BLOCKCHAIN AT BERKELEY

# Cryptocurrencies for the Masses

## Scaling Blockchain

SEHYUN CHUNG
ALPIN YUKSELOGLU

# Table of Contents

01 Background

02 Taproot

03 Sharding

04 Segregated Witness

05 Lightning Network

06 Plasma, Optimistic Rollups

Introductions

# Meet Your Lecturers

**Sehyun Chung**

Education/Xcelerator

**Alpin Yukseloglu**

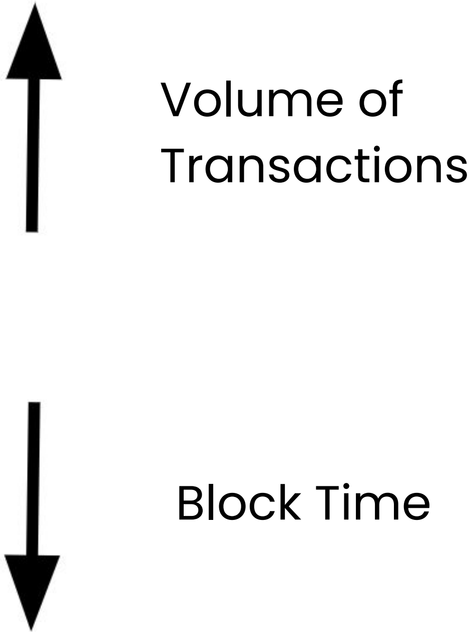
Consulting/Xcelerator

# Background

# What is the Scalability Problem?

**Goal:**
Provide all of the services that a blockchain offers to all users, *independent* of how many users there are

# What is the Scalability Problem?

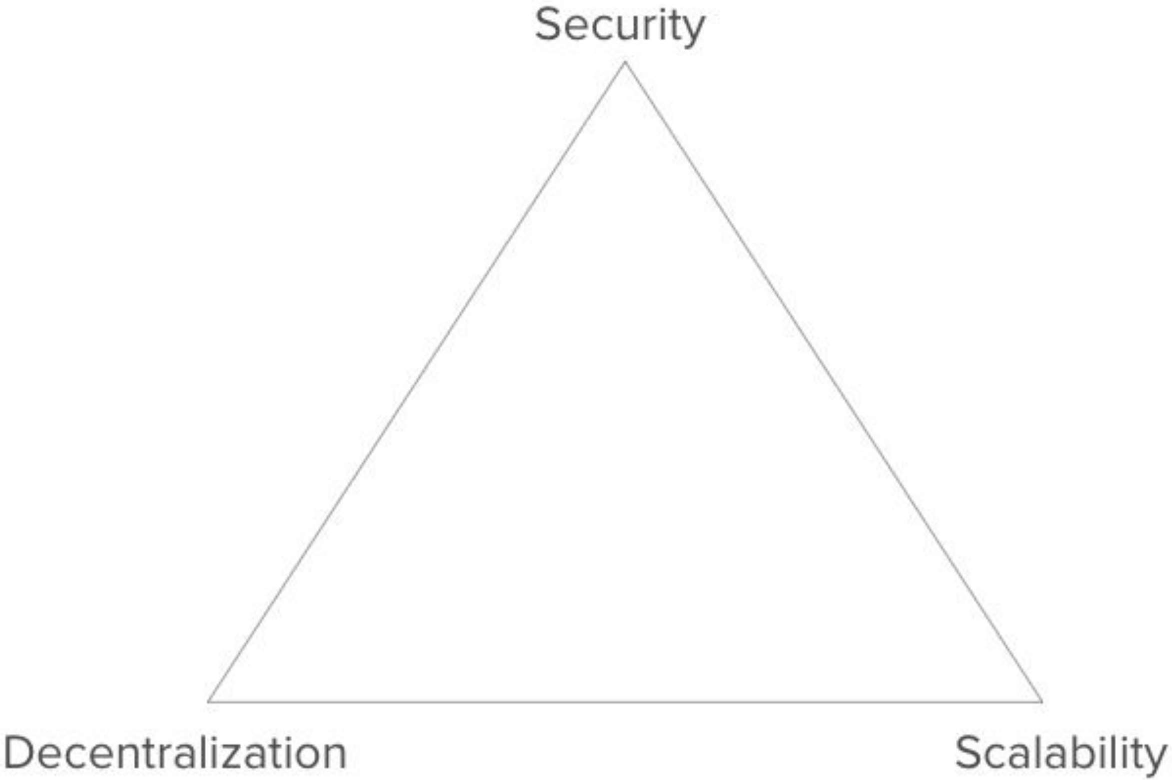


Volume of Transactions

Block Time

Background

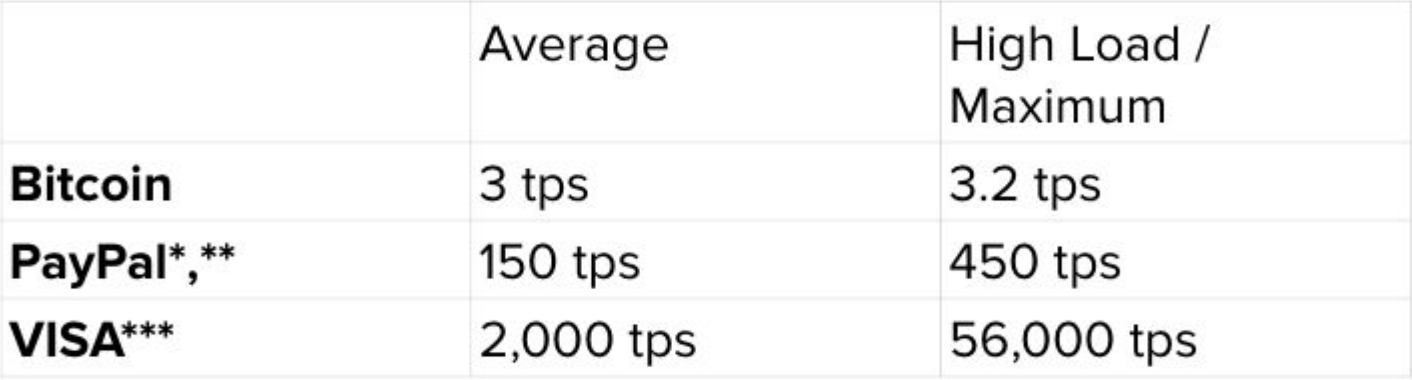
# Scalability Triangle



Background

# TPS Comparisons

How does Bitcoin compare with other traditional payment systems*?



| | Average | High Load / Maximum |
|---|---|---|
| **Bitcoin** | 3 tps | 3.2 tps |
| **PayPal*,*** | 150 tps | 450 tps |
| **VISA***** | 2,000 tps | 56,000 tps |

*debatably an unfair comparison/metric, but people care about it anyways

# Layer 1 vs. Layer 2

Layer 1

- refers to the main blockchain architecture
- e.g. the Bitcoin Blockchain, the Ethereum Blockchain

L1 Solutions

- changes made **on-chain** that improve scalability

Layer 2

- refers to a secondary framework or protocol that is built on top of an existing blockchain
- e.g. the Lightning Network, Ethereum Plasma

L2 Solutions

- changes made **off-chain** that improve scalability

# QUESTIONS?

# Layer 1

What Layer 1 scaling solutions have you heard of?

# Taproot

Problems before Taproot

- Signature verification is slow, but required **everywhere** in the network
  - case 1: when validating a new block
  - case 2: when a node newly connects (or reconnects) to the network
- Users can see many details of a transaction (all scripts, multisignature, etc.)

# Taproot + Schnorr

- Schnorr is a type of signature, Taproot is the Bitcoin upgrade (BIP) that uses Schnorr
- Taproot combines Schnorr, Merkelized Abstract Syntax Trees (MAST)
- Idea: If signature validation is faster, much of the network can run faster (more throughput)
- Taproot replaces ECDSA signatures with Schnorr Signatures
- Schnorr benefits:
    - key aggregation
    - verification in linear time

Source Bitcoin Optech Schnorr Taproot Workshop

# Benefits of Taproot + Schnorr

- allows multiple signers in a transaction to create an aggregate public key and an aggregate signature so multisig ends up **looking** the same and **costing** the same as a single signature
    - 30-75% savings on multisig
- allows for **batch validation**
    - a process in which many signatures can be verified *at once*, faster than than verifying each signature individually
    - this speedup grows logarithmically with the number of sigs to verify, so verifying 1 billion signatures (IBD) could be 4x as fast!
- when a transaction has many scripts, they do not need to be revealed or evaluated on the network

Source Bitcoin Optech Schnorr Taproot Workshop

# Benefits of Taproot

- More Privacy:
  - many transaction types (multisig, scripts, etc.) look the same as regular transaction
    - impossible for user monitoring the network to figure out if a transaction was signed by one person or multiple people
  - all scripts of a transaction are not broadcast for everyone to see
- Functionality
  - enables very large k of n multisig
  - can use much larger scripts

Source Bitcoin Optech Schnorr Taproot Workshop

# QUESTIONS?

Layer 1

# Segregated Witness (L1 + L2)

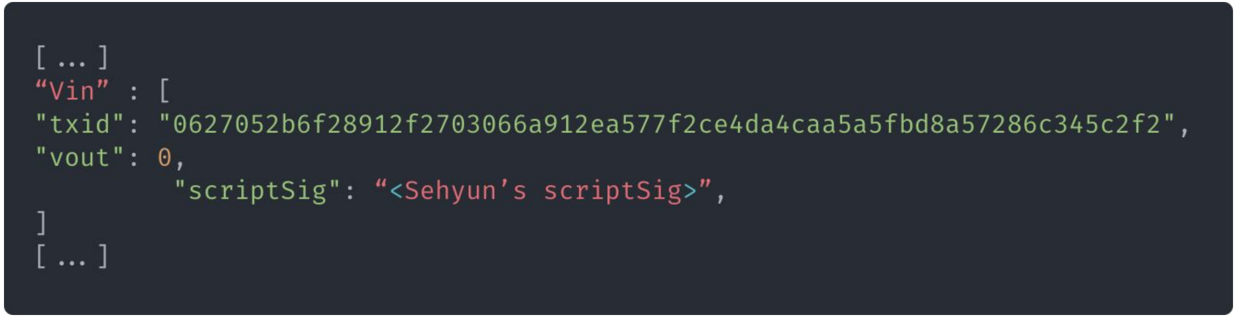
Problems before Segwit:

- transaction malleability
- inefficient script versioning system
- inefficient signature hashing operations
- historic signature data everywhere

Segwit

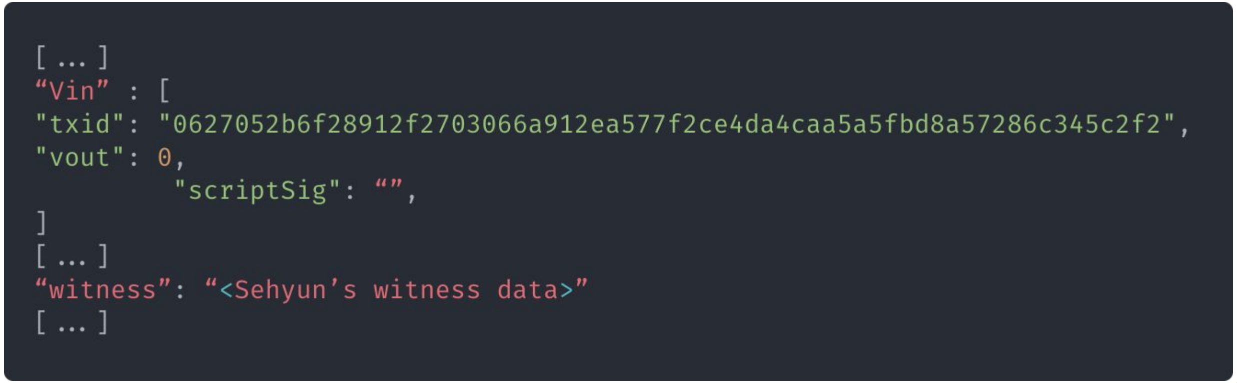
# Inside Transactions

**Before**



```
[ … ]
“Vin” : [
"txid": "0627052b6f28912f2703066a912ea577f2ce4da4caa5a5fbd8a57286c345c2f2",
"vout": 0,
            "scriptSig": “<Sehyun’s scriptSig>”,
]
[ … ]
```

**After**



```
[ … ]
“Vin” : [
"txid": "0627052b6f28912f2703066a912ea577f2ce4da4caa5a5fbd8a57286c345c2f2",
"vout": 0,
            "scriptSig": “”,
]
[ … ]
“witness”: “<Sehyun’s witness data>”
[ … ]
```

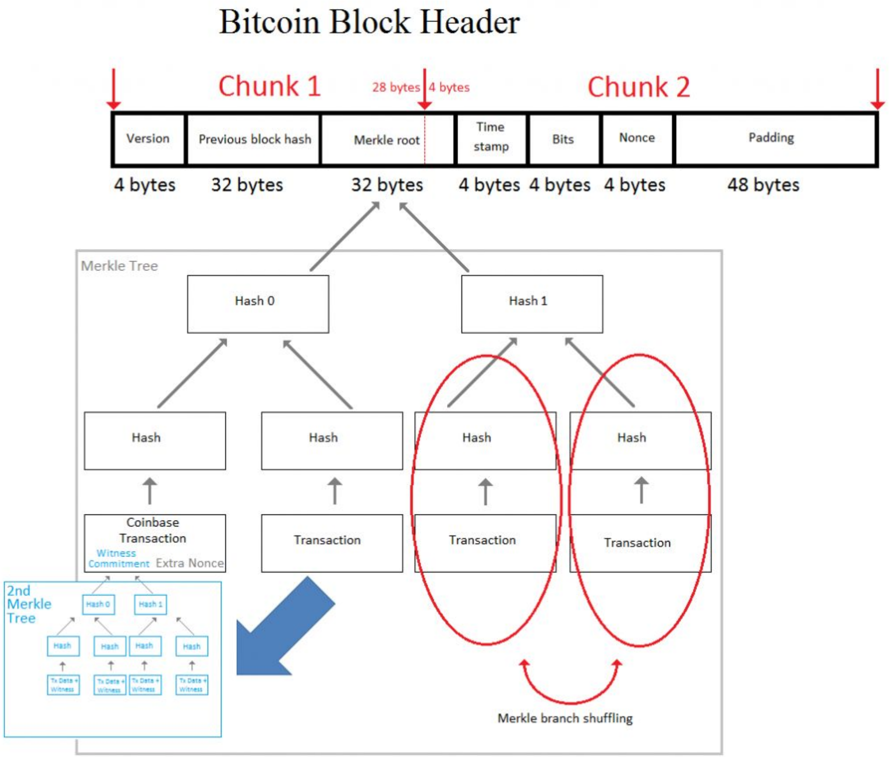
Source: *Mastering Bitcoin* by Andreas Antonopoulos

Segwit

# Mirrored Signature Tree

**Q**: How can we verify that the correct signatures were included with each transaction in the first place?

**A**: Construct a merkle tree of signatures that mirrors the merkle tree of transactions



# Script Versioning and Signature Hashing

Scripts

- a series of opcodes
- To implement changes, one had to replace an extra opcode with the new one
- Segwit introduced version numbers so that upgrading to a new script version could be done by simply increasing this version number
- made upgrades to Schnorr and MAST much easier!

Signature Hashing

- resolved the issue of quadratic time signature hashing by only requiring each byte to be hashed at most twice

Segwit

# Pros/Cons

**Pros:**

- Fixes Transaction Malleability
    - Allows Lightning Network and sidechains to work
- Soft Fork
- Increased efficiency
- Increased Block Size
- Smaller Size of Blockchain

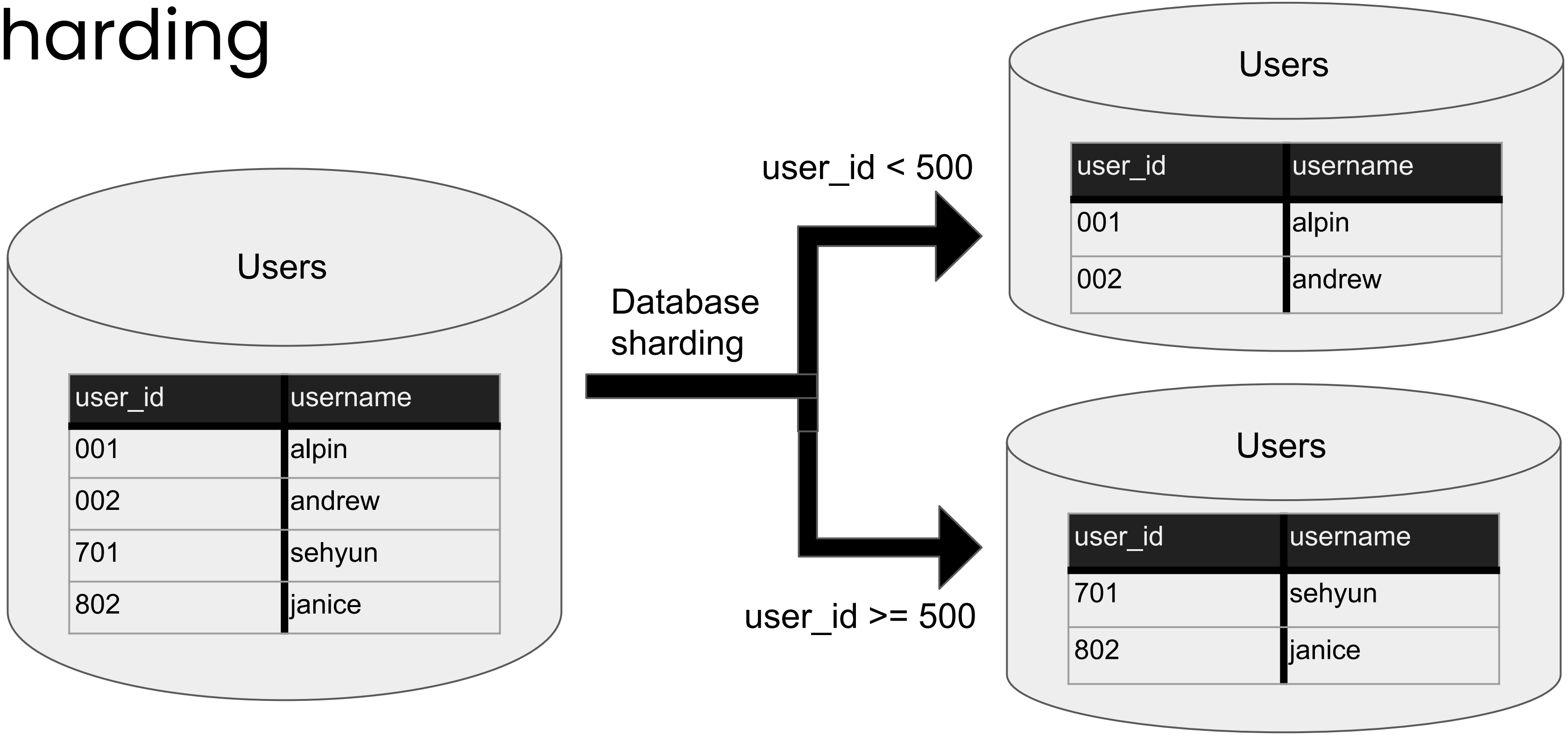
**Cons:**

- One-time linear capacity increase
- Obligated wallets to upgrade
- Better ways to solve malleability exist, but had to keep into account the needs of developers, miners, and users

# QUESTIONS?

Layer 1

# Sharding



Users

| user_id | username |
|---|---|
| 001 | alpin |
| 002 | andrew |
| 701 | sehyun |
| 802 | janice |

Database sharding

user_id < 500

Users

| user_id | username |
|---|---|
| 001 | alpin |
| 002 | andrew |

user_id >= 500

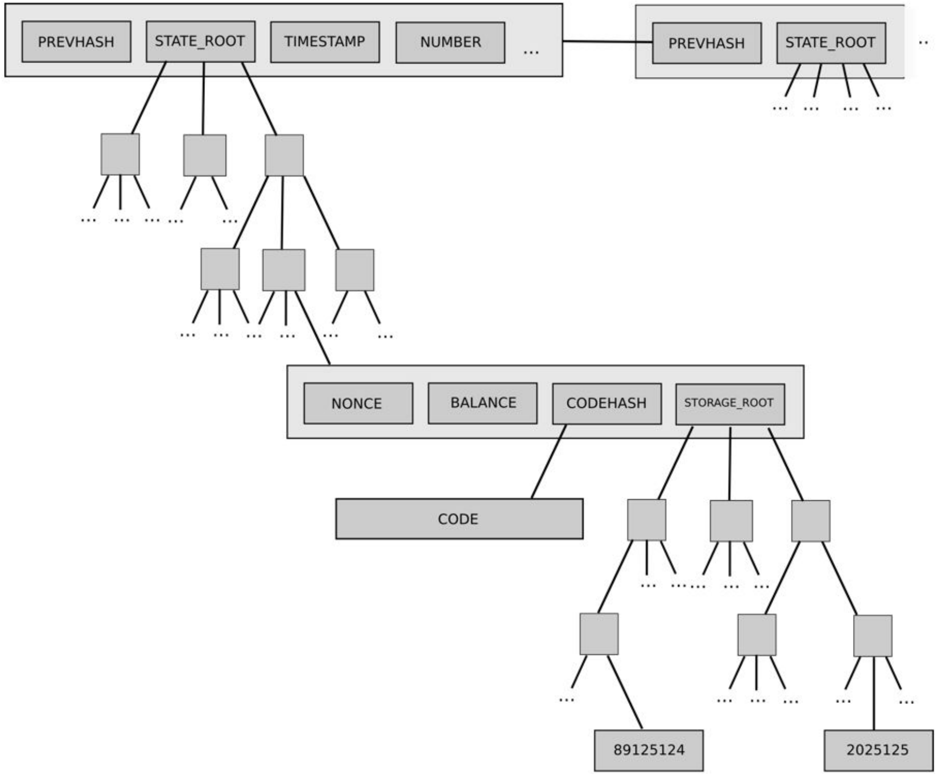
Users

| user_id | username |
|---|---|
| 701 | sehyun |
| 802 | janice |

Layer 1

# Sharding

Sharding is the idea of not requiring every miner to be working on every single block, essentially creating parallel but connected blockchains.



Source: https://github.com/ethereum/wiki/wiki/Sharding-FAQs

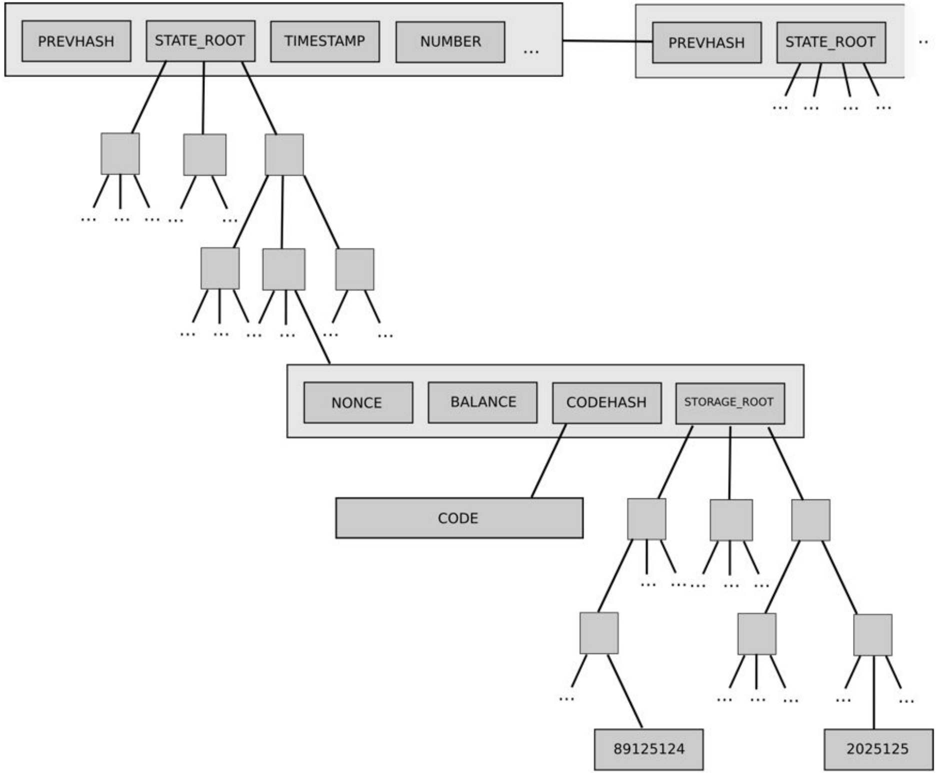
# Sharding

Node categories:

- super-full node
- top-level node
- single-shard node
- light node

Some challenges:

- Cross-shard communication
- Single-shard takeover



Source: https://github.com/ethereum/wiki/wiki/Sharding-FAQs

# QUESTIONS?

# Layer 2

What Layer 2 scaling solutions have you heard of?

Layer 2

# Lightning Network

Recall:

- Every transaction is put onto the blockchain
- 6 block confirmation time
  - ~1 hour wait time
- Each transaction has a transaction fee
  - Inconsistent
  - Not economical for low-value items
  - Micro-payments impossible

Layer 2

# Lightning Network

Idea:
Don’t put every transaction on the blockchain

Layer 2

# Lightning Network

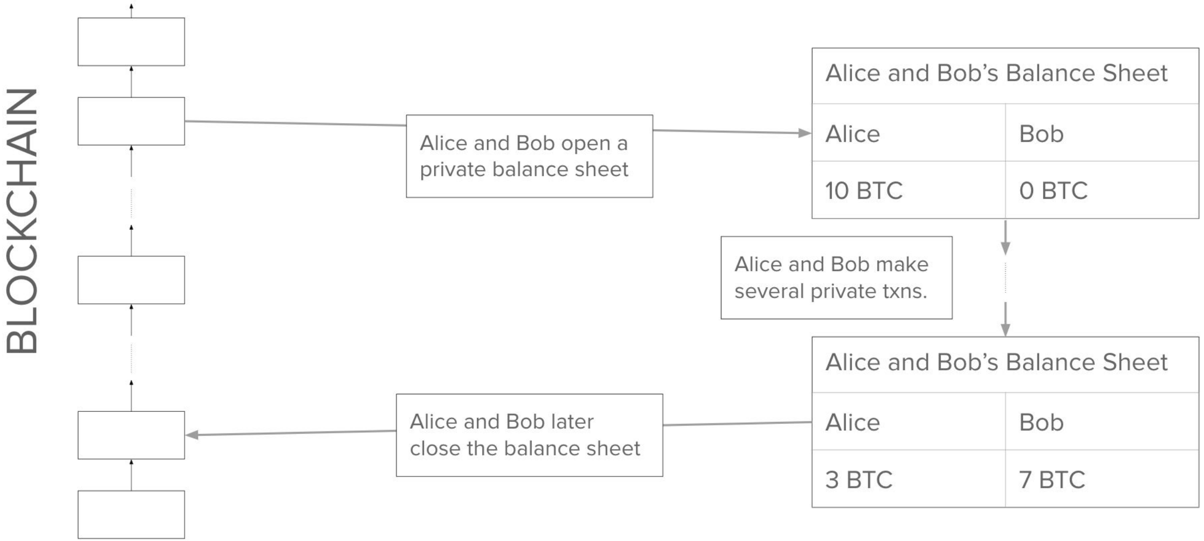
Idea:
Don’t put every transaction on the blockchain

Problem:
Can Alice and Bob make payments between themselves without always needing to consult the blockchain?

# Payment Channels

Idea:

- What if Alice and Bob maintain a private balance sheet
  - update the private balance sheet with every payment
  - only consult the blockchain when they want to settle the balance



Alice and Bob open a private balance sheet

| Alice and Bob’s Balance Sheet | |
|---|---|
| Alice | Bob |
| 10 BTC | 0 BTC |

Alice and Bob make several private txns.

| Alice and Bob’s Balance Sheet | |
|---|---|
| Alice | Bob |
| 3 BTC | 7 BTC |

Alice and Bob later close the balance sheet

# Payment Channels

**Q**: What if either Alice or Bob try to cheat the other by settling early, or never?

**A**: Incorporate fallbacks using bitcoin Script to create blockchain-enforceable contracts between Alice and Bob so that neither party can cheat the other, while maintaining the private balance sheet functionality!

# Payment Channels

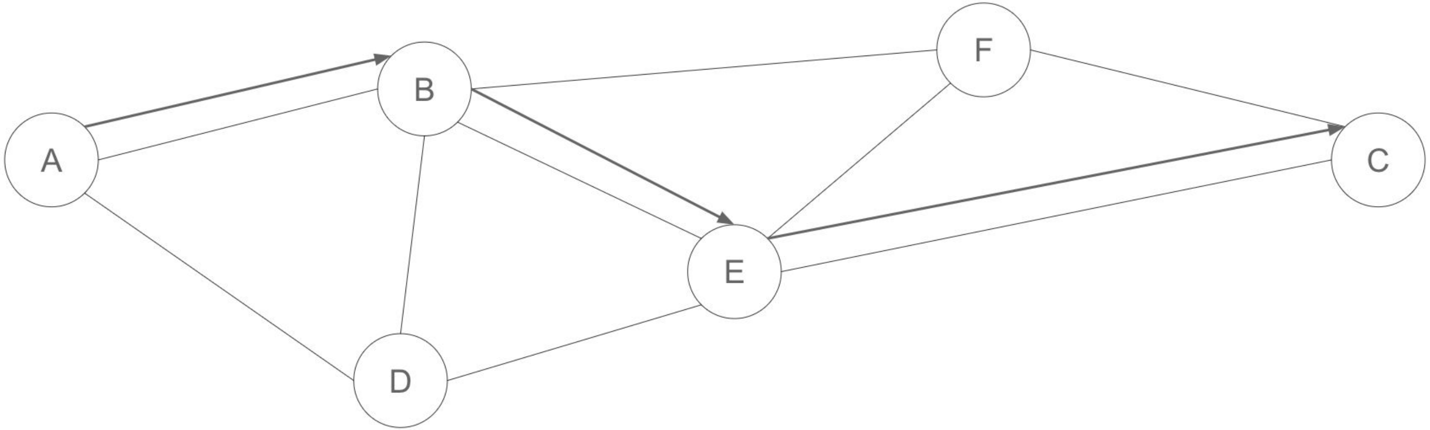
Observation:

- If either Alice or Bob cheat
    - can always override and take all the money in the deposit.
- If Alice and Bob always cooperate
    - don’t have to touch the blockchain, except when creating the payment channel and settling the balance.
- Only two transactions on the blockchain (initial and settlement)
    - Supports arbitrary number of local transactions between Alice and Bob

Layer 2

# Lightning Network

Alice sends money to Charlie through this hypothetical payment channel network



Lightning Network

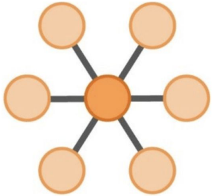
# Pros/Cons

**Pros:**

- Assuming enough capital, *people can make payments instantly*
- No need to wait for confirmation times
- Only use Bitcoin blockchain to settle disputes and close out payment channels (super efficient)
- Only pay transaction fees on channel open/close
- 3 TPS => 10,000's+ TPS

**Cons:**

- Nodes need to keep **large amounts** of capital locked up in payment channels
- **Risk of centralization** since only nodes with significant capital can afford to run payment channels for long
- Tendency towards *hub-and-spoke* topology



# QUESTIONS?

# Raiden Network Scalability

- Ethereum’s Payment Channel Network
  - Lightning, on Ethereum
  - Only occasionally updates main chain for settlements
  - Smart contract
  - ERC20-compliant token transfers

# Plasma

- Nested blockchains
- Child chain attached to main ("root") chain
- Set of smart contracts (no change to consensus
- Fraud proofs – builds "court system"
- Struggles with supporting operations more complex than payments
- Scalability with enforceability
- Compatible with sharding, rollups etc.

Source: https://www.youtube.com/watch?v=nf1iEbBtbCE

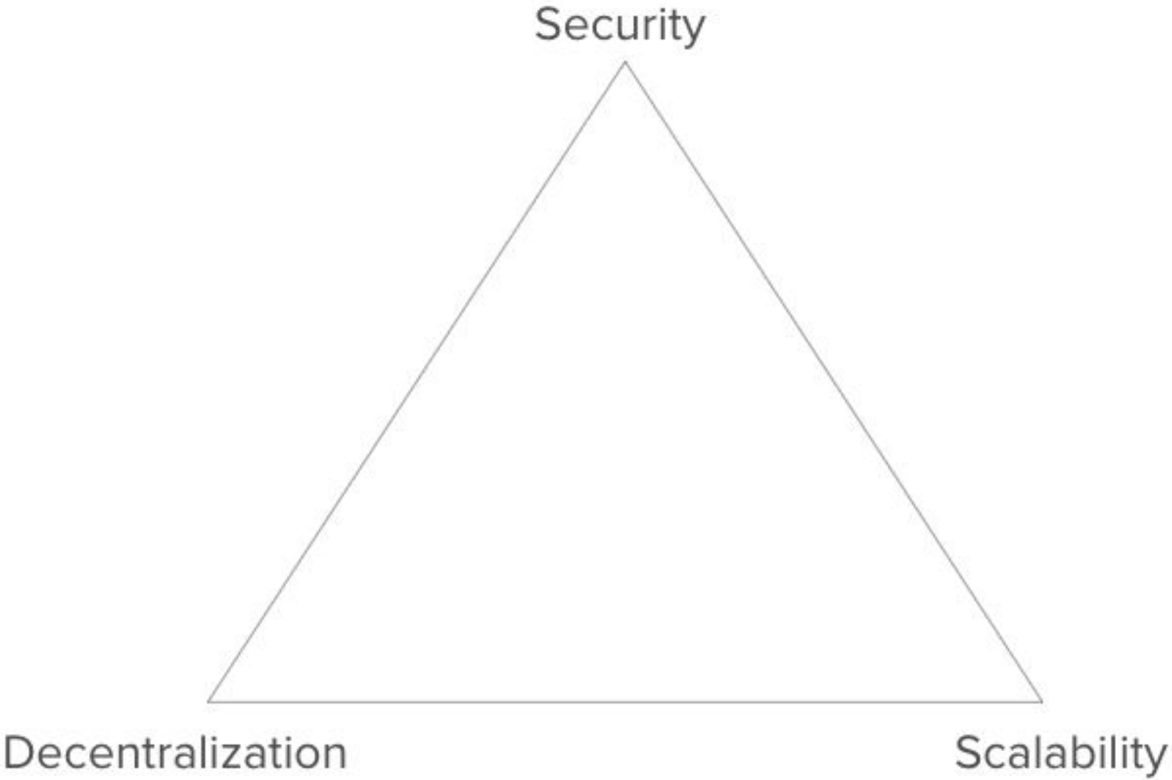
Layer 2

# Optimistic Rollups

- Data storage on-chain, computation off-chain
- Employs fraud proofs to verify correctness of off-chain computation
- Supports more complex operations (i.e. smart contracts)
- Ready for deployment!
- Provides less of an increase in scalability than other solutions

Layer 2

# Scalability Triangle



# QUESTIONS?

Appendix

# Ethereum 2.0

- Great scalability use case to track:
  - Phase 0: Transition to Proof of Stake
  - Phase 1: Data Sharding
  - Phase 2: State + Execution (computation and smart contracts)
  - Phase 3+: Other scaling solutions
- Multiple year timeline, but existing scaling solutions like optimistic rollups show a lot of promise for short-term scaling

Source: https://forkast.news/vitalik-buterin-explains-ethereum-2-0s-four-phases-sharding-scaling-proof-of-stake-and-more/

# THANK YOU