HYPERLEDGER SAWTOOTH

App Development

Zac Delventhal

Bitwise IO / Senior Engineer
Hyperledger Ambassador

github.com/delventhalz
@delventhalz

What is a Blockchain?

What is Hashing?

Creates a deterministic digest of some arbitrary data

Even slightly different data produces a completely different hash

```
sha256('Hello, World!')
// dffd6021bb2bd5b0af676290809ec3a53191dd81c7f70a4b28688a362182986f

sha256('Hello, World?')
// f16c3bb0532537acd5b2e418f2b1235b29181e35cffee7cc29d84de4a1d62e4d
```

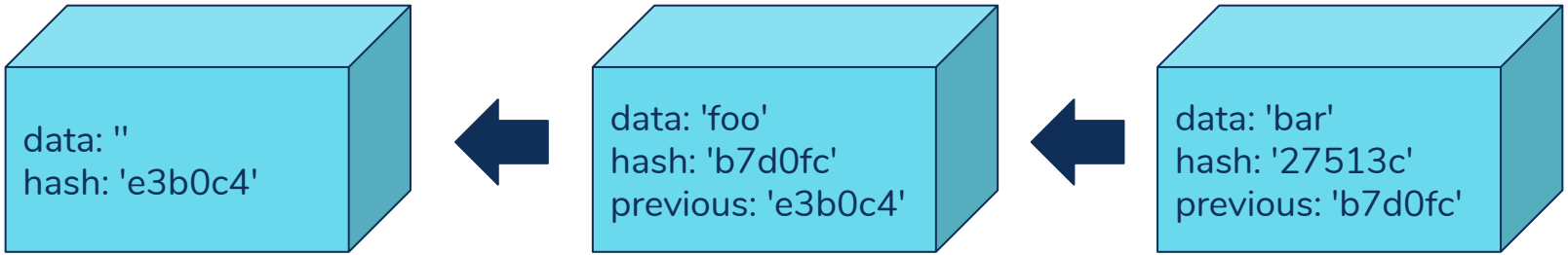
Blockchain Structure

Uses hashes to link discrete "blocks" of data

Hash is generated by combining data with previous hash

"Genesis" block only one allowed to have no previous hash

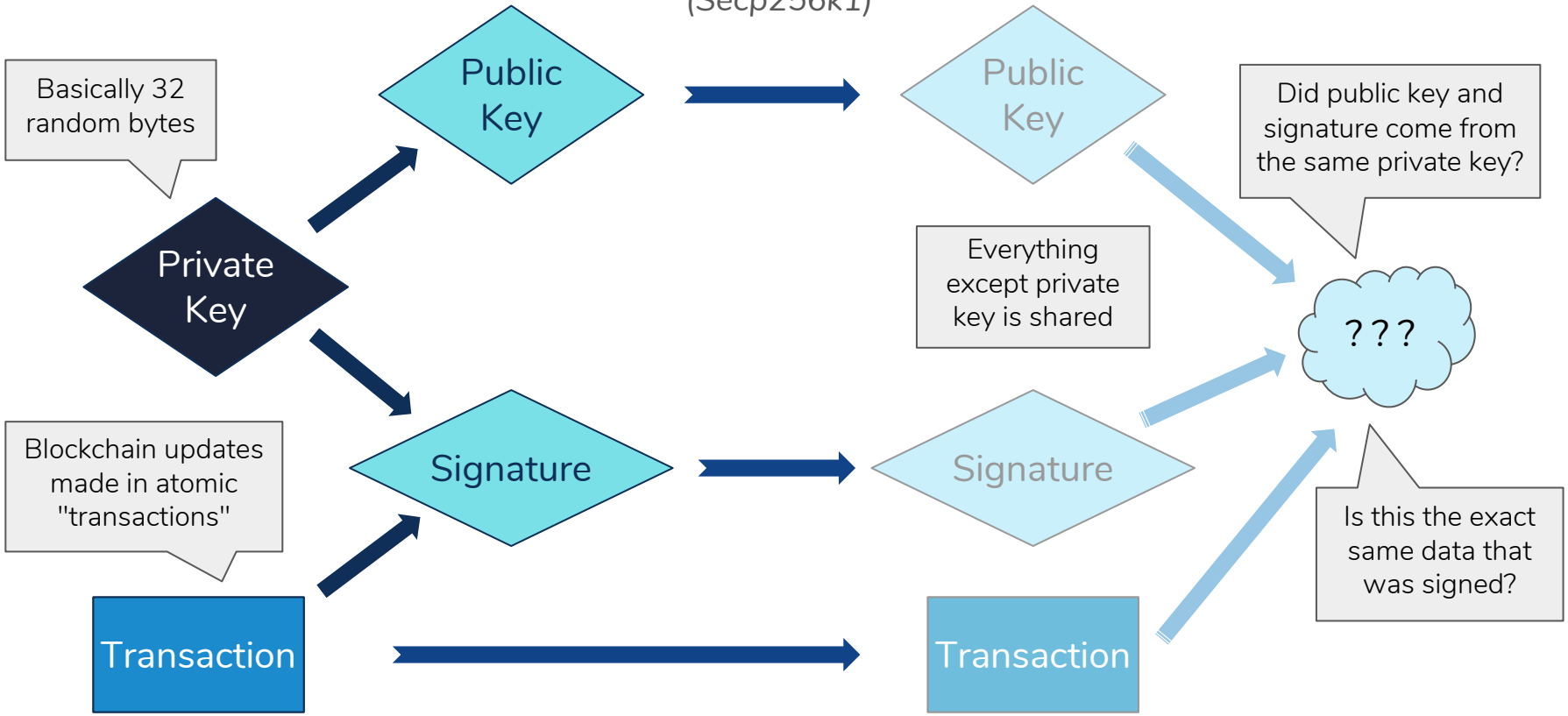
Immutable. Cannot alter blocks without altering **every** later block



sha256(data + previous) // first six chars displayed

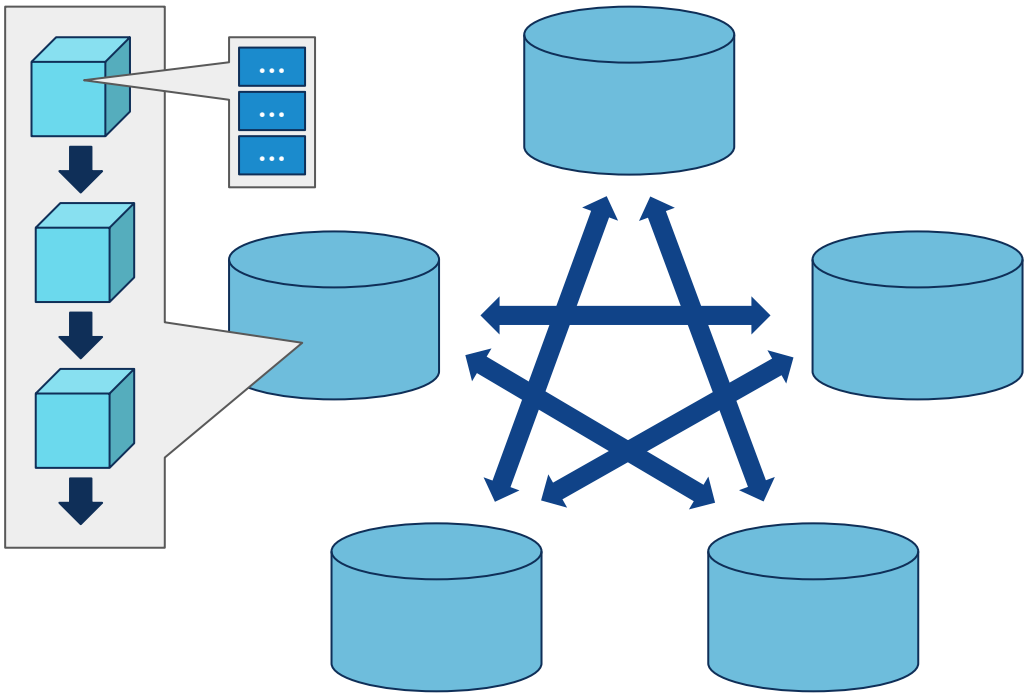
Signing

(Secp256k1)



Consensus

(Byzantine Fault Tolerance)



Use a lottery to determine who gets to create the next block

Always prefer "longest" chain, bad actors need 51% to catch up

Bitcoin and Ethereum use **Proof of Work** to randomly choose a "leader"

Sawtooth features **Proof of Elapsed Time** (and others)

Why Blockchain?

Share a database between mutually distrusting organizations

No reliance on a trusted third party

Immutable transaction history

High availability

- Crash fault tolerant
- Byzantine fault tolerant
- Liveness

Why NOT Blockchain?

Wrong use case:

- Internal-only business model
- A centralized blockchain is just bad database

Active areas of research:

- Transaction throughput
- "Private" transactions

What is Sawtooth?

What is Sawtooth?

Began as a hardware experiment in **Intel**'s labs

Contributed to **Linux Foundation** as a part of Hyperledger

Now a general purpose blockchain platform (not tied to specific hardware)

Designed first as a permissioned blockchain for consortiums

Permissioned Blockchains

Targeted at small(ish) groups

Endpoints are restricted, not available to the general public

Not centralized

No single canonical peer-to-peer network (deploy your own!)

No need for a currency (different incentives)

Why Sawtooth?

Highly modularized

Designed to scale:

- On-chain settings
- Smart contracts
- Parallel transaction processing

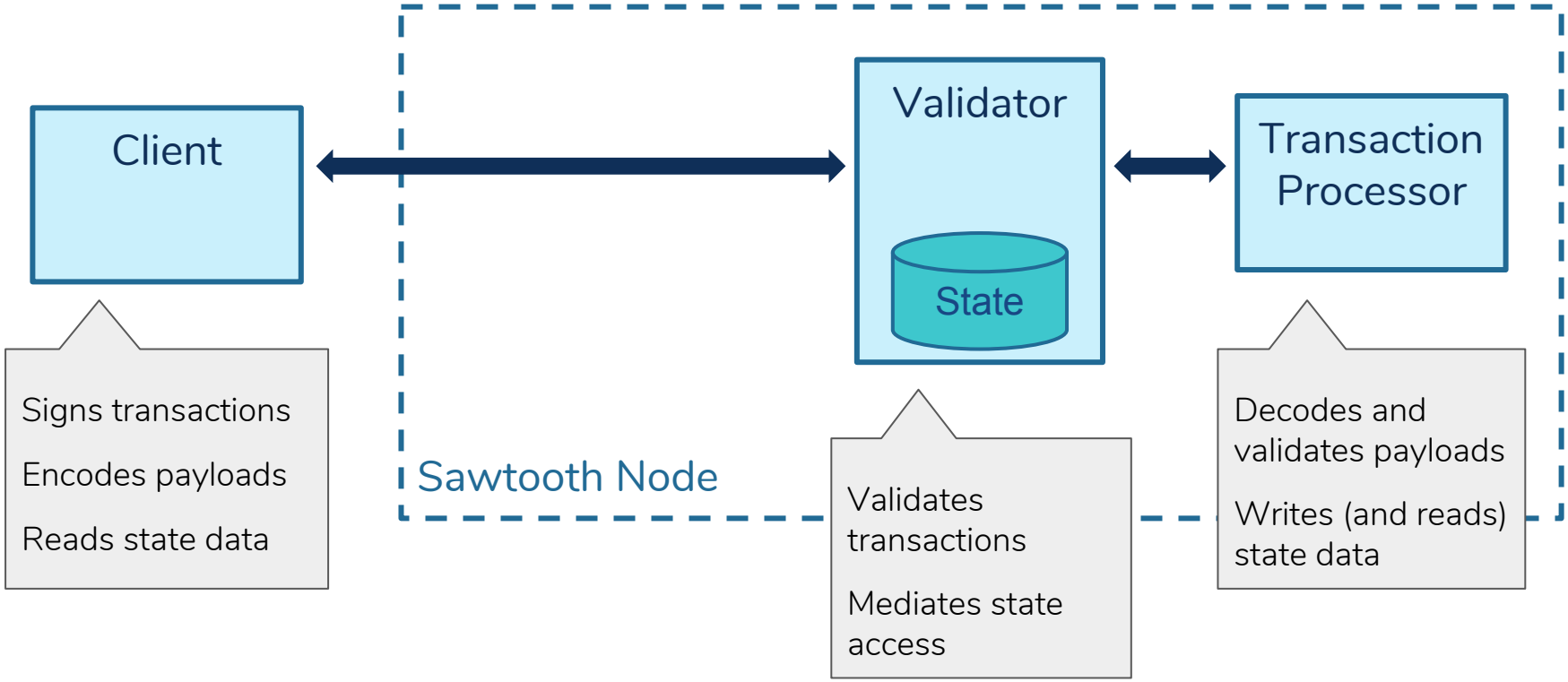
Byzantine Fault Tolerance (PoET)

Global state agreement

Develop in a variety of languages:

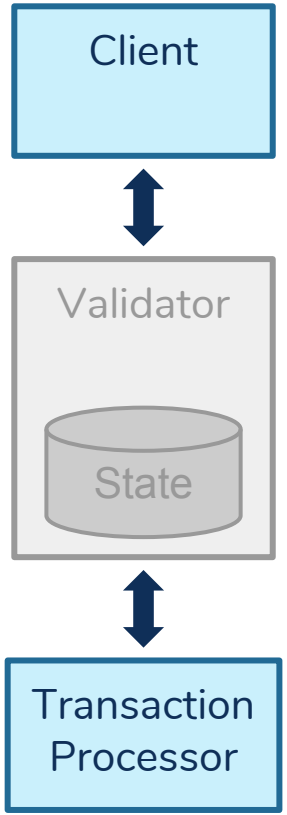
- Javascript
- Python
- Rust
- Solidity
- Go
- Java

Architecture



Building a Sawtooth App

Processor vs Contract



App logic is off-chain

Must manually deploy TPs to all participants

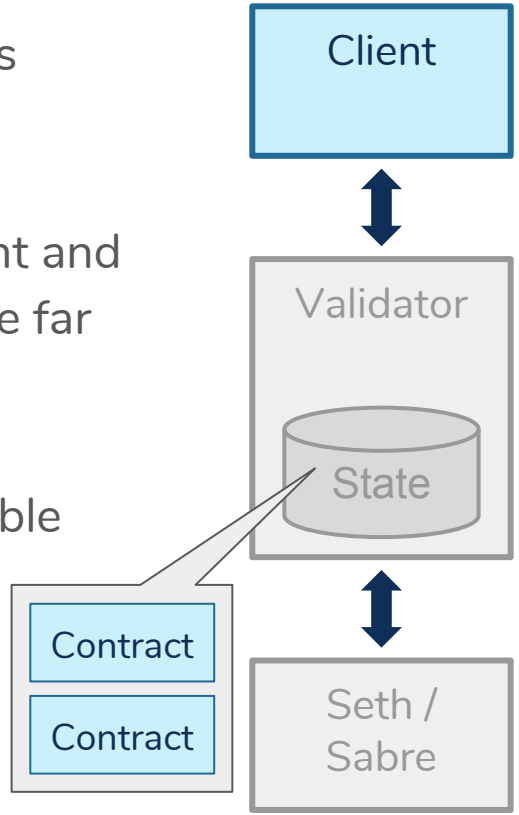
Transaction processor model is faster, simpler

More mature

App logic is on-chain

Deployment and updates are far easier

More scalable



Transaction Processor

Communicates with validator over ZMQ

The SDK handles most of the networking details

Handles transactions specific to one " transaction family"

Implements an `apply()` method called for each transaction

apply()

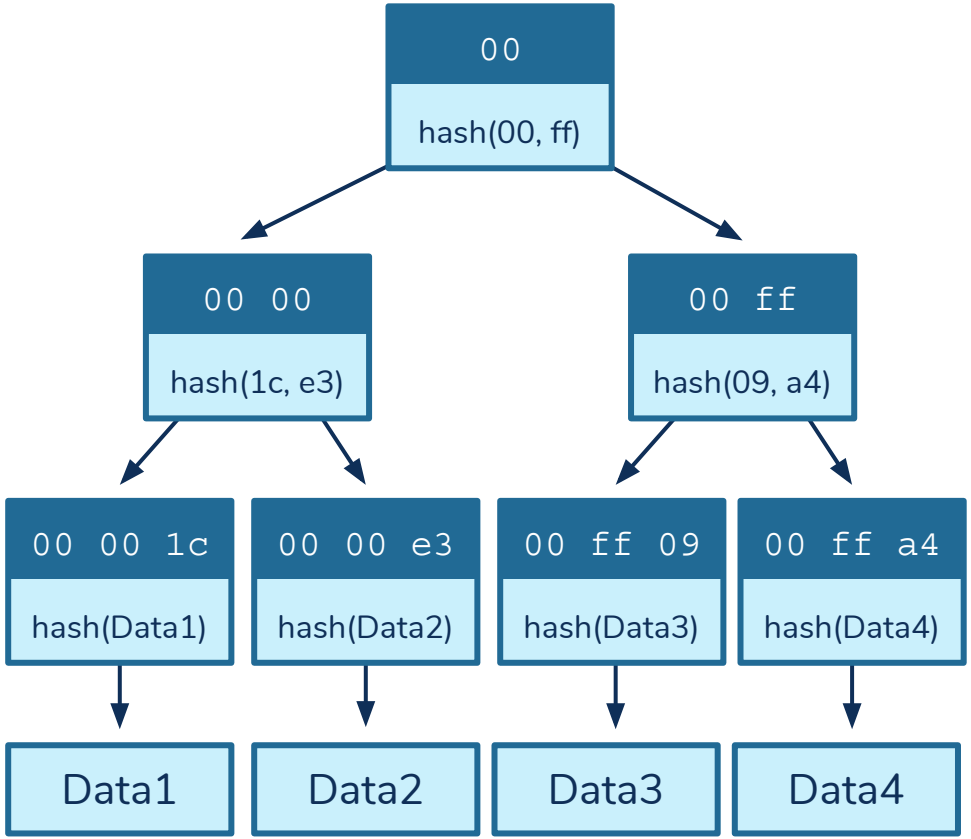
Both transaction processors and Sabre smart contracts follow this pattern

Receives a **transaction** request and a **state context** with access methods

```
def apply(txn, context):
    try:
        entity = handle(decode(txn.payload))
    except Exception as e:
        raise InvalidTransaction(e)

    context.set_state({getAddress(entity): encode(entity)})
```

State (the merkle tree)



State is stored in a "merkle tree" or "hash tree" so it can be verified

Each node is addressed with a single byte (i.e. two hex characters)

Any arbitrary data can be stored at any address you like

Prefixes can be used to quickly fetch all child nodes

The Client

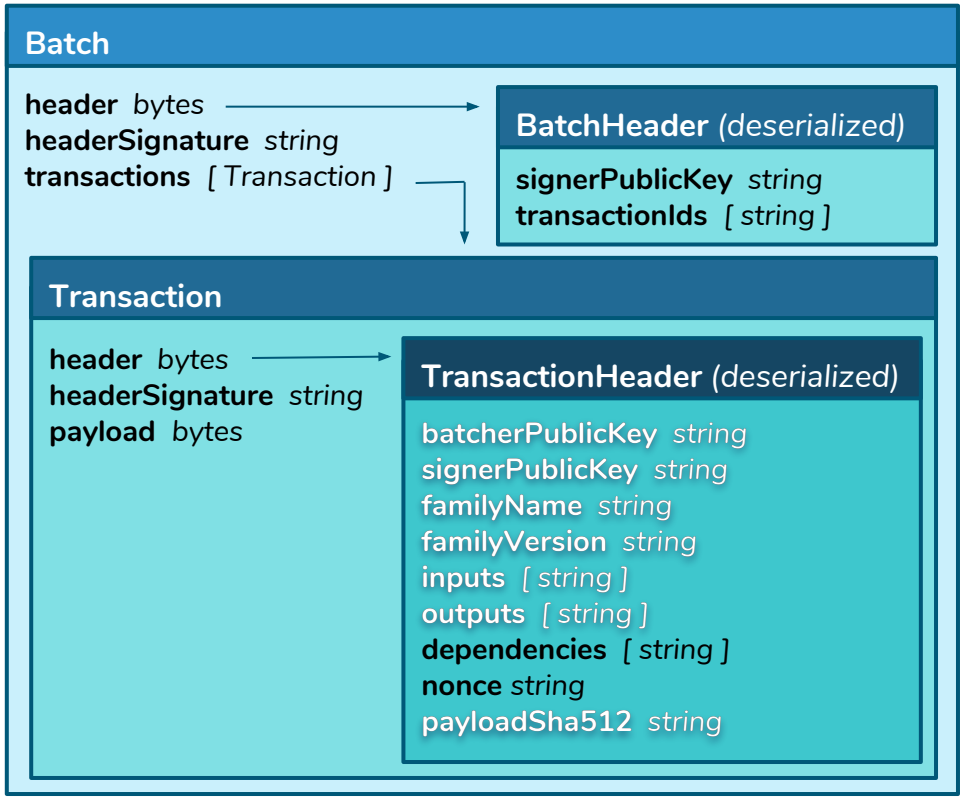
Can be simple or complex:

- a CLI
- a web app
- a whole ecosystem with its own custom REST API and a local copy of state in a database

Responsible for:

- encoding payloads
- creating and signing **transactions** and **batches**
- submitting transactions
- fetching and displaying state data for users

Transactions and Batches



Every transaction in a batch must be valid, or the batch is rejected

Signature also serves as a unique id

Provided by the SDK as protobufs

Signed and encoded by the client

Inputs/outputs can be prefixes, but ideally are full addresses

REST API Routes

POST /batches

- Submit transactions to the validator
- Body is a BatchList protobuf
- Use the header: `"Content-Type: application/octet-stream"`

GET /state?address={prefix}

GET /state/{address}

- Fetches encoded entities from state

ws: /subscriptions

- Websocket subscription to new state changes

YARRRRRRRR

(demo)

Links

GitHub
github.com/hyperledger/sawtooth-core

Docs
sawtooth.hyperledger.org

Chat
chat.hyperledger.org/channel/sawtooth

Talk Like A Pirate
github.com/delventhalz/pirate-talk

Hyperledger Cryptomoji
github.com/hyperledger/education-cryptomoji

That's it! Thank you!

Questions?