Blockchains for Fun and Profit

Midwest JS 2018

Zac Delventhal

Sawtooth Maintainer
Hyperledger Ambassador

github.com/delventhalz
@delventhalz

What is a Blockchain?

What is a Blockchain?

A blockchain is a growing list of records, called blocks, which are linked using cryptography.

Each block contains a cryptographic hash of the previous block and transaction data (generally represented as a merkle tree root hash).

Wikipedia

What is Hashing?

Creates a deterministic, fixed-length, digest of some arbitrary data

Even slightly different data produces a completely different hash

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

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

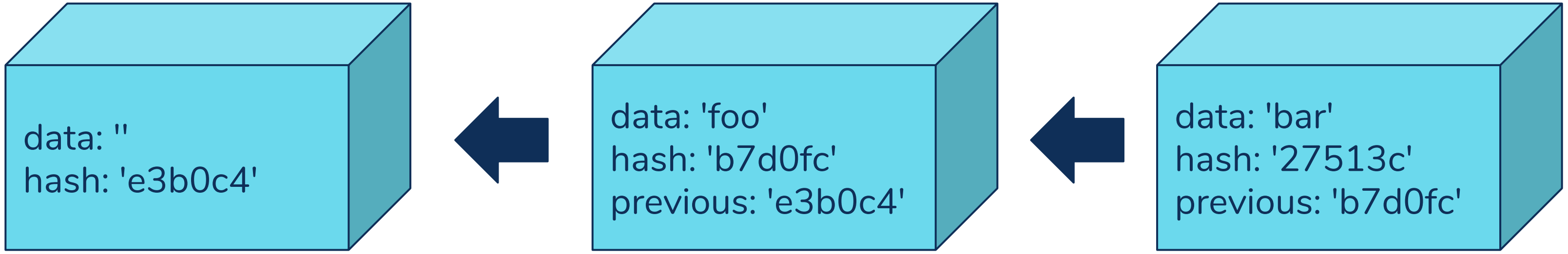
Basic Blockchain Structure

Hashes link discrete "blocks" of data

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

Hash is generated by combining the data with previous hash

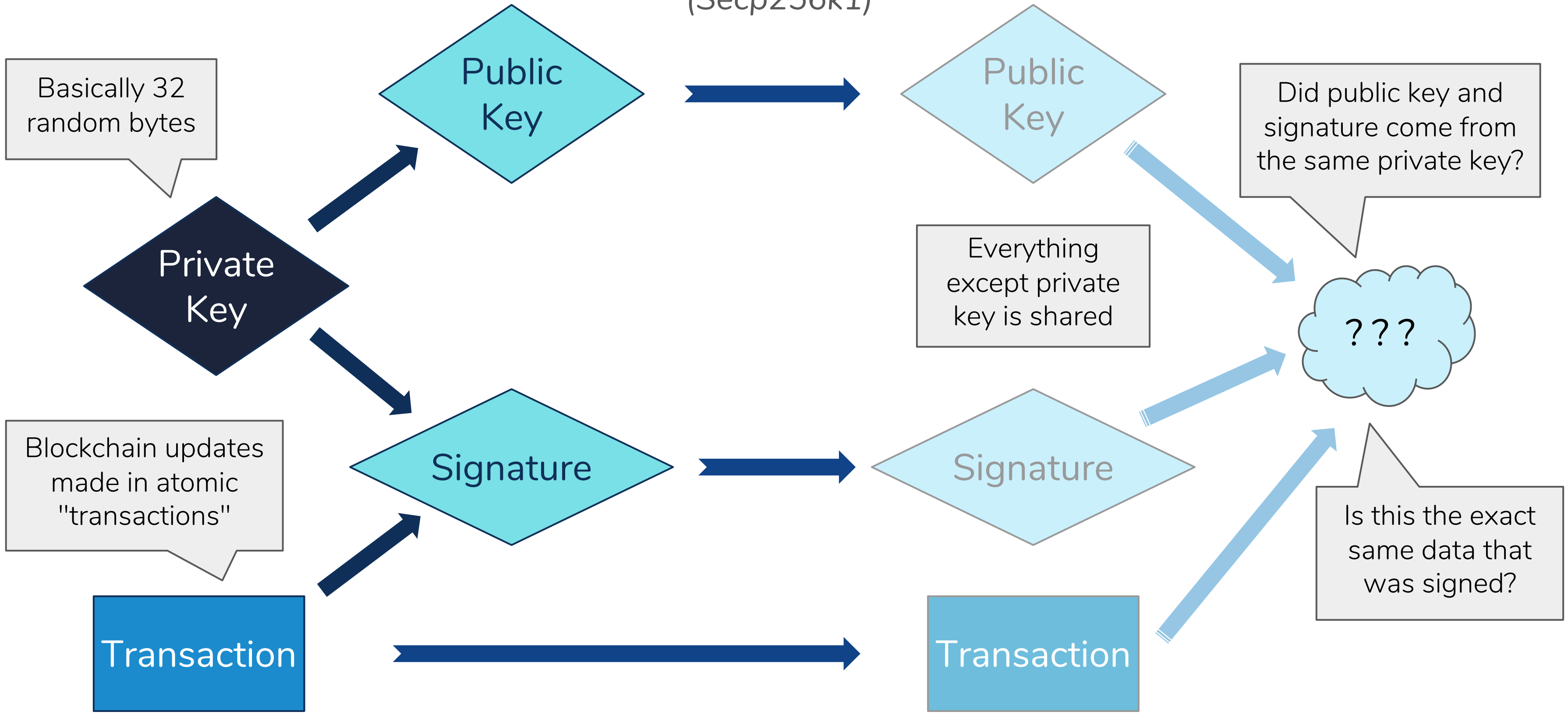
Immutable. Cannot alter blocks without altering **every** later block



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

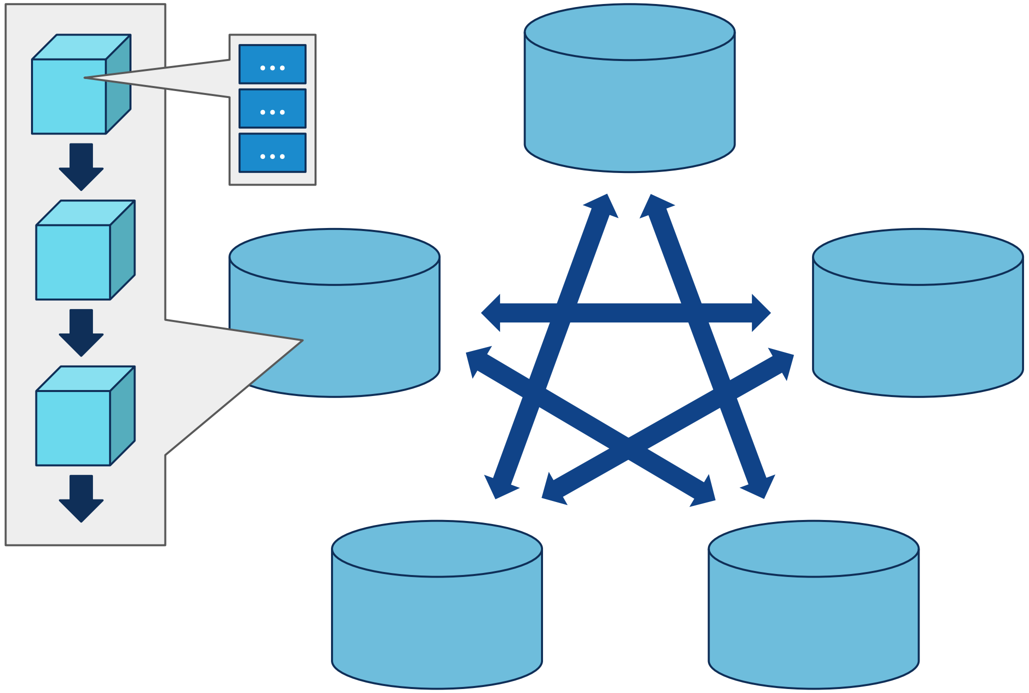
Signing

(Secp256k1)



Basically 32 random bytes
Private Key
Public Key
Signature
Blockchain updates made in atomic "transactions"
Transaction
Public Key
Everything except private key is shared
Signature
Transaction
Did public key and signature come from the same private key?
? ? ?
Is this the exact same data that was signed?

Consensus



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

Always prefer "longest" chain, bad actors need 51% control

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

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

Distributed Applications

What is a Distributed Application?

A program which relies on multiple nodes in a peer-to-peer network (i.e. a blockchain) to perform certain tasks

Also referred to in some contexts as a "smart contract"

May be fully custom or rely on one of the emerging general purpose blockchain platforms like Ethereum or Sawtooth

Insufferably abbreviated "dApp"

Why a Blockchain?

Share data between mutually distrusting entities

No reliance on trusted third parties

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

Bitcoin

A digital currency, the first to solve the double-spend problem without a trusted third party

Invented the concept of a blockchain

Faces scaling issues, very difficult to upgrade

Cryptokitties

Collect, breed, and sell unique immutable digital cats

One of the first blockchain games

Introduced the concept of "cryptocollectibles"

Built on **Ethereum**

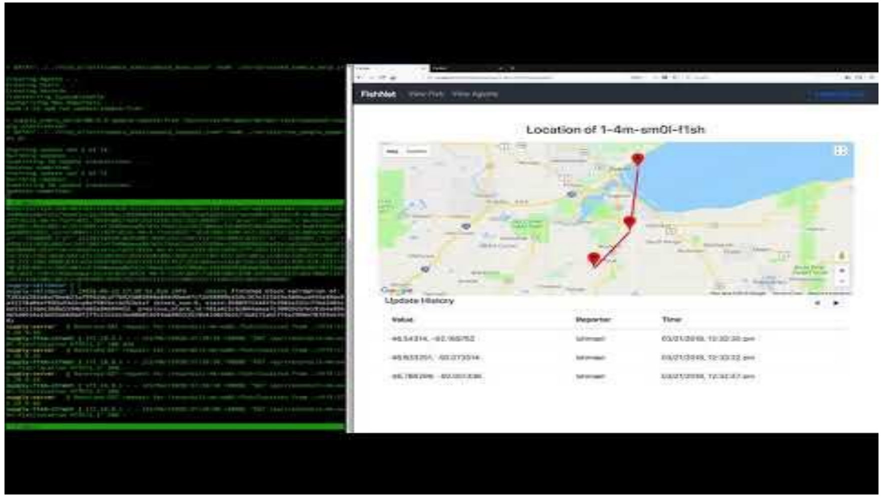
Sovrin

Decentralized digital identities

Goal is to be a portable global replacement for all online identity verification

Relies on **zero knowledge proofs** to limit the visibility of sensitive information

Sawtooth Supply Chain



Location of 1-4m-sm0l-f1sh

Update History

Value	Reporter	Time
46.54314, -82.165752	[illegible]	03/21/2018, 12:30:30 pm
46.633201, -82.273314	[illegible]	03/21/2018, 12:33:32 pm
46.766206, -82.001336	[illegible]	03/21/2018, 12:32:37 pm

Track the provenance of assets from production to consumption

Global agreement on data like location, certifications, and ingredients

Built on **Hyperledger Sawtooth**

What is Sawtooth?

Hyperledger Sawtooth

Began as a hardware experiment in Intel's labs

Contributed to Linux Foundation as a part of Hyperledger

General purpose blockchain platform

Designed as a **permissioned** blockchain for consortiums

Not tied to specific hardware

What is a Permissioned Blockchain?

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 use Sawtooth?

Highly modularized

Designed to scale:

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

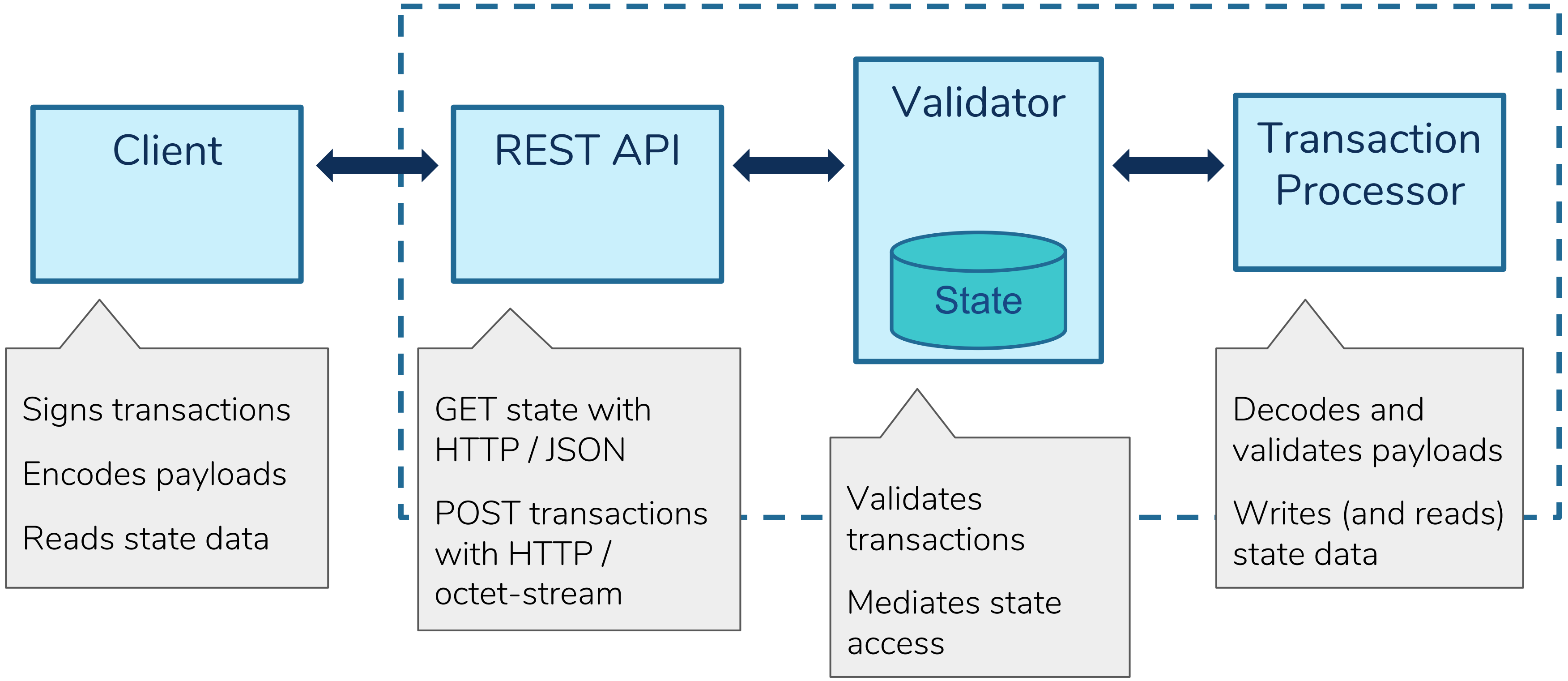
Byzantine Fault Tolerance (PoET)

Global state

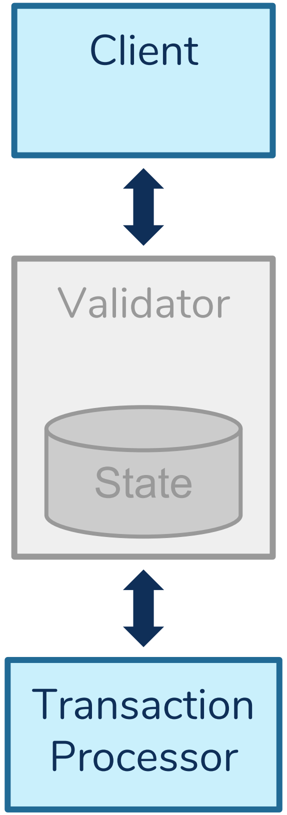
Develop in a variety of languages:

- Javascript
- Python
- Rust
- Solidity (Ethereum VM)
- Go
- Java

Architecture



Processor vs Contract



App logic is off-chain

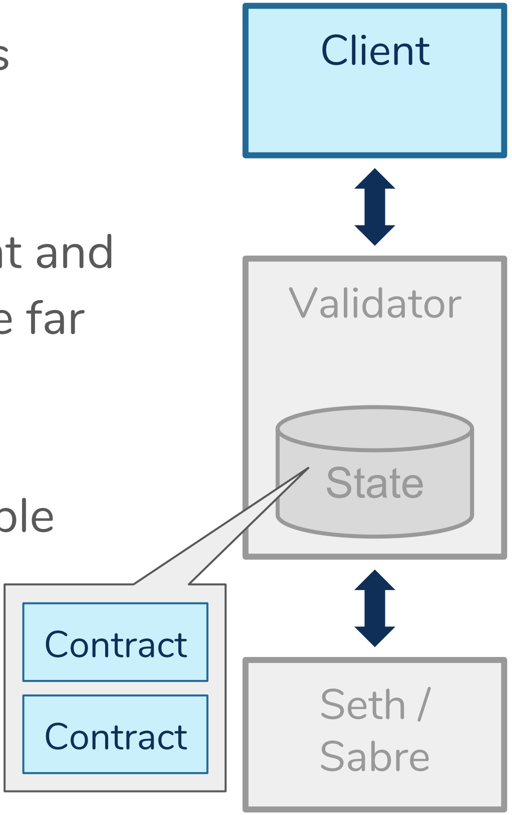
Must manually deploy TPs to all participants

Transaction processor model is faster, simpler

App logic is on-chain

Deployment and updates are far easier

More scalable



Building a Sawtooth App

The Client

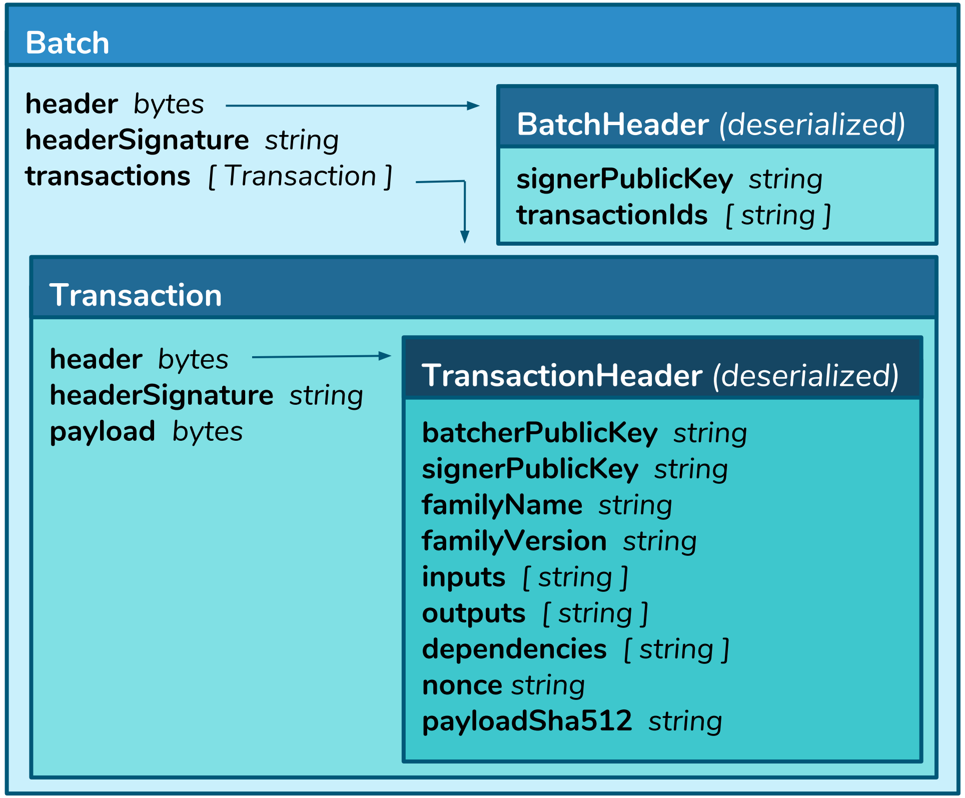
Can be simple or complex:

- 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



Encoded as Protobufs

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

Family name/version designate a transaction processor

Inputs/outputs are the addresses in state you expect to read/write

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

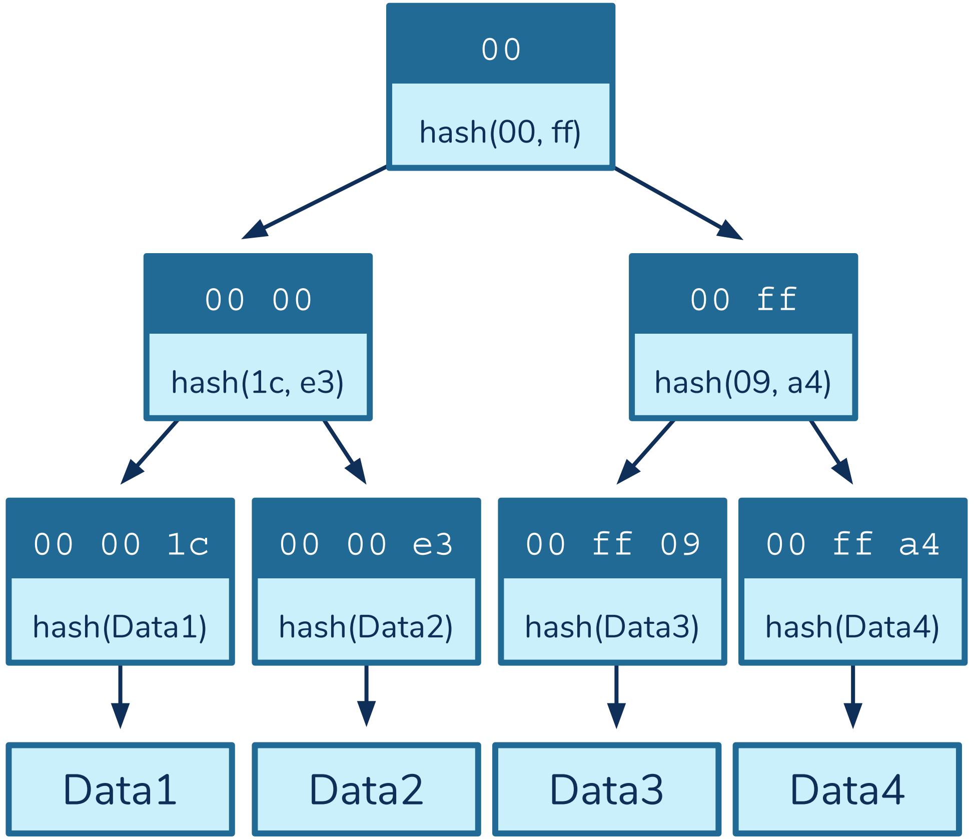
apply()

Both transaction processors and Sabre smart contracts follow this pattern

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

```
apply (txn, context) {
  try {
    const entity = handle(decode(txn.payload))
  } catch (err) {
    throw new InvalidTransaction(err)
  }
    return context.setState({ [getAddress(entity)]: encode(entity) })
}
```

The Merkle Tree



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

The leaves hash some data, and each parent hashes its children all the way up to the **state root hash**

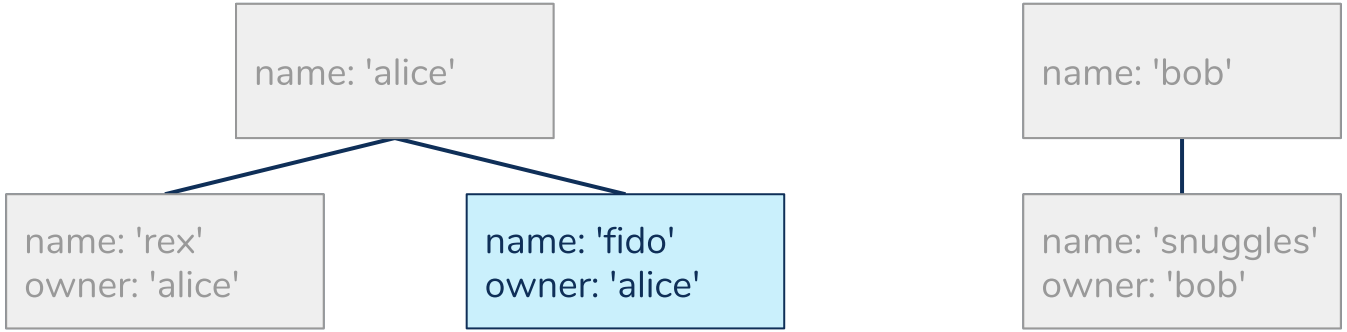
Each node is addressed with a single byte (i.e. two hex characters)

Any arbitrary data can be stored at any address you like, but consider how prefixes can be used to quickly fetch child nodes

Addressing Example

	Namespace	Type	Owner (optional)	Identifier
owner	sha512('pets').slice(0, 6)	'00'	--	sha512(name).slice(0, 62)
pet	sha512('pets').slice(0, 6)	'01'	sha512(owner).slice(0, 16)	sha512(name).slice(0, 46)

```
102fe1 01 408b27d3097eea5a 1f443e224ee0de0afdc46cd68820de1494687677bed1d3
 'pets'          'alice'                      'fido'
```



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?