PenguinCoin

CRYPTOCURRENCY IN TYPESCRIPT

SENSOURCE

sensourceinc.com

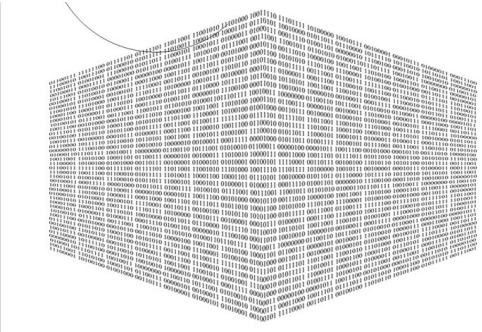
jdforsythe@gmail.com

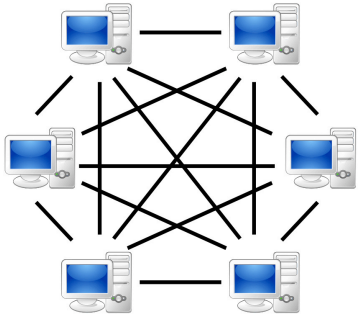
github.com/jdforsythe

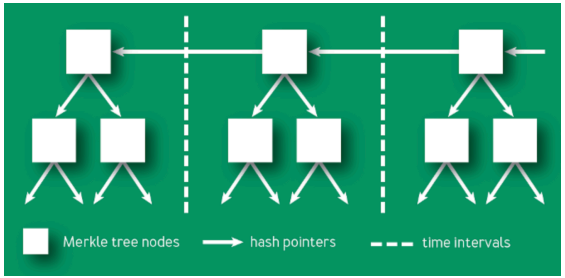
Jeremy Forsythe

The Blockchain

3 UNDERLYING TECHNOLOGIES







THE BLOCKCHAIN

Append-only (immutable) ledger

Keeps track of accounting records

PEER-TO-PEER NETWORKS

Connections are made between peers instead of through a central server or lieutenant servers

CONSENSUS

Peers agree on the ledger validity

Can't "double spend" or change

The Blockchain

An immutable, append-only accounting record

1	GENESIS BLOCK The first block and only one of its kind - it has no previous block to reference
2	BLOCK A Has a link to the previous block, the Genesis Block, as a hash listed inside
3	BLOCK B Has a link to the previous block, Block A, as a hash listed inside
4	BLOCK C Has a link to the previous block, Block B, as a hash listed inside
5	BLOCK D Has a link to the previous block, Block C, as a hash listed inside

The Blockchain

Use Cases

Track Movement of Things of Value

- E-Commerce
- Global Payments
- P2P Lending
- Microfinance

Immutable Records of Truth

- Healthcare Records
- Title Records
- Voting
- Intellectual Property

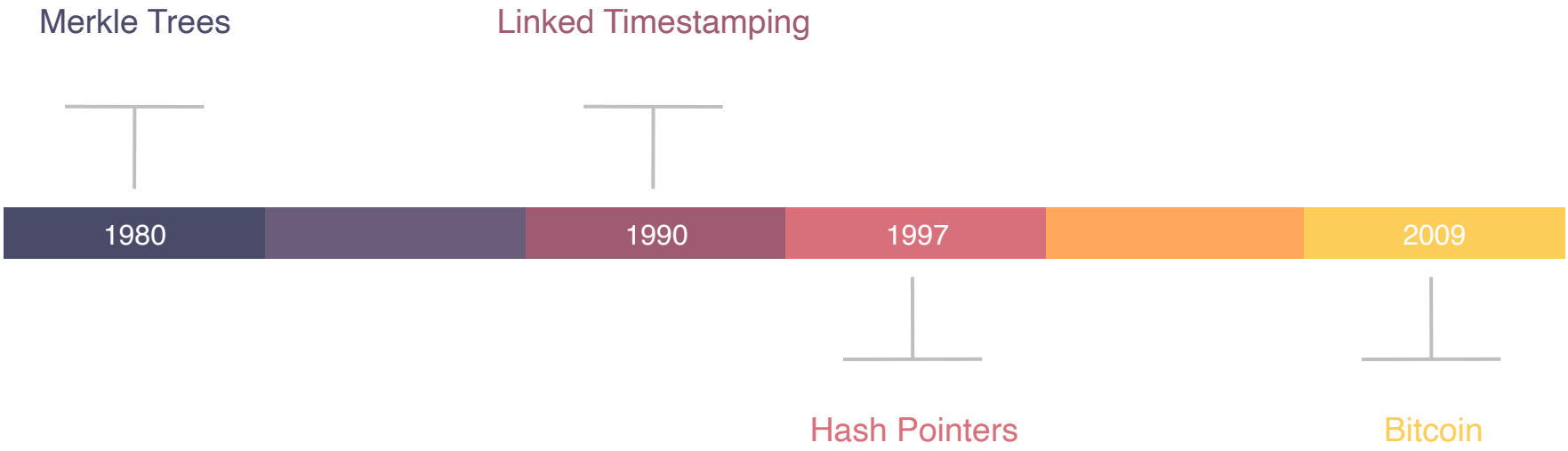
Smart Contract

- Digital Rights Management
- Wagers
- Escrow

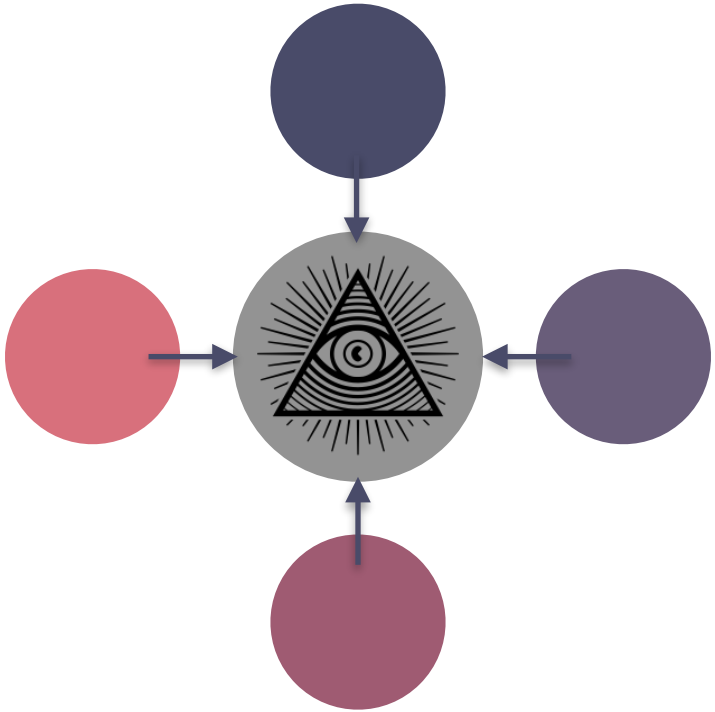
Bitcoin vs Blockchain

The Blockchain

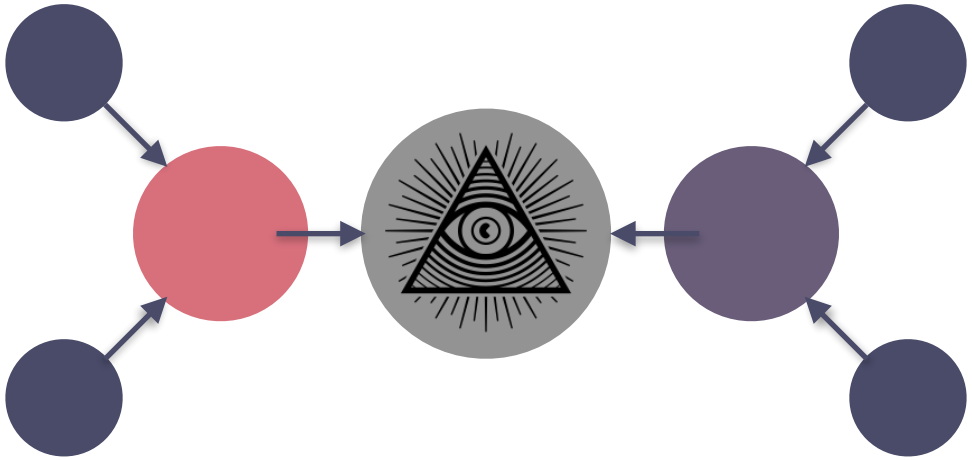
It's not new...



Trusted Authority - Centralized



Middleman / Lieutenant - Decentralized



Peer-to-Peer - Distributed

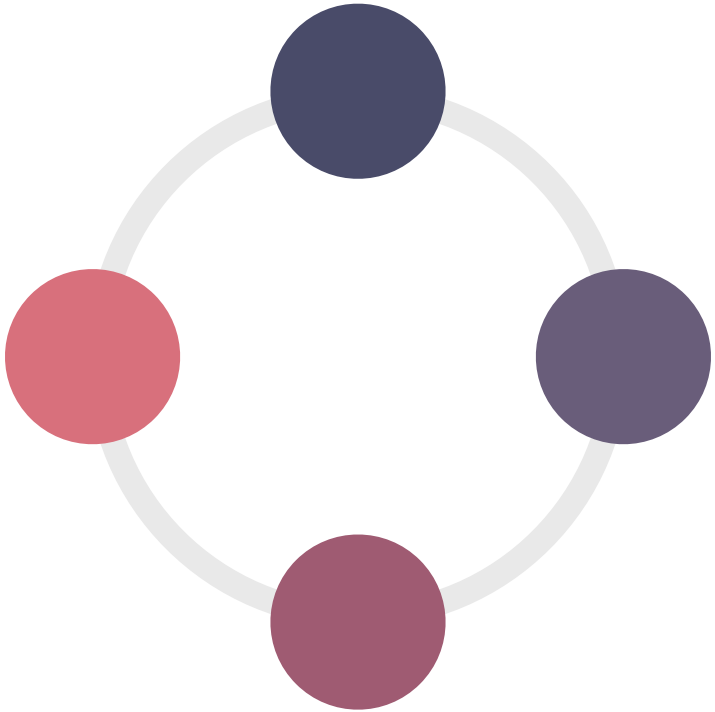
Entire ledger is distributed across every node

You can always audit for yourself

No relying on one party for truth, security, equity, and reliability

No relying on a single point of failure

Assume you can't trust anyone



Peer-to-Peer - Distributed

C2 **Business** M 1 | TUESDAY | SEPTEMBER 16, 2008 | ST. LOUIS POST-DISPATCH | STLTODAY.COM

Lehman Brothers files biggest bankruptcy

Choked by credit crisis, investment bank seeks protection, will sell units.

By Vinnee Tong and Joe Bel Bruno
THE ASSOCIATED PRESS

NEW YORK • Lehman Brothers, a 158-year-old investment bank

age firms that are insured by the Securities Investor Protection Corp.

The SEC noted in a statement that Lehman's decision to file for bankruptcy protection does not

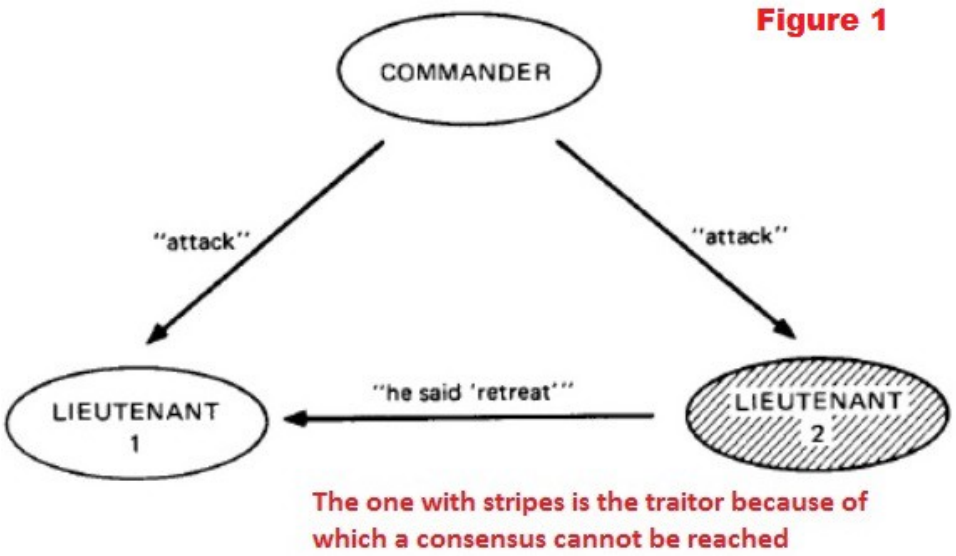
UBS
SOCIETE GENERALE
Deutsche Bank
Raiffeisen BANK
Bank of America
ING
TD Bank
CHASE
citi

Assume you can't trust anyone

Assume you can't trust anyone

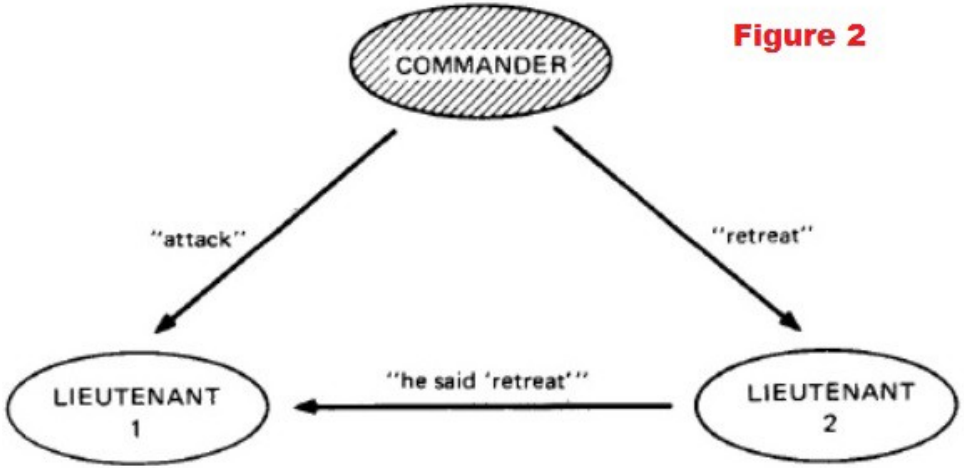
Consensus Mechanism

Fault tolerant distributed computing



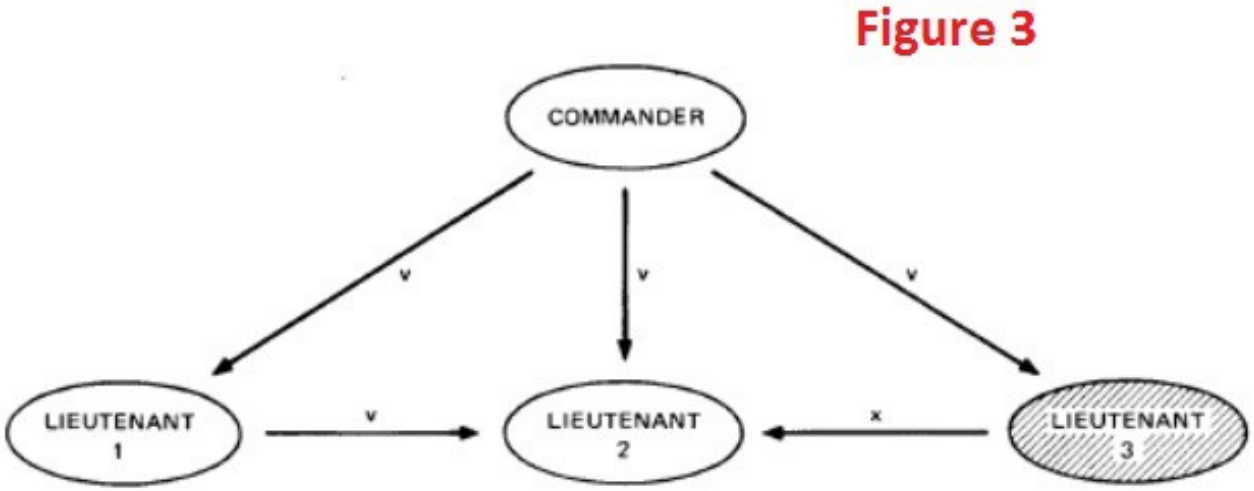
Consensus Mechanism

Fault tolerant distributed computing



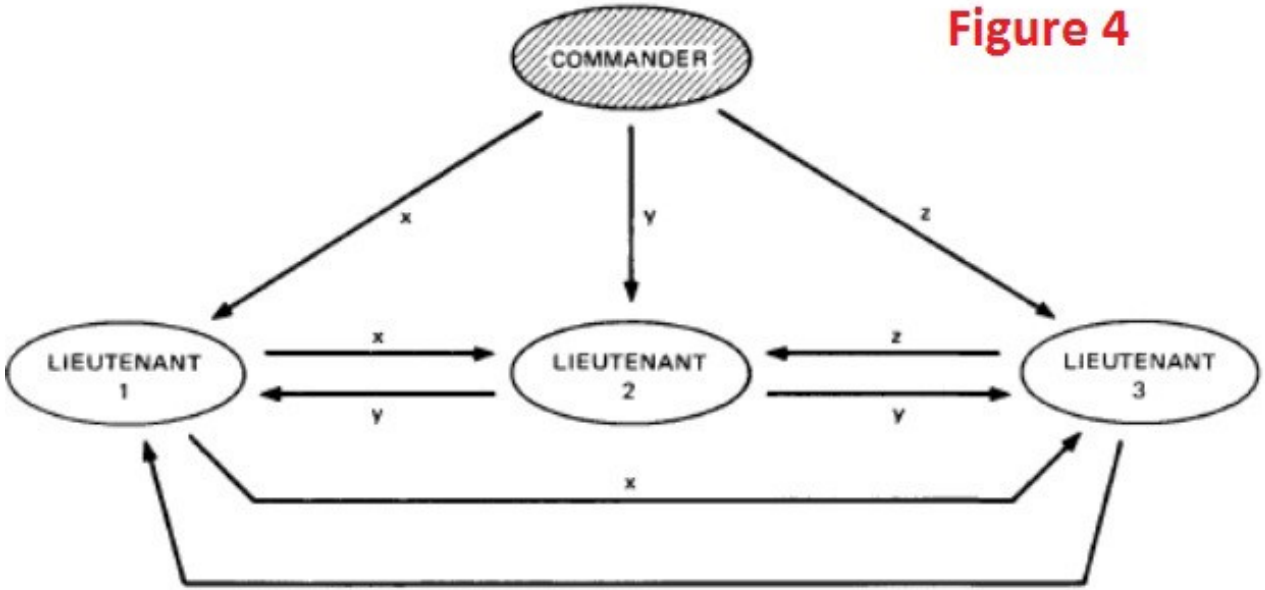
Consensus Mechanism

Fault tolerant distributed computing



Consensus Mechanism

Fault tolerant distributed computing



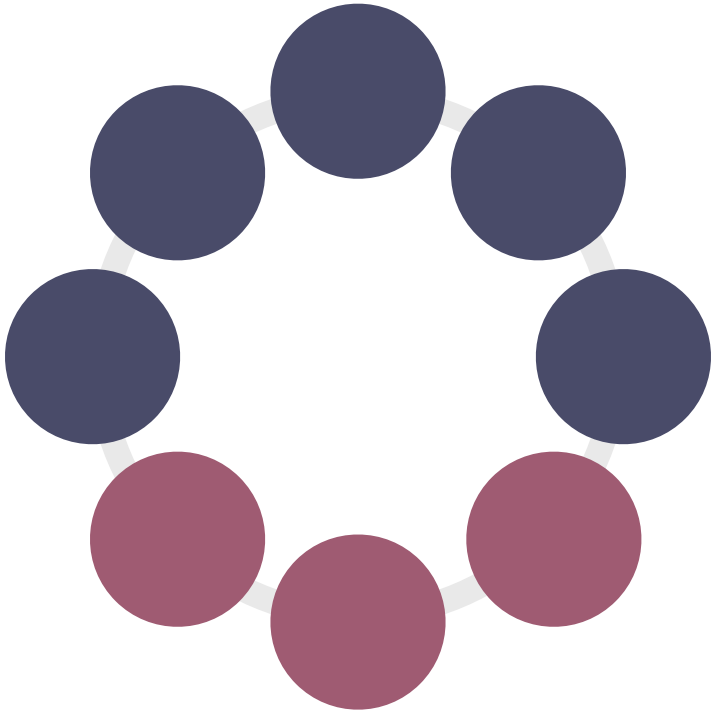
Consensus - Sybil Attack

In a p2p network there is no registration

Nodes freely join and leave

Can create enough Sybils (sockpuppet nodes) to overcome the network

Different but similar problem as DoS and spam



Proof of Work

Puzzle solving

Spam Deterrent

Proposed but unsuccessful

Captcha

Human PoW

1992

1997

2009

Preventing DoS

Hashcash

Use hashes for PoW - easy to verify, hard to create

Uses leading zeroes - 10 leading zeros = 2^10 tries avg

Bitcoin

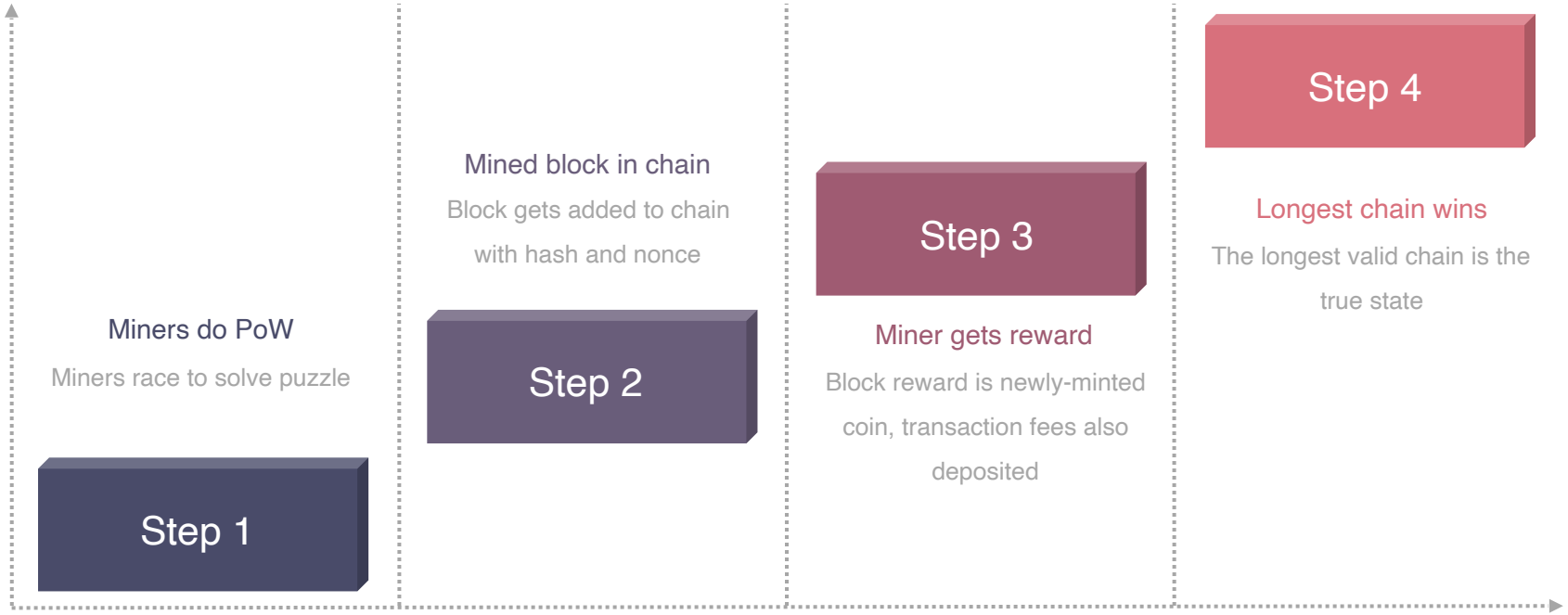
Use PoW to secure transactions

Coin itself is the data

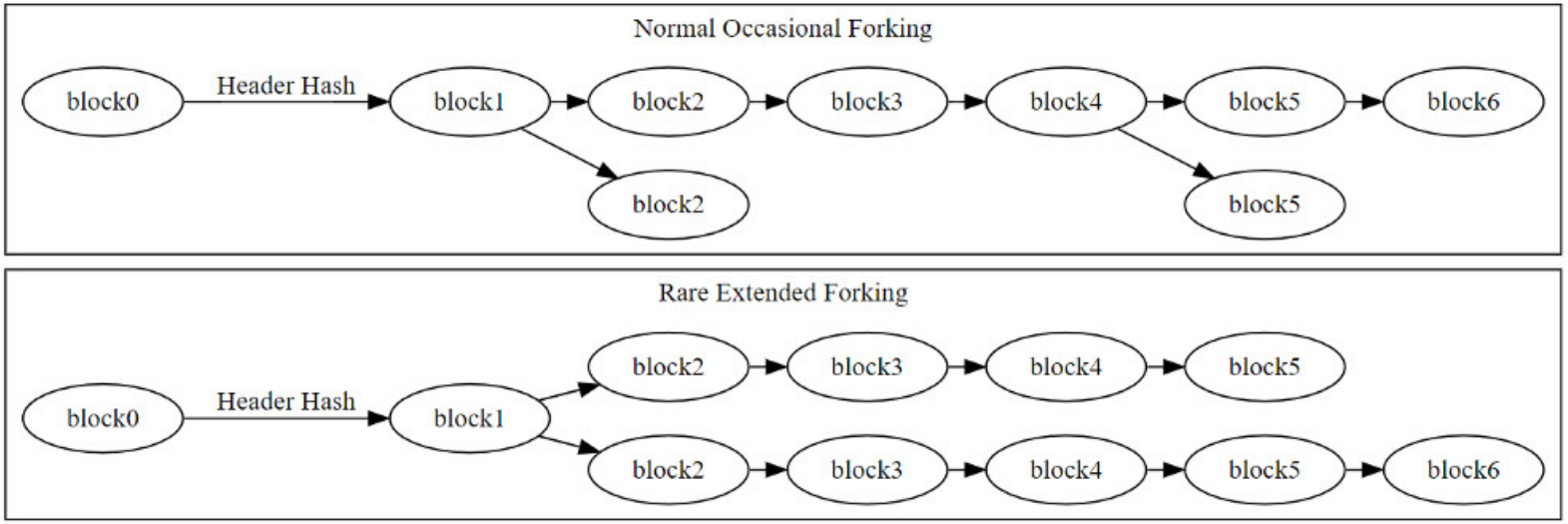
Reward for PoW and tx fees

Reward for being good

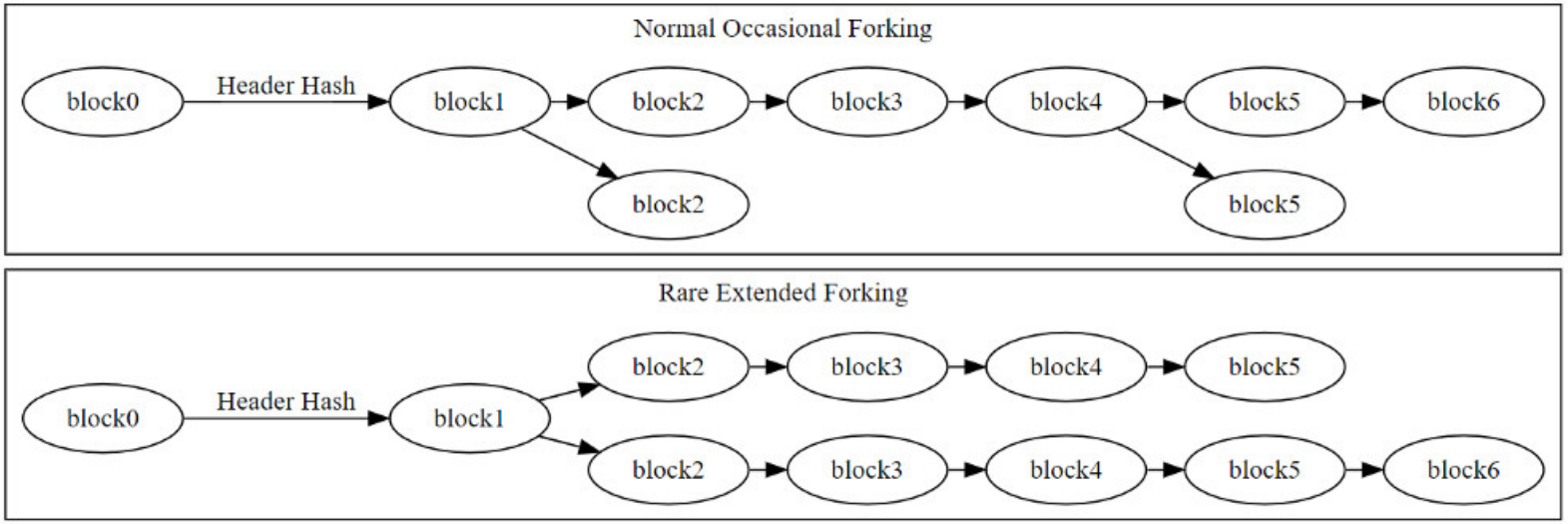
Consensus


Miners do PoW
Miners race to solve puzzle
Step 1
Mined block in chain
Block gets added to chain with hash and nonce
Step 2
Step 3
Miner gets reward
Block reward is newly-minted coin, transaction fees also deposited
Step 4
Longest chain wins
The longest valid chain is the true state

Longest Chain Wins



Longest Chain Wins



TypeScript

Superset of JavaScript with Types

```
function greet(person) {
    return `Hello, ${person}`;
}

let user = 'Jacob';

console.log(greet(user)); // Hello, Jacob
```

TypeScript

Superset of JavaScript with Types

```typescript
function greet(person: string) {
  return `Hello, ${person}`;
}

let user = 'Jacob';

console.log(greet(user)); // Hello, Jacob
```

TypeScript

Superset of JavaScript with Types

```
function greet(person: string) {
    return `Hello, ${person}`;
}

let user = [0, 1, 2];

console.log(greet(user));

// error TS2345: Argument of type 'number[]' is not assignable to parameter of type 'string'.
```

TypeScript

Superset of JavaScript with Types

```typescript
interface Person {
  firstName: string;
  lastName: string;
}

function greet(person: Person) {
  return `Hello, ${person.firstName} ${person.lastName}`;
}

let user = { firstName: 'Jennifer', lastName: 'Jones' };

console.log(greet(user)); // Hello, Jennifer Jones
```

TypeScript

Basic Types

```typescript
const isDone: boolean = false;

const decimal: number = 6;
const hex: number = 0xf00d;

const color: string = 'blue';

const list: number[] = [1, 2, 3];
```

TypeScript

Tuples and Enums

```typescript
// tuples
const x: [string, number] = ['hello', 42];

// enum
enum Color { Red, Green, Blue}
enum Color { Red = 1, Green, Blue }
let c: Color = Color.Green;
let colorName: string = Color[2]; // Green
```

TypeScript

Any type - don't use it!

```
// any type - don't use this - no type safety
let notSure: any = 4;
notSure = 'now a string';
notSure = false;
```

TypeScript

Void

```
// void
function warnUser(): void {
  console.log('Warning!');

  // implicit return
}

function warnAgain(): void {
  console.log('Warning again!');

  // explicit return
  return; // or `return undefined;`
}
```

TypeScript

Undefined and null

```typescript
// null and undefined
let u: undefined = undefined;
let n: null = null;

u = 1; // nope

let num: number = 1;
num = undefined; // yep
num = null; // yep
```

TypeScript

Type assertion

```
// type assertion - i know more than you do
let val: any = 'this is a string';

let strLength: number = (<string> val).length;
```

TypeScript

Interfaces and Extending Them

```typescript
interface Vehicle {
  color?: string;
  wheels: number;
}

interface Motorcycle extends Vehicle {
  wheels: 2;
}

interface Car extends Vehicle {
  wheels: 4;
}

const vStar: Motorcycle = {
  color: 'blue',
  wheels: 2,
};

const accord: Car = {
  // color is optional
  wheels: 2, // nope
}
```

TypeScript

Enums in Interfaces, Union and Intersection Types

```typescript
enum MessageType {
  Incoming = 1,
  Outgoing,
};

interface Message {
  type: MessageType;
  message: string;
}

interface IncomingMessage extends Message {
  type: MessageType.Incoming;
  source: string;
}

interface OutgoingMessage extends Message {
  type: MessageType.Outgoing;
  destination: string;
}

// union type (one or the other)
type AnyMessage = IncomingMessage | OutgoingMessage;

// intersection type (both)
type MixedMessage = IncomingMessage & OutgoingMessage;
```

TypeScript

Generics

```typescript
// generics
interface Container<T> {
  id: number;
  data: T;
}

const box: Container<number> = {
  id: 1,
  data: 42,
}

// type alias
type NumberContainer = Container<number>;

// multiple type parameters
interface DoubleContainer<T, K> {
  id: number;
  dataOne: T;
  dataTwo: K;
}

// functions with generic types
function getArray<T>(val: T): T[] {
  return [val];
}
```

Code Demo

CRYPTOCURRENCY IN TYPESCRIPT

SENSOURCE

sensourceinc.com

jdforsythe@gmail.com

github.com/jdforsythe

Jeremy Forsythe