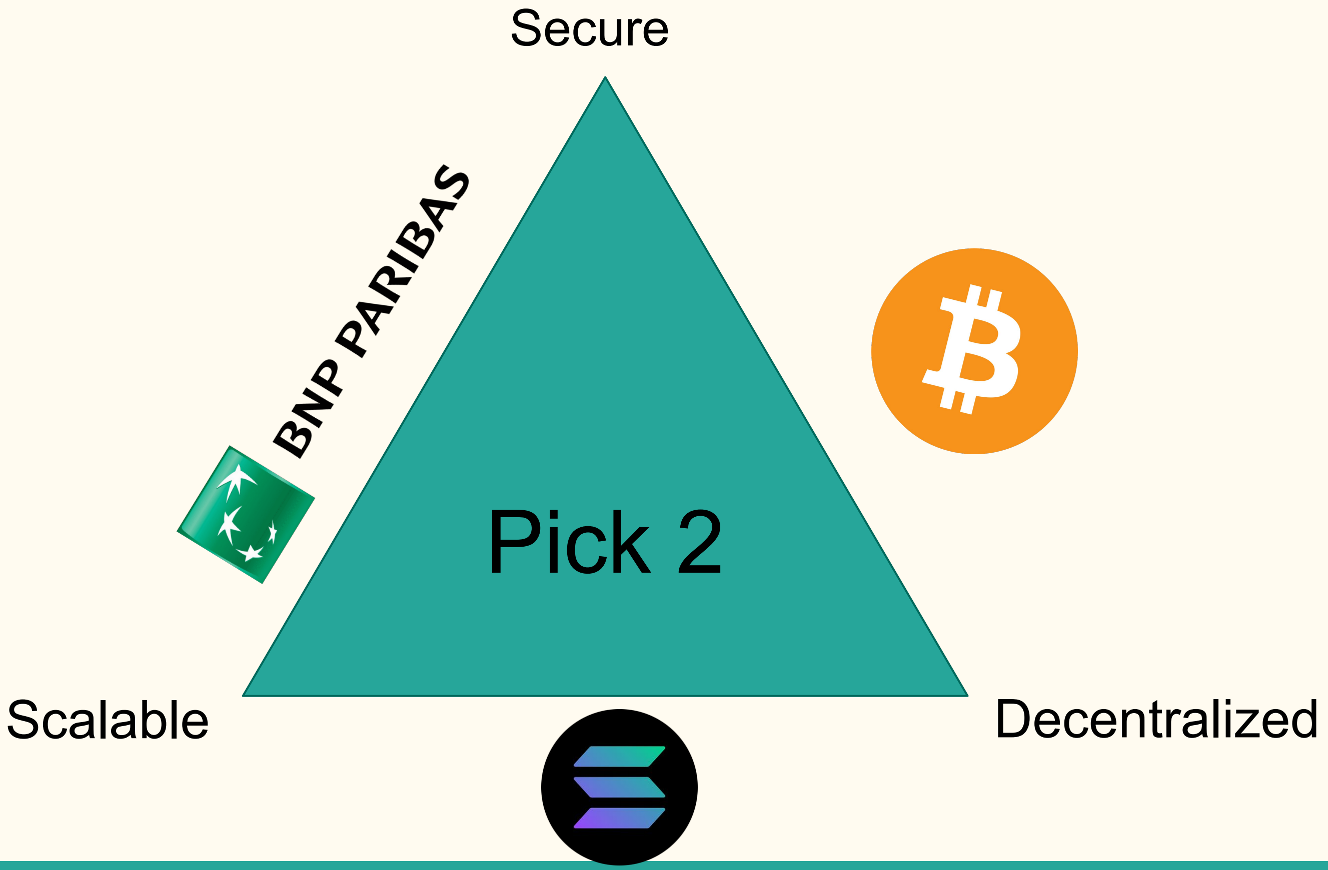
Lecture 12

Future of Ethereum: Scaling, ZKP

The Blockchain Trilemma



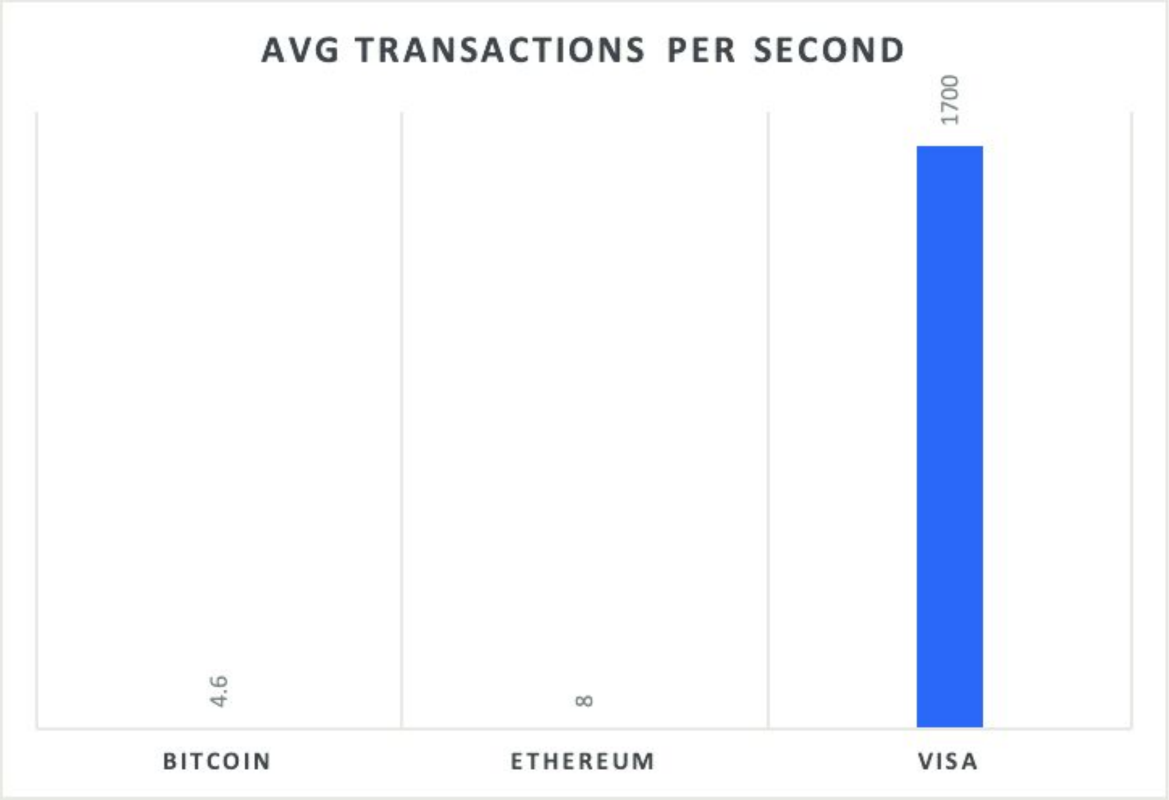
Transactions per Second

Financial Services generating transactions:

- ATMs
- POS devices
- Bank transfers
- Mobile banking
- Online payments

Speed is key to real world adoption.

24 hour SEPA mandate
How long would you wait for online payment confirmation?



Types of scaling

Layer 1 Scaling

Improvements that are made directly to the blockchain itself.

These improvements involve speed and utility increases to the chain itself. Eg. proof and consensus mechanisms. Block architecture.

Data processing and storage improvements of the chain.

Layer 2 Scaling

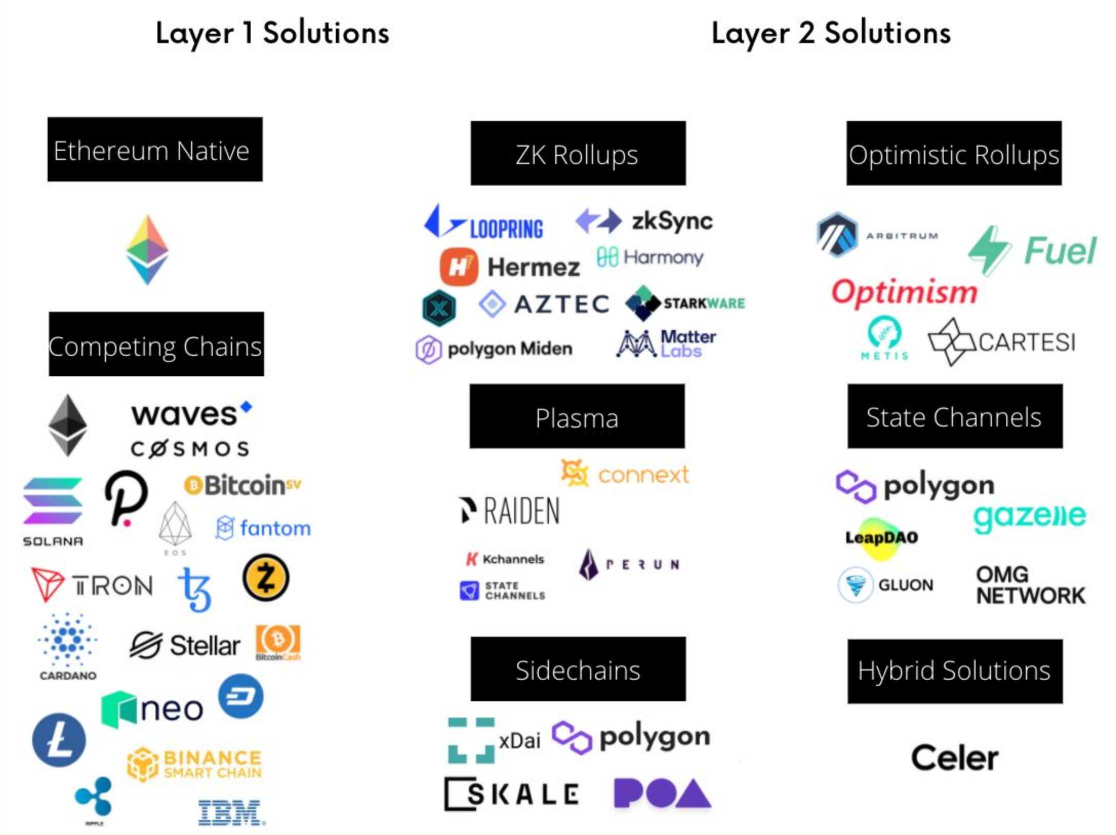
Improvements that are made on top of the L1 chain

L2 creation and operations are defined on L1 (Ethereum) but move the computation and storage demands off of L1.

The validity of L2 information is of concern when it is posted back onto L1 chains.

Layer 3 Hyperscaling

Blockchain Scaling Landscape



Layer 1 Scaling

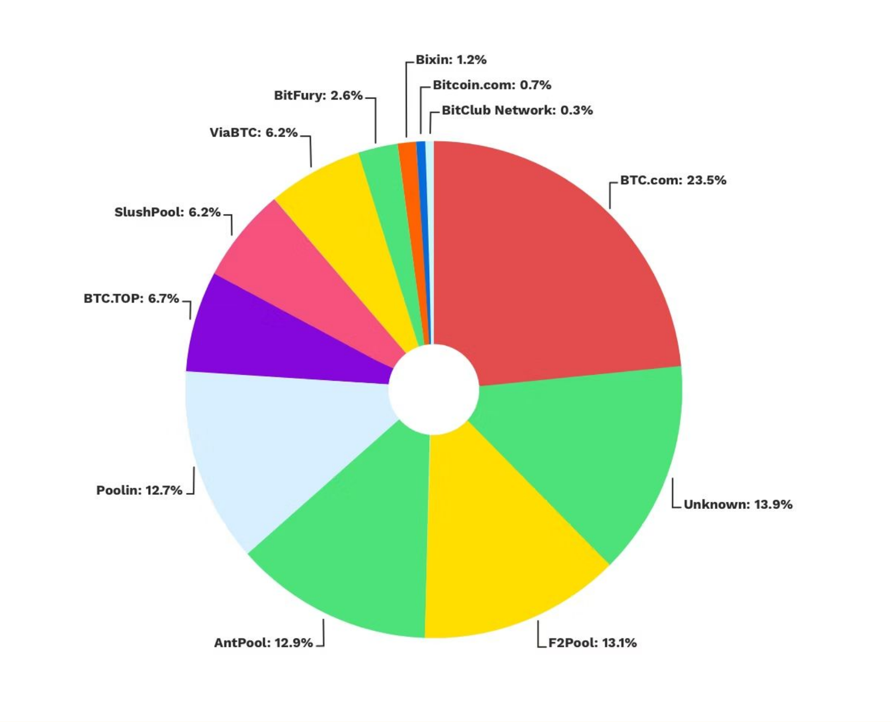
Consensus & Proofs

Consensus mechanism improvements - Bitcoin

Proof of Work

Earn rewards by solving hashing puzzles of new blocks. The hash is usually required to produce a certain number of zeros at the beginning of the block. Once a new block is created and added to the chain, it is mined.

- Bitcoin rewards halving and puzzle difficulty increase reduces profitability.
- Currently puzzle difficulty means only specialised hardware (ASIC) can solve these puzzles.
- Individual mining is now impossible. Only mining pools profiting. Not decentralized?
- Nakamoto consensus.- longest chain wins. 51% attacks? Discard shorter chains.



Consensus mechanism improvements - Ethereum

Proof of Stake

Validators stake ETH vs Miners stake computation power. Validators can **propose** new blocks or **attest** blocks being propagated.

- Ethereum Classic (ETC) used PoW. Switched to PoS in 2022 (ETH).
- Staking methods - join activation pool to rate limit new validators (extremely rich players?)
  - Solo staking requires 32ETH (65k USD) + hardware (computation + redundancy)
  - Staking as a Service (minus hardware costs?)
  - Staking pools
- Every 12s (1 **slot**), a validator is chosen to be the proposer and a group is selected to be attestors (re-execute block for correctness). Every 32 slots is considered 1 **epoch**. All validators check proposed 32 blocks to prevent subgroup dishonesty.
- Consensus now 67% majority, uncle blocks rewarded as opposed to orphan blocks in Bitcoin.
- Beware of slashing, check rules.
  - Attesting two competing blocks
  - Proposer spams blocks or proposes malformed blocks (Gasper)
  - ETH penalty and ban period. Or permanent expulsion.
  - Whistleblower rewards

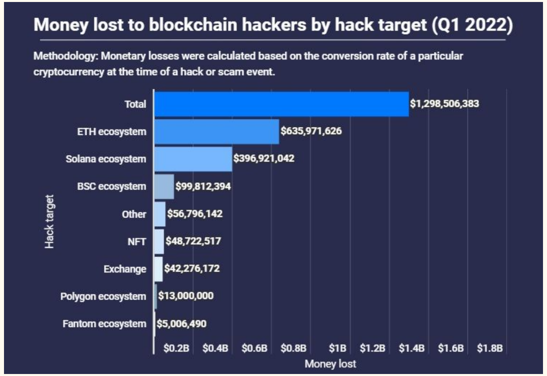
Consensus mechanism improvements - Solana

POS + Proof of History

A cryptographic clock that is implemented on top of PoW or PoS. A different take on Byzantine fault tolerance (51% attack, Sybil attack). Ethereum requires ⅔ attester confidence. PoH server generates a Verified Delay Function (VDF) and stamps each transaction. Validators check timestamp and send vote to PoH server. **No need for validators to come to consensus about a block.**

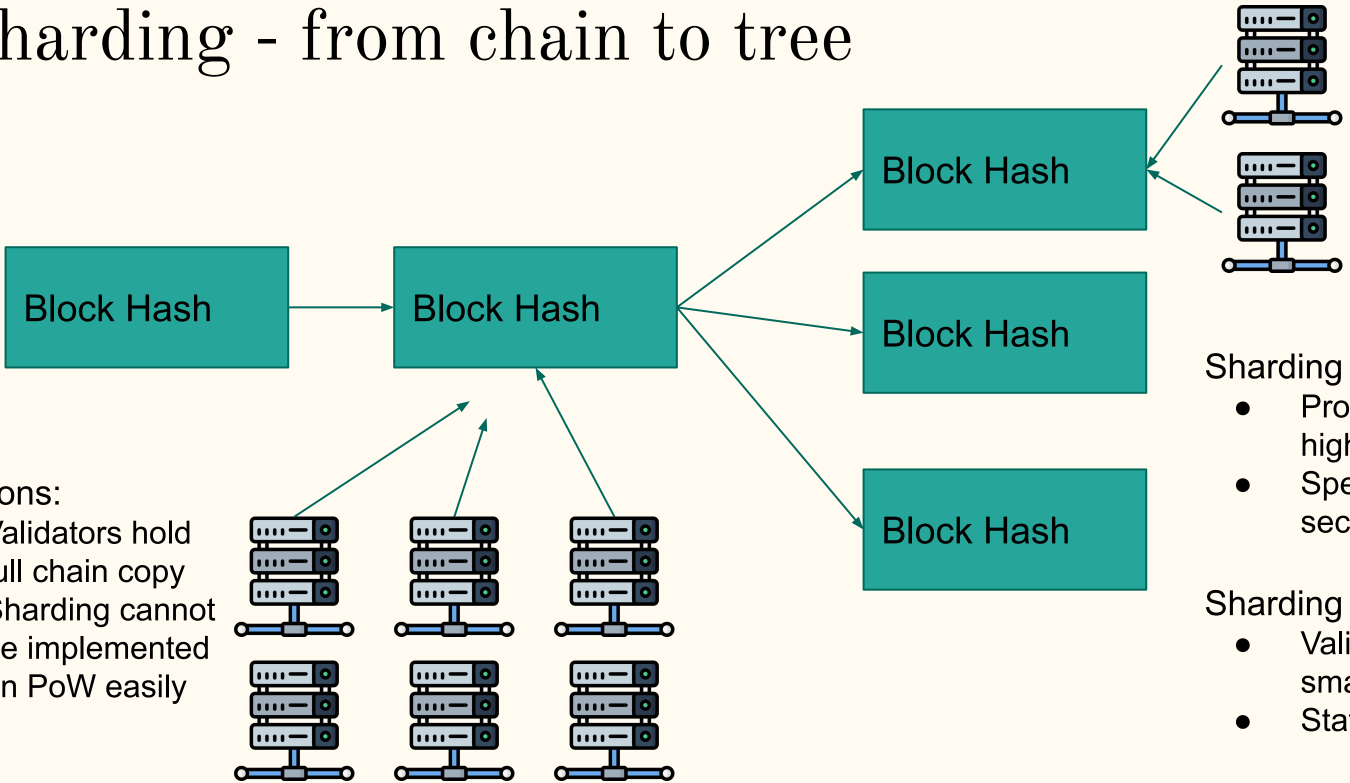
- PoH server a huge source of centrality.
- Added complexity increases network outages
- Decreased security checks means a lot of hacks!

Ethereum: 67
BNB Chain: 33
Fantom: 4
Solana: 5
Avalanche: 6
Arbitrum: 3
Harmony: 2



Live Transaction Stats	
Transaction count	275,180,063,153
Transactions per second (TPS)	2,287

Sharding - from chain to tree



Sharding advantages:

- Process transactions in parallel, higher TPS!
- Speed improvement without security loss unlike PoH.

Sharding challenges:

- Validators are now diluted into smaller shard pools (still BFT?)
- State collisions must be resolved

Limitations:

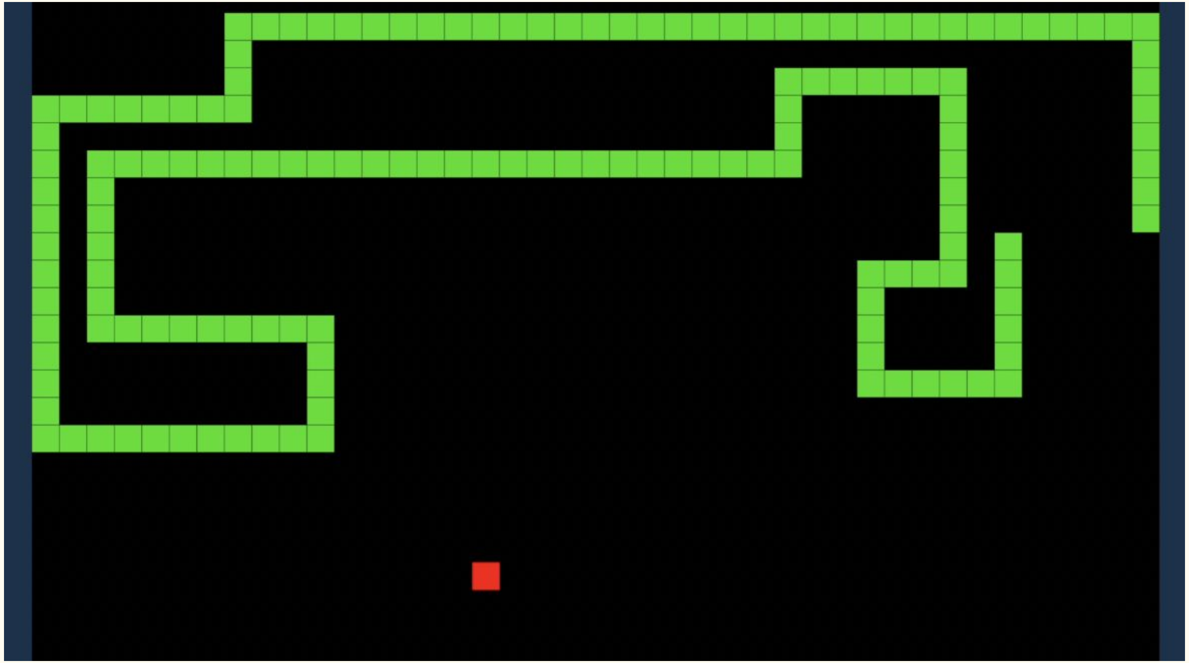
- Validators hold full chain copy
- Sharding cannot be implemented on PoW easily

Sharding, ultimately, was never implemented on Ethereum even though it was discussed for a long time. With EIP-4844, Ethereum implemented **Danksharding** to work with Layer 2 solutions.

Layer 2 Scaling

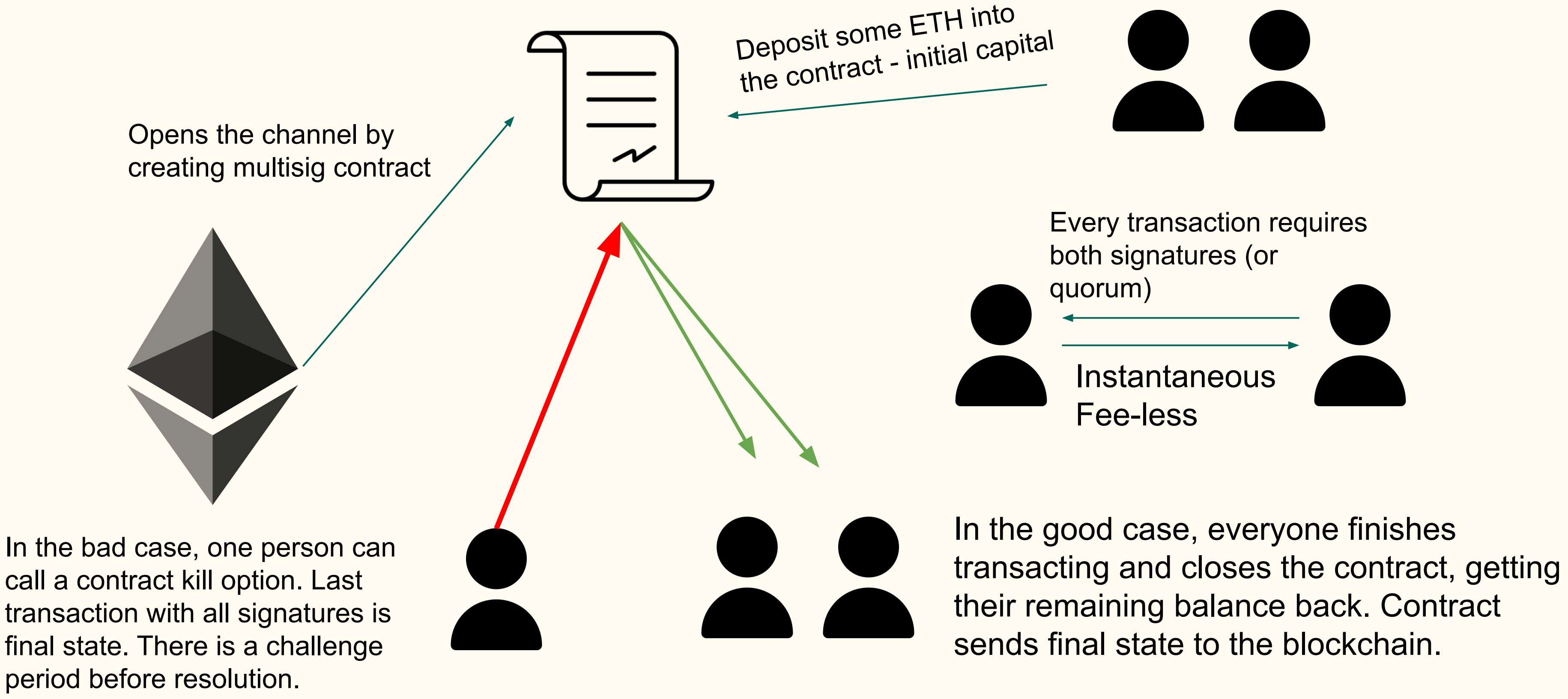
State Channels, Side Chains, Rollups

State Channels - Moving transactions off chain

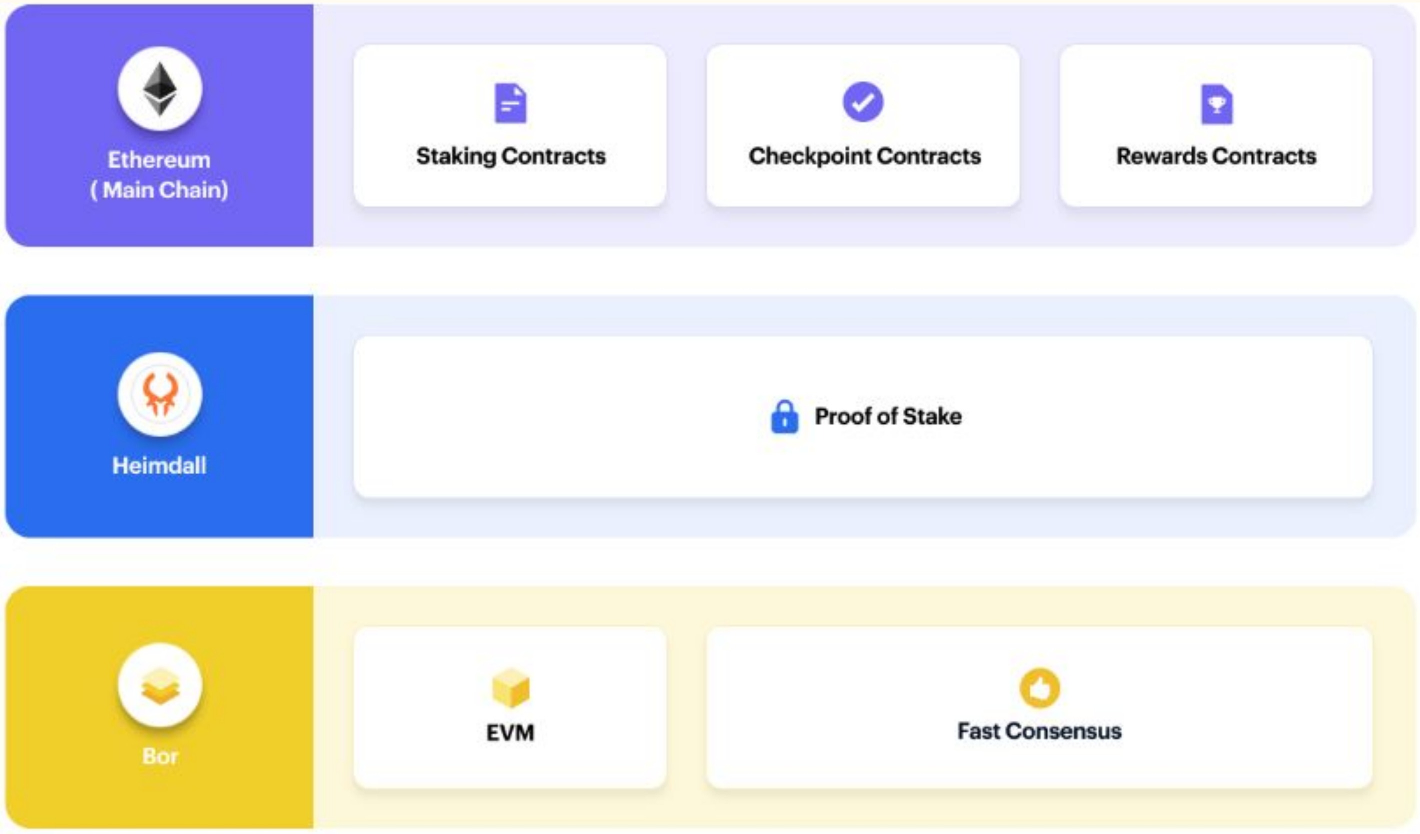


Only 2 transactions: channel open state and channel close state

State Channels - multisig smart contracts



Side Chain Example - Polygon Matic



Heimdall Checkpoints

```
type CheckpointBlockHeader struct {
    // Proposer is selected based on stake
    Proposer        types.HeimdallAddress `json:"proposer"`

    // StartBlock: The block number on Bor from which this checkpoint
    StartBlock      uint64                `json:"startBlock"`

    // EndBlock: The block number on Bor from which this checkpoint en
    EndBlock        uint64                `json:"endBlock"`

    // RootHash is the Merkle root of all the leaves containing the bl
    // headers starting from start to the end block
    RootHash        types.HeimdallHash    `json:"rootHash"`

    // Account root hash for each validator
  // Hash of data that needs to be passed from Heimdall to Ethereum ch
    AccountRootHash types.HeimdallHash    `json:"accountRootHash"`

  // Timestamp when checkpoint was created on Heimdall
    TimeStamp       uint64          `json:"timestamp"`
}
```

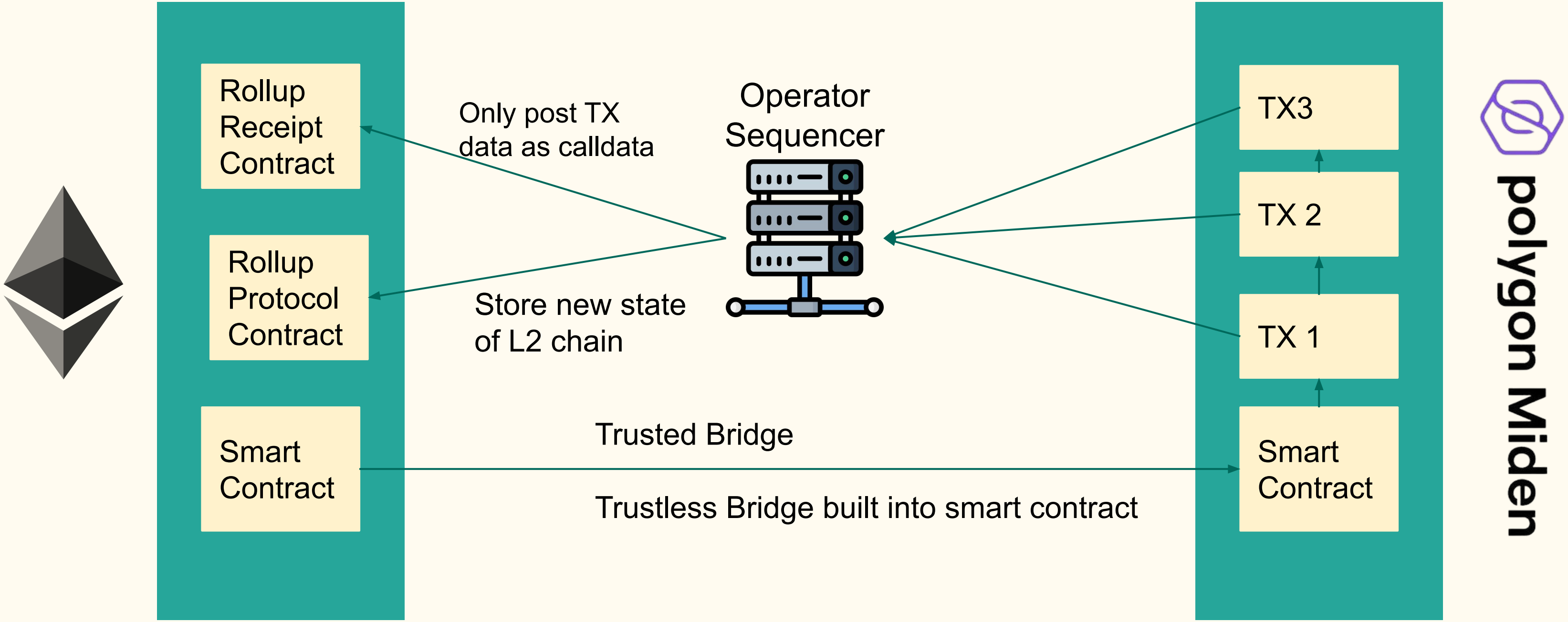
Heimdall Nodes must produce checkpoints which conform to the struct shown in code.

The RootHash hashes all blocks from start to end.

Validator group must vote on the validity of this checkpoint before it is submitted to the Checkpoint contract on Ethereum.

Rollups - best of both world

State Channels are only useful for limited use cases.
Side Chains sacrifice security for speed



Rollups - Optimistic vs Zero Knowledge

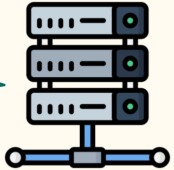
Optimistic - Fraud Proof

Rollup Receipt Contract

Rollup Protocol Contract

Challenge Assertion

Operator Sequencer



Rollups are posted as assertion blocks. There is a challenge window per assertion. If challenged, all transactions in rollup gets executed on the Ethereum chain. This can get expensive! If fraudulent, the operator gets slashed.

ZK- Validity Proof

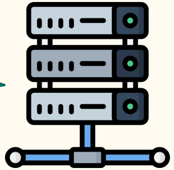
Verifier

Rollup Protocol Contract

TXs
ZK SNARK
ZK STARK

Batch root

Operator Sequencer

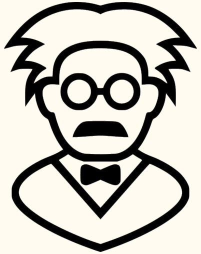


In this setup, no challenges are necessary!

Zero Knowledge Proofs

ZK proof - Challenge Example

Prove without showing



We know everything about Blockchain and want an A in the class

Prove to me you have knowledge of Blockchain

We don't want to reveal what we know about Blockchain

0xFd348ab656a6127f4280C5b1218D46D
80a41e224
I expect that you will hack my wallet and your wallet increases in money

Here is the block explorer where all money from your wallet goes into mine

Okay you proved to me you know about Blockchain. Automatic 20 for all homeworks.

ZK proof - Use Cases

Real World

- Non Proliferation of Nuclear Weapons Treaty
  - US and Russia both has large nuclear stockpiles
  - How to prove they are dismantling their weapons without giving away military secrets?
- Vote Transparency
  - Everyone wants to know their vote was accurately accounted for
  - How to prove each vote without revealing identity and political affiliation?
- Supply Chain Management
  - Confirm origin, fair trade, ethical production practices.
  - Prove compliance to environmental and labour laws.
  - Do not reveal any business secrets.

Web3 Specific

- ZK identity
  - DeFi KYC and AML compliance through sending proofs.
  - ZK Mail - Request 3rd party service to send email. 3rd party sends email and provides proof. Try it yourself!
- Privacy preserving Transactions
  - Execute transaction without publishing transaction data, only proof
  - Obscure transaction history, confidential payments
- Private Rollups - Aztec
  - No transaction storage on the L1.
  - Decouple private and public state

ZK Proof - Mathematical Example

Verifier

Prove x

Prover

g

$y = g^x$

$t = g^v$

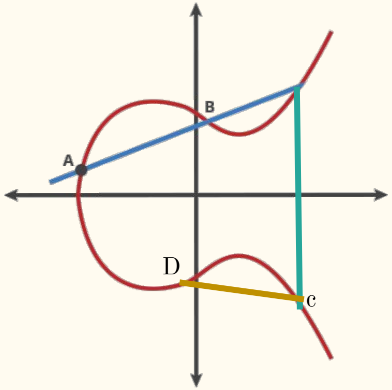
(t, r, c)

$t = g^r * y^c$

$$v = r + cx$$
$$t = g^{r+cx}$$
$$t = g^r * g^{cx}$$
$$t = g^r * (g^x)^c$$
$$t = g^r * y^c$$

ZK Proof System Construction Comparison

- SNARK - Succinct Non-interactive Argument of Knowledge
  - Succinct because verification is much faster than proof generation
  - Requires a trusted setup environment - longer proving times
  - Generated from an elliptic curve - fast and secure
- STARK - Scalable Transparent Argument of Knowledge
  - Extremely complex hashes making proofs very big
  - Can be verified publically by anyone - scalable and transparent!
- Bulletproof
  - Tries to get the best of both worlds - small proof size and no setup
  - Unfortunately extremely long proving and verification times



	Trusted setup	Verification Time	Proof Size	Prover Time
SNARK	Yes	10ms	~200b	2.3s
STARK	No	<16ms	~45,000b	~1.6s
Bulletproof	No	~1,100ms	~1,300b	~30s

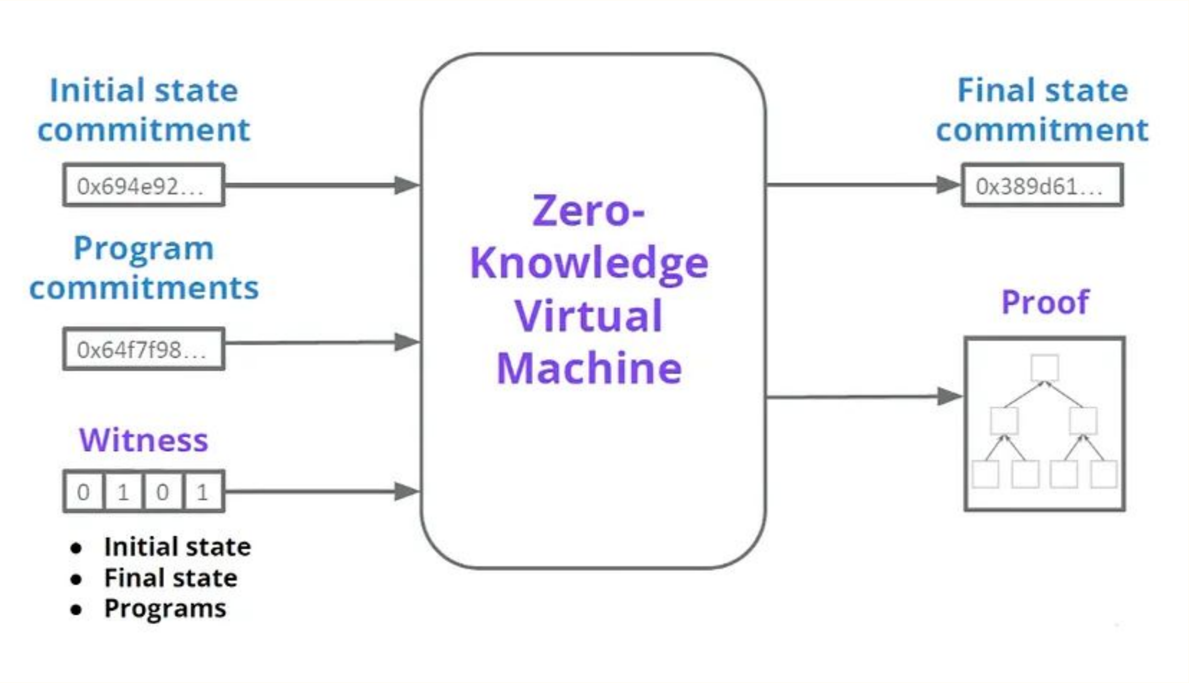
- Setups are a potential attack vector. Require the correct hardware and configuration.
- Larger proofs are more network intensive.
- Time needed to generate proof and verify affect throughput.

zkEVM - Proof generating VMs

Currently zk rollups can only generate proofs for transactions of certain structures. The EVM is responsible for executing requests to create the next transaction. Can it generate a proof for any transaction? Yes but very difficult!

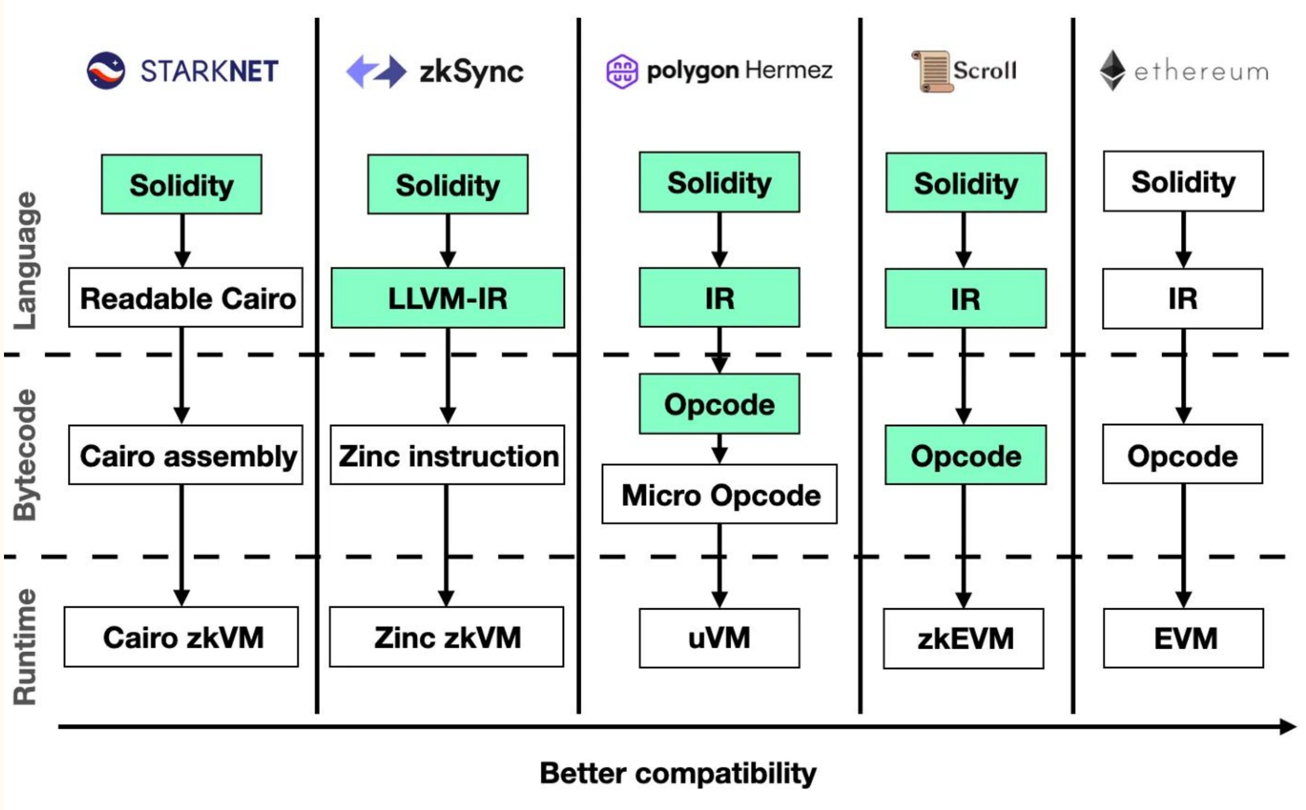
- EVM uses a stack to manage calculations during code execution. Easier if data can be grabbed from registries (key - value pairs).
- How to encompass all opcodes into a proof? Especially contract calls.
- EVM stores global variables in Merkle Trees hashed with a keccak hash. This is really computationally expensive to prove than other data structures.

zkEVM architecture



1. Stores the initial state
2. Tracks the execution
3. Simultaneously produce a proof
4. Commit the transaction to chain and the proof to the verifier contract.

zkEVM - Design Considerations



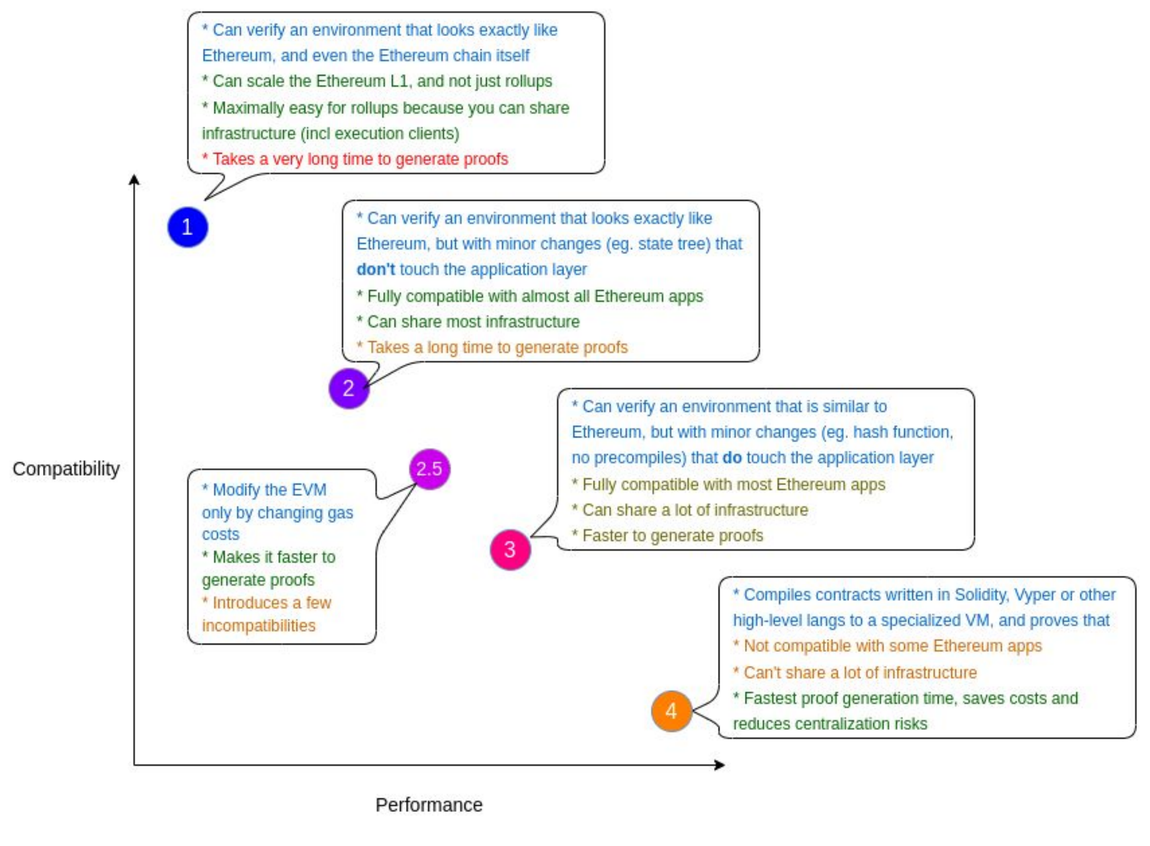

```
[profile.default]
solc-version = "0.8.17"
via_ir = true

[profile.lite.optimizer_deta
optimizerSteps = ''
```

Forge is able to generate IR Yul syntax and go through optimization rounds automatically

forge build

zkEVM - Design Considerations



More compatible

- Less new architecture
- Compatible with current Ethereum dApps
- Extremely slow

Less compatible

- High capital investment into infrastructure
- Must create new dApps
- Very fast