow private transactions.
It is a combination Optimistic rollups and zero knowledge, optimistic rollups for scalability and zk for privacy.
There is a beta version available on mainnet.
