



BLOCKCHAIN
TRAINING ALLIANCE



BLOCKCHAIN
TRAINING ALLIANCE

Ethereum Development Training

Revision 3.2

www.blockchaintrainingalliance.com

●○○

Let's Start At the Beginning

No prior knowledge
of *Blockchain* required,
“Coding” Knowledge is beneficial

Ethereum Blockchain and Solidity
programming language will be covered in-depth

Start with a simplified overview of how it all works, then dive
deeper into each section



What will I learn?

- ❖ What are Use cases for Ethereum?
 - ❖ Learn about the many areas where Ethereum is being used
- ❖ What does an Ethereum app look like?
 - ❖ Learn what real life decentralized apps look like and what the complete cost are
- ❖ How do I design an Ethereum Dapp?
 - ❖ Basic principles of designing an Ethereum blockchain application
- ❖ How do I develop an Ethereum Dapp?
- ❖ How do I test an Ethereum Dapp?

Course Modules



- ❖ Modules covered:
 - ❖ What is Blockchain?
 - ❖ Blockchain and Cryptocurrency
 - ❖ Why Use Blockchain
 - ❖ Decentralized Networks and Ledgers
 - ❖ Blockchain 2.0 and Ethereum
 - ❖ Programmable Blockchains “Smart Contracts”
 - ❖ Types of Blockchain
 - ❖ Ethereum Blockchain Tooling Infrastructure
 - ❖ How Blocks are Created
 - ❖ Cryptography and Hashing
 - ❖ Blockchain Nodes And Web3
 - ❖ Smart Contract Life Cycle
 - ❖ Smart Contract Concurrency and Events
 - ❖ Solidity Functions and Modifiers
 - ❖ Solidity Mappings and Structs
 - ❖ Solidity Files and Inheritance
 - ❖ Understanding Transactions And Gas
 - ❖ Types of Consensus
 - ❖ Mining a Block
 - ❖ Ethereum Tooling
 - ❖ Go-Ethereum In-Depth
 - ❖ Truffle and Local Development
 - ❖ Truffle Deployment/Migration
 - ❖ Truffle Unit Testing
 - ❖ Security & Upgradeable Contracts
 - ❖ Smart Contracts Best Practices
 - ❖ Blockchains of Storage
 - ❖ Compilation of Smart Contracts
 - ❖ Blockchain Use Cases
 - ❖ Blockchain Adoption
 - ❖ Web 3.0
 - ❖ Blockchain Implementation



What is Blockchain?

The Basics

Immutable
Security Trustless



The Core Principles of Blockchain

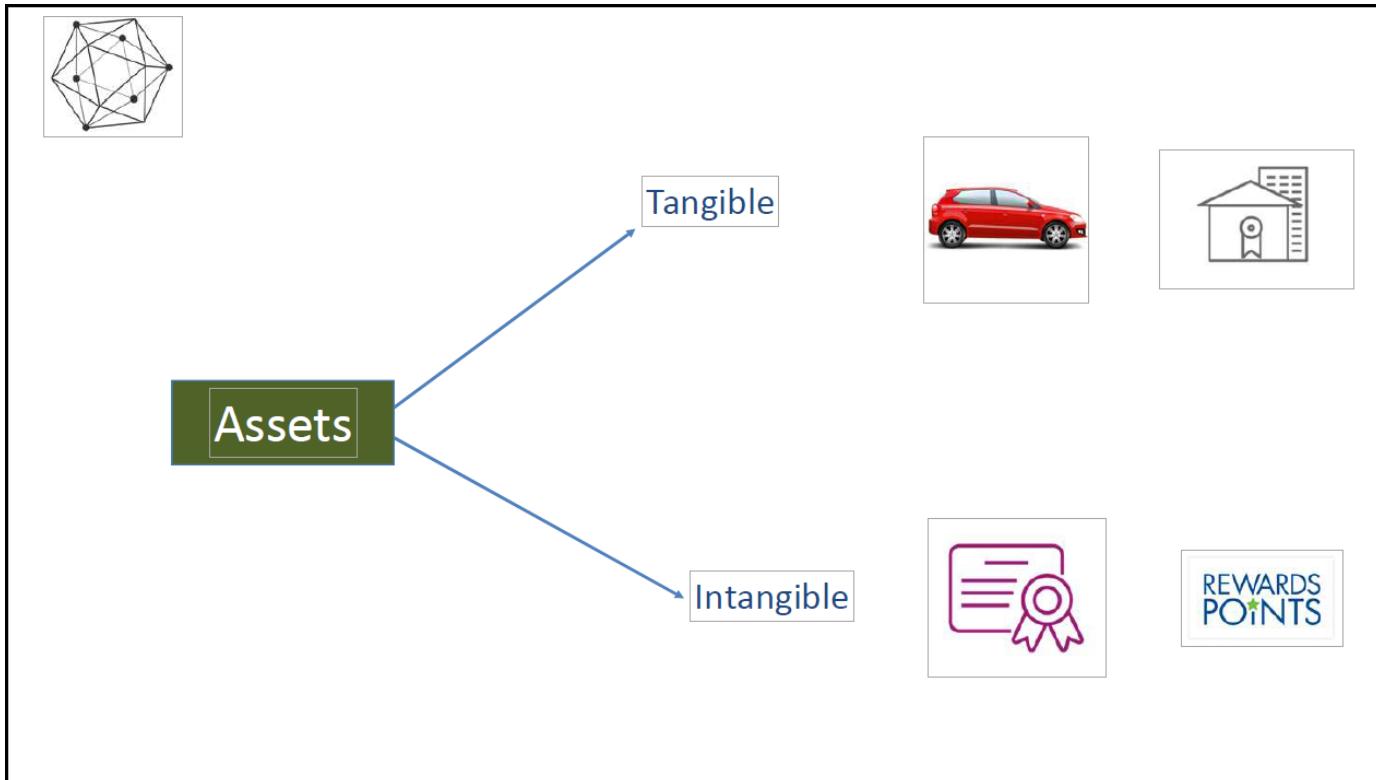
Distributed Ledger Decentralized
Group Consensus

Blockchain Defined



- ❖ In its simplest form, Blockchain is a distributed database, an unchangeable record (or Ledger) of asset ownership.
- ❖ Blockchain is primarily defined as a shared immutable ledger, or just an “unchangeable record of who owns what”
 - ❖ Global, peer to peer, and distributed immutable record of transactions.
 - ❖ Used to transfer and permanently record any change of assets between two or more parties without intermediaries.
 - ❖ Assets are defined as anything of value that requires accountability of ownership, i.e. money, cryptocurrency, real estate, records of any kind, identities, personal property, etc.

Assets Further Defined





Blockchain Uses Old Technology

- ❖ Blockchain is a combined use of existing older technology.
 - ❖ **Ledger** – 7,000 year old technology, triple-entry accounting
 - ❖ **Cryptography** – “coding messages” has been used for thousands of years, and still used in complex S/W algorithms for military and business applications like Blockchain
 - ❖ **Computer Networking Technology** – Blockchain makes extensive use of P2P networking architectures

The History of Ledgers



Sidebar - A Brief History of Accounting

- ❖ Ledgers appear around 5,000 BC
 - ❖ Single entry only
- ❖ 300 BC – Chanakya
 - ❖ First documented accounting standards
- ❖ Double-entry ledger appears in 1340 A.D.
 - ❖ Track debits and credits
 - ❖ Tell the story of a transaction from both / all sides
- ❖ Triple-entry ledger appears in 2009
 - ❖ aka Blockchain!
 - ❖ Debits, credits, and an immutable link to all past debits and credits

Blockchain Rules



- ❖ Once something has happened, and a record is created, the fact that it happened never changes
- ❖ Data written into a blockchain is a historical record and is immutable
- ❖ Blockchains have to prove they haven't been tampered with
- ❖ All the nodes (computers) running on a blockchain must agree (i.e. have consensus) on ALL the data stored on it



Blockchain is

- ❖ A record keeping system
 - ❖ To record the transfer of “tokens” or “coins” representing wealth (Monetary/currency)
 - ❖ Bitcoin and other cryptocurrencies such as Ether, LiteCoin, Monero, etc.





Blockchain is

- ❖ A record keeping system (for any kind of data)
 - ❖ To record the transactions of importance (Non-Monetary)
 - ❖ Update to a medical record
 - ❖ Transfer of ownership
 - ❖ Training certification on Blockchain
 - ❖ Recording important single-party announcements

Transferring Assets, Building Value



- Anything that is capable of being owned or controlled to produce **value**, is an **asset**
- Two fundamental types of asset
 - Tangible, e.g. a house
 - Intangible e.g. a mortgage
- Intangible assets subdivide
 - Financial, e.g. bond
 - Intellectual e.g. patents
 - Digital e.g. music
- Cash is also an asset
 - Has property of anonymity



15

Blockchain is:



Blockchain is:

- ❖ An event tracking system
 - ❖ Announcements mark events
 - ❖ Events are actionable
 - ❖ Smart Contracts running on the Blockchain allow actions to be taken on events/transactions
 - ❖ Blockchain is a workflow platform
 - ❖ Now being implemented in financial contracts, manufacturing supply chains, quality tracking, shipping and international transactions, international currency exchange/transfers.
- Unlimited Possibilities!



Blockchain is Immutable

- ❖ Cannot change the data once it's committed to the ledger
- ❖ Data is auditable
- ❖ Change by issuing offsetting transaction
- ❖ Smart contract code

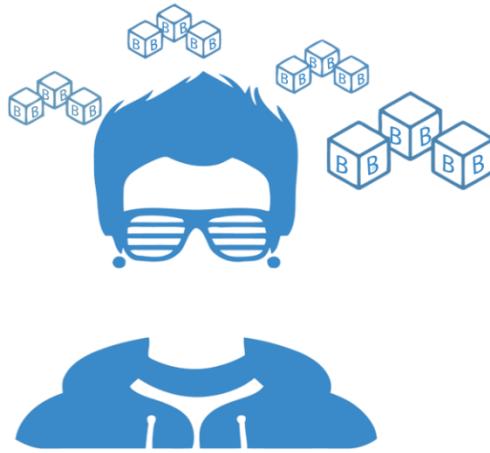


Consensus in Distributed Networks

- ❖ In order to update the ledger, the network needs to come to consensus using an algorithm
- ❖ Consensus: what does it mean to come to consensus on a distributed network?
- ❖ Consensus methodologies will be discussed a little later



Blockchain Beginnings



In 2008 “Satoshi Nakamoto” (who is a real person), published a whitepaper which outlined the architecture of Bitcoin

Interesting Blockchain Dates



- ❖ 2009 first block created
- ❖ Satoshi disappears December 2010 - date of last post
- ❖ 2015 – Ethereum and Hyperledger both go live
- ❖ 2018 – 14 open jobs for every blockchain developer
- ❖ 2019 – Walmart requires produce suppliers to be using blockchain
- ❖ 2021 – Dubai hosts all gov't operations and record-keeping on blockchain



Blockchain and Cryptocurrency

Blockchain's Relationship to Bitcoin



Bitcoin is a **digital, decentralized, disintermediated, trustless** currency



Digital Currency

Bitcoins are completely digital in nature and operates like any independent currency.

Decentralized

Bitcoins are open source peer to peer money with data stored on multiple 'nodes' simultaneously

No Intermediary

Bitcoin enables participants to transact between themselves without the need of an intermediary

Trust-less

Transactions are anonymous, thus ensuring a higher level of privacy

Blockchain is the underlying security software that manages and controls the WW Bitcoin Network, allowing for safe, trustless, and secure P2P transfer of Bitcoin, or any other cryptocurrency.

Bitcoin/Ether/Cryptocurrencies have some similar characteristics as a fiat currency



- ❖ **Durability**
 - Safe for long term storage
(crypto - susceptible to individual coin market volatility)
- ❖ **Portability**
 - Easy to move around and spend
(crypto - can be exchanged P2P with small transaction fee)
- ❖ **Divisibility**
 - So you can spend small amounts
(crypto - used in fractional amounts generally based on Bitcoin)
- ❖ **Fungibility**
 - Each unit of value is equal
(crypto - based on market value fluctuations - BTC/U.S.\$ based)
- ❖ **Scarcity**
 - To preserve value
(crypto - based on market capitalization at time of ICO)
- ❖ **Acceptability**
 - So you can actually spend it
(crypto - solely based on country monetary policy and merchant acceptability, plus transaction fees)

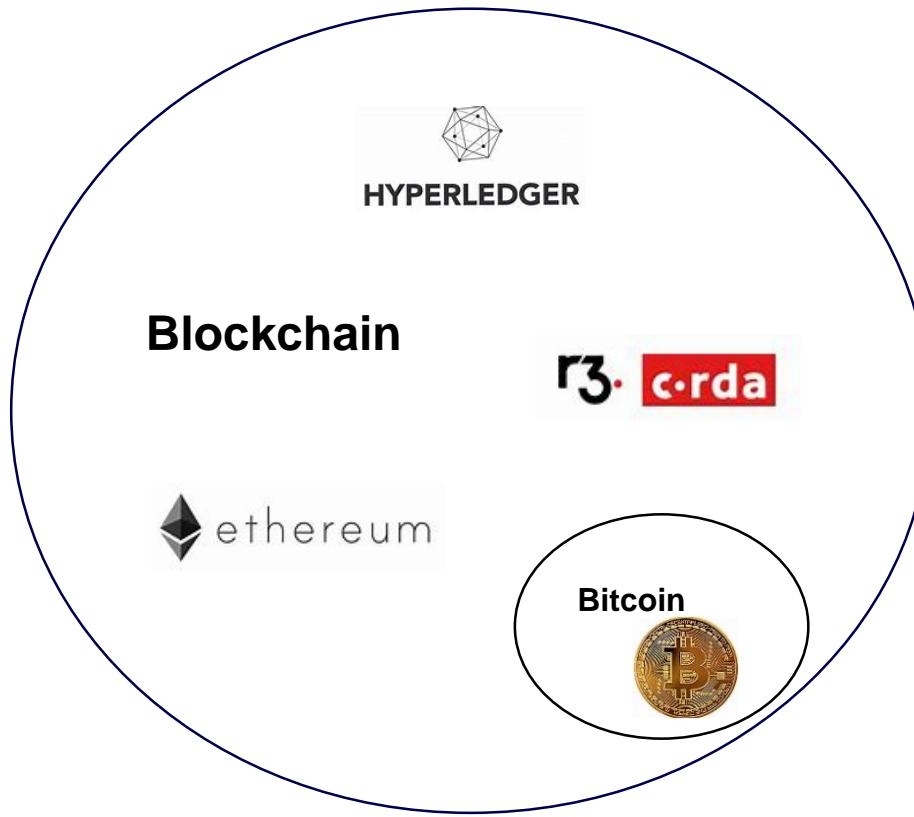


Blockchain vs. Tokens (Currency)

Bitcoin	Ethereum	Hyperledger
<ul style="list-style-type: none">• Token: Bitcoin	<ul style="list-style-type: none">• Token: Ether• EVM• ERC20 Token	<ul style="list-style-type: none">• Private permissioned blockchain (no public tokens)



Bitcoin was the first Blockchain



Bitcoin is an application that runs on the Blockchain. Blockchains can be either public or private but share the same technology.

Similar to Facebook, where Facebook is simply an application that runs on the Internet.



You Control Access to Ether

- ❖ Nobody actually ‘has’ Ether – they can’t be download, or stored on a computer
- ❖ Remember: The Blockchain is a ledger - it records the transfer of Tokens (Ether, Bitcoin,...) or other assets between people or other entities i.e. companies, groups, etc.
- ❖ The records stay on the Blockchain - what is transferred is the ‘control’ of the bitcoin or asset.
- ❖ An example would be if ‘Alice gives Bob 2btc’ i.e. ‘Alice transfers control of 2 of her bitcoins to Bob’
- ❖ Control is enabled through cryptography

Control via Cryptography



- ❖ When someone sends you coins they publicly place them under the control of your public key (in the form of an Address)
- ❖ If you can prove that you have the matching public key, and the matching private key, the network lets you control the coins



Why Use Blockchain

Decentralization



KEY CONCEPT: Decentralization

- ❖ Decentralized - Peer-to-Peer data sharing, hosting hardware owned by many not few, fault tolerant, secure, lower performance
- ❖ Distributed - Solution components spread across hardware assets, solution components and data maintained and controlled by central authorities
- ❖ Centralized - Solution components, data, and control all maintained by a central authority



Benefits of Blockchain

What are the benefits of Blockchain?

- ❖ Publicly verifiable
 - ❖ Accountability to customers and end-users
 - ❖ (permission-less)
- ❖ Secure
 - ❖ Control who sees what data when (permissioned)
- ❖ Quality assurance
 - ❖ Track origin of all supply chain components
 - ❖ Example – Food origin and/or safety recalls
 - ❖ Smart Contracts as a replacement for middlemen operators
- ❖ Lower transactions costs
 - ❖ Removing middlemen reduces cost



Benefits of Blockchain

What are the benefits of Blockchain?

- ❖ Tokenization
 - ❖ Create trade-able tokens backed by real-world value
 - ❖ Fractional ownership
 - ❖ Example - Own 1 car in 1 city, or 100 cars in 100 cities
- ❖ Trade, commerce, and business process automation
 - ❖ Smart Contracts as a replacement for middlemen operators

Drawbacks of Blockchain



What are the drawbacks of Blockchain?

- ❖ Very new technology
 - ❖ Constantly changing, still evolving
 - ❖ Number of trained resources still lagging
 - ❖ High cost for trained resources
- ❖ Best practices, recommended patterns still being formed
- ❖ Scalability, transaction speed / cost
- ❖ BaaS (Blockchain as a Service) now offered but expensive

Drawbacks of Blockchain

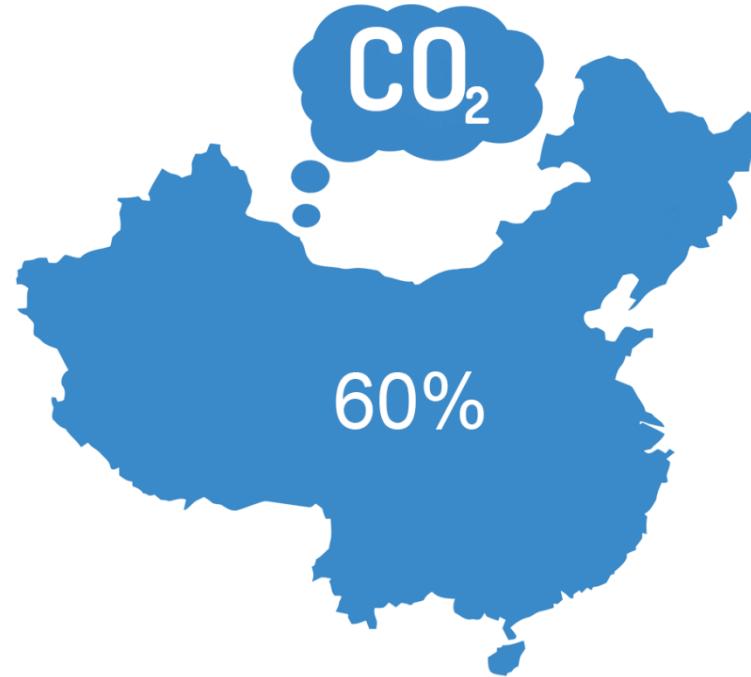
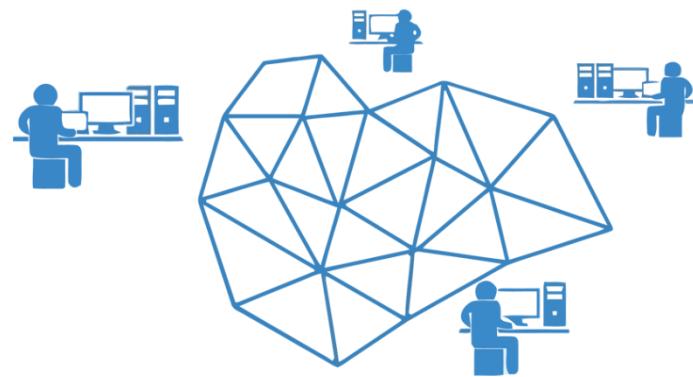


What are the drawbacks of Blockchain?

- ❖ Still learning good and bad use cases
- ❖ Lack of knowledge of the technology's capability
- ❖ Stigma and history of Blockchain
 - ❖ Association with Cryptocurrency and the dark web
 - ❖ ICO/ITO scams
- ❖ Anonymity of origin – Satoshi Nakamoto
- ❖ Data in the blocks



Drawbacks of Blockchain (PoW)



Because All global nodes must validate all transactions, speed, scalability and cumulative power consumption are critical issues facing Blockchain.

Why Not a Database



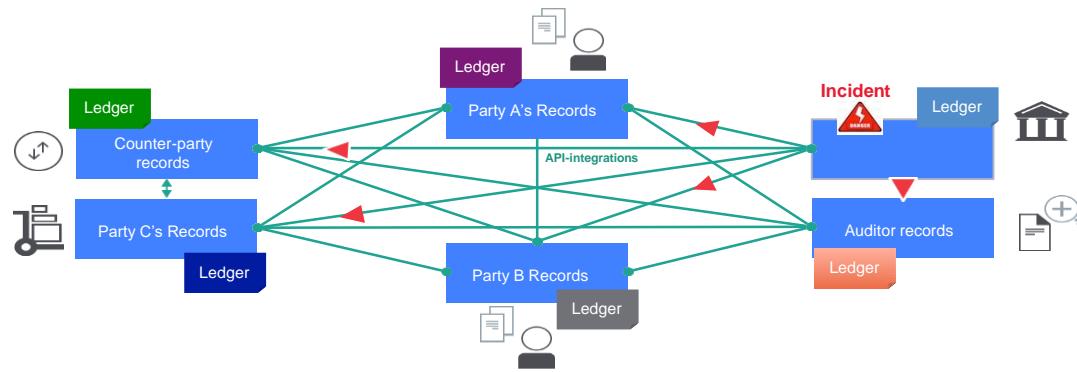
Blockchains solve specific problems:

- ❖ Fully distributed - highly fault tolerant
 - ❖ No centralized authority
 - ❖ Low barrier to entry
 - ❖ Instant, global transactional capability
 - ❖ No double spending
 - ❖ Very low transaction costs
-
- ❖ Traditional Databases have centralized control and do not perform these functions.

Conventional System Problem



Problem - Difficult to monitor asset ownership and transfers in a trusted business network



Inefficient, expensive, vulnerable

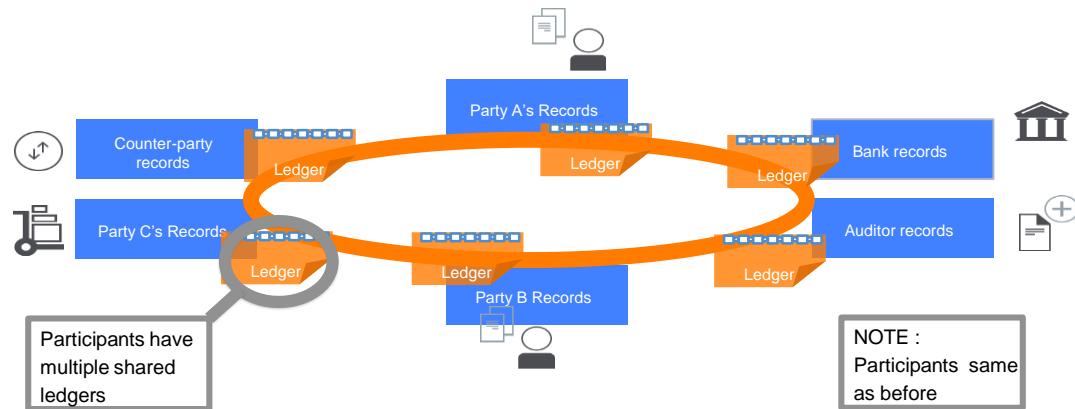
36

Blockchain Shared Ledger Solution



Solution – a permissioned, replicated, shared ledger

ALL Nodes have (and share) the same exact copy of the ledger



37

Consensus, provenance, immutability, finality



Decentralized Networks and Ledgers

Blockchains as Distributed Databases



- ❖ In Blockchain, everyone has a copy of the Ledger
 - ❖ Everyone running a Blockchain node is part of the network
 - ❖ New transactions are broadcast to and recorded by the network
 - ❖ Everyone updates their local copy of the blockchain
 - ❖ If everyone has a copy of the blockchain, when queried, everyone gets the same answer

Centralized vs Decentralized Ledger

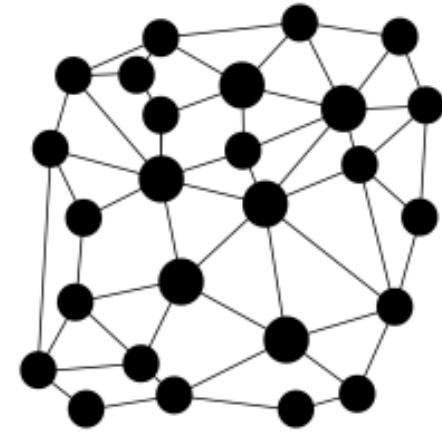


- ❖ If the single copy of the ledger were changed by any means, wealth would be lost unfairly
- ❖ With a decentralized ledger, nobody has to trust anyone else
 - ❖ Trustless environment is assumed from the beginning

Why Decentralized?



- ❖ Distributed network
 - ❖ Many nodes or peers that are connected in a network with no single point of failure or centralized control
 - ❖ Security and resiliency: design the network so that if some peers crash or attack the network maliciously, the network can still operate (Byzantine Fault Tolerance)



A distributed network.



Blockchain 2.0 and Ethereum



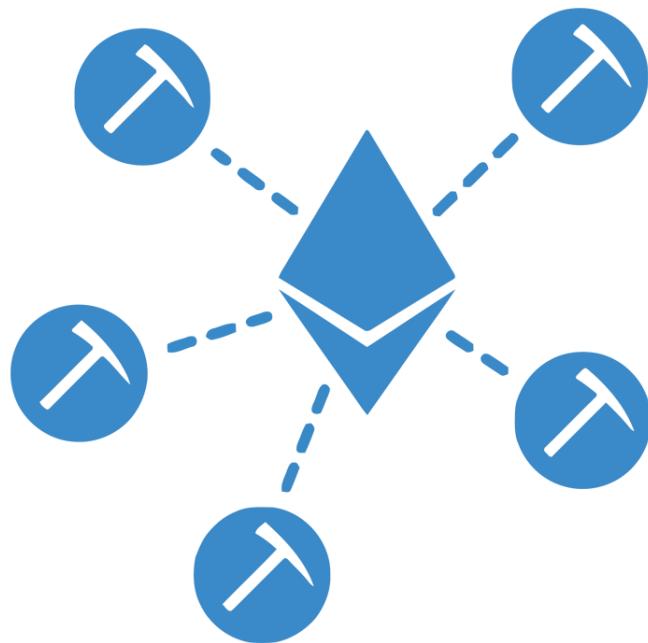
Ethereum Overview

Ethereum?



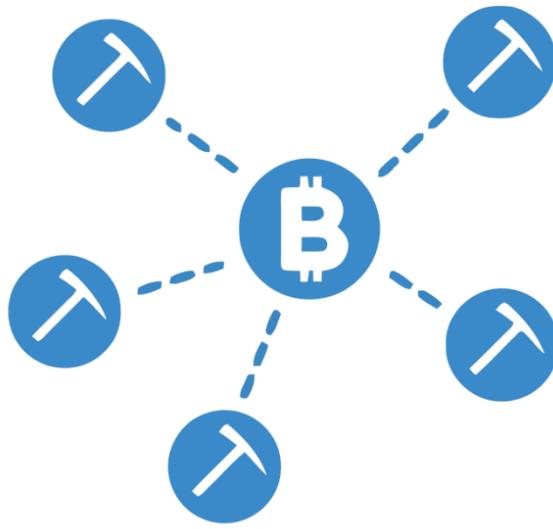
- Open source public Blockchain network
- Value token = Ether
- De-centralized Turing-complete Virtual Machine
- Smart contracts platform
- Execution requires payment - gas

Blockchain - Ethereum

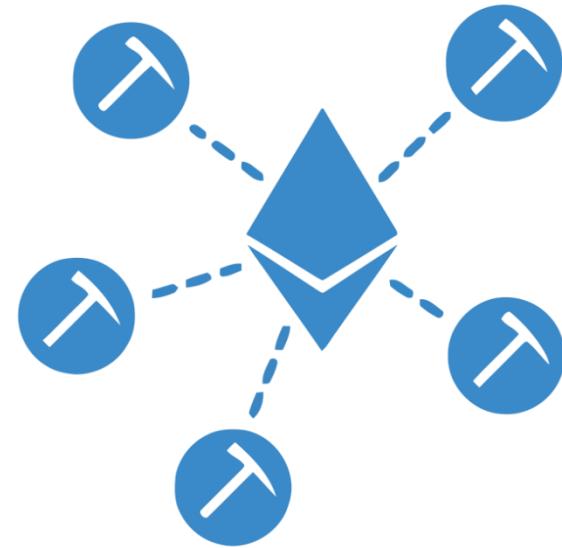


It provides a decentralized Turing-complete virtual machine, which can execute computer programs using a global network of nodes

Blockchain – Current Transactions per Second



TPS = 7



TPS = 15

Visa can process over 70,000 TPS, Facebook can handle 175,000 TPS



Types of Blockchain Transactions

- ❖ Three Types of “Transactions”
 - ❖ Two or more parties exchange of monetary value
 - ❖ Crypto
 - ❖ Remove the idea of monetary exchange
 - ❖ Two or more parties. No exchange of monetary value
 - ❖ Update medical records, Notary services
 - ❖ Remove idea of two or more parties
 - ❖ One party announcing something(immutable)
 - ❖ Supply chain Management, Business Process Automation



Ethereum Smart Contract

Smart Contract

Computer code, written in multiple languages

- Contract lives on the network
- Enforces rules
- Performs negotiated actions





ETH Valuations

Unit	Wei Value	Wei
wei	1 wei	1
Kwei (babbage)	1e3 wei	1,000
Mwei (lovelace)	1e6 wei	1,000,000
Gwei (shannon)	1e9 wei	1,000,000,000
microether (szabo)	1e12 wei	1,000,000,000,000
milliether (finney)	1e15 wei	1,000,000,000,000,000
ether	1e18 wei	1,000,000,000,000,000,000



ETH Supply

Ethers Supply

- **Ether creation**
 - Presale (2014): 60 Million
 - 12 Million created to fund the development
 - 5 Ethers created as reward for every block; roughly ~14 seconds
 - Sometimes 2-3 Ethers for non-winning miners (uncle rewards)
- Contract invocation – Users pay by Ethers
- Incentive for the miners



Ethereum EVM Software

EVM

- An software that can execute Ethereum Bytecode
 - Follows the EVM specifications (Ethereum protocol)
 - Runs as a process on a computer/sever



- EVM implemented in multiple languages



Ethereum Gas

Gas

- User invoking the transaction pays for the execution



Measures: kWh used



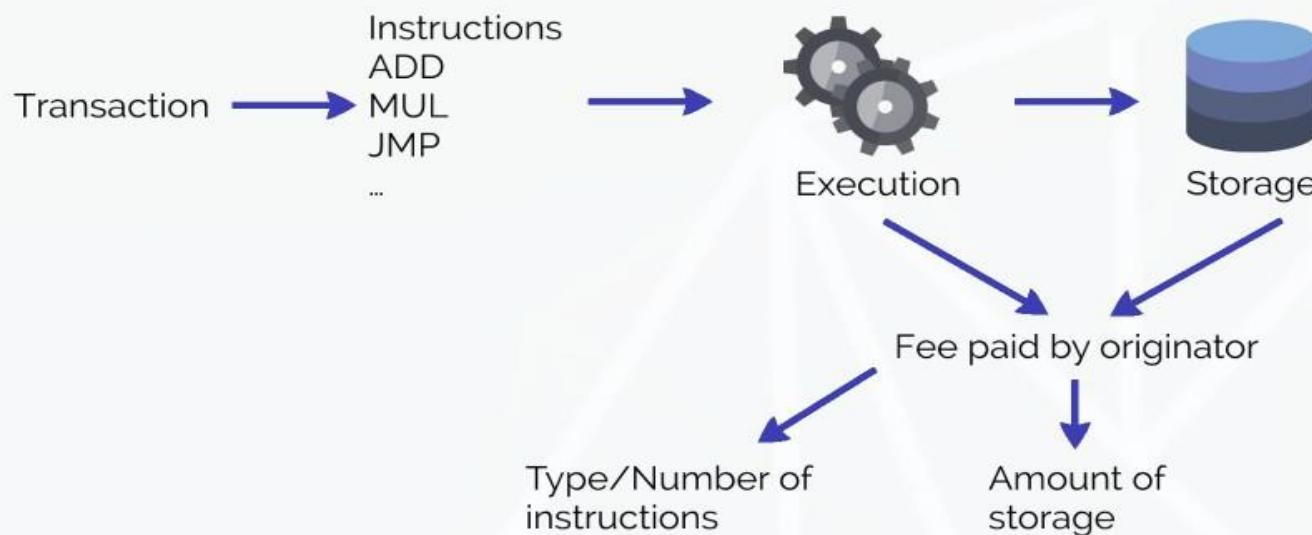
Measures: Gallons of water used

- Gas is the unit in which EVM resource usage is measured



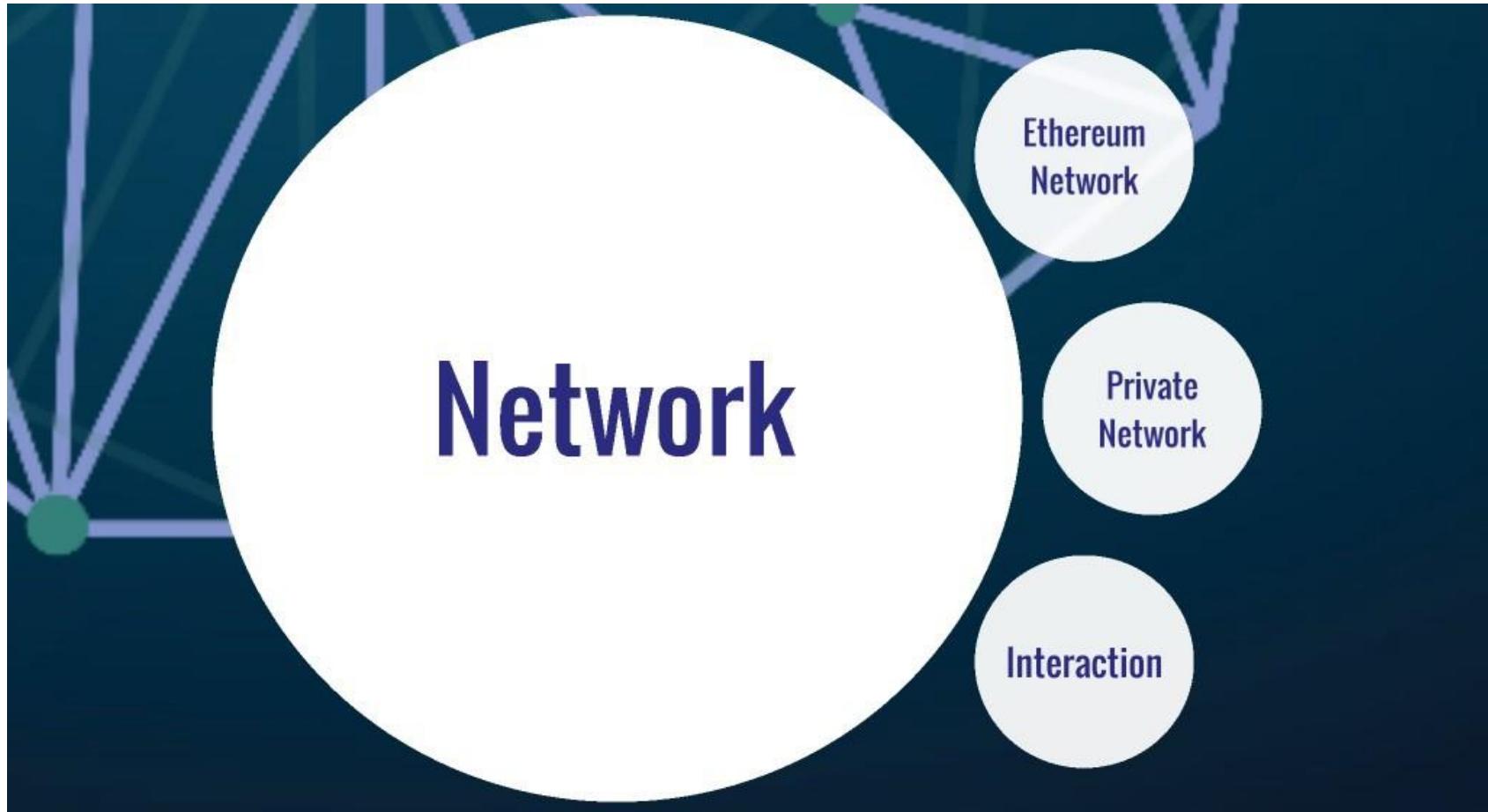
Ethereum Gas

Gas Calculation





Ethereum Network





Ethereum Network

Ethereum Network



- Network ID = 1
- Network ID = 2 Morden retired
- Network ID = 3 Ropsten current
- KOVAN RINKEBY (ID=4) current
- Network ID = Assigned



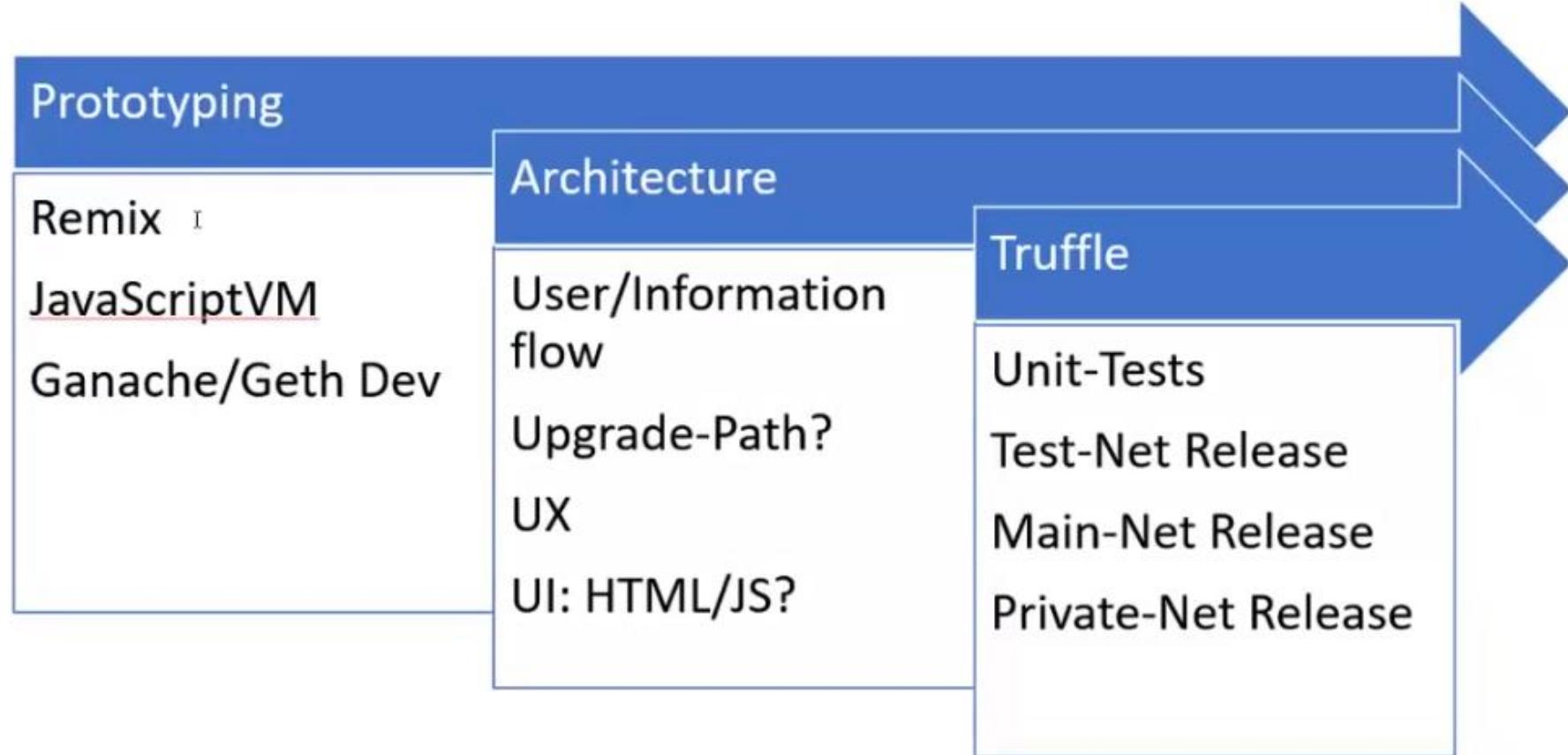
Ethereum Dev Tools

Dev Tools

- Remix IDE (Browser)
- Geth (Real Blockchain Node)
- Geth (Developer Network)
- Ganache(In Memory Blockchain Simulator)
- MetaMask(Browser-Plugin for hot wallet(client))
- Truffle(Framework for migrations and deployments of smart contracts)
- Web3.js /Truffle-Contract
 - Libraries JavaScript <-> Blockchain-Node



Dev Tools Workflow



Lab 1 Review Lab 0 setup prior to start



Programmable Blockchains

“Smart Contracts”



Smart Contracts

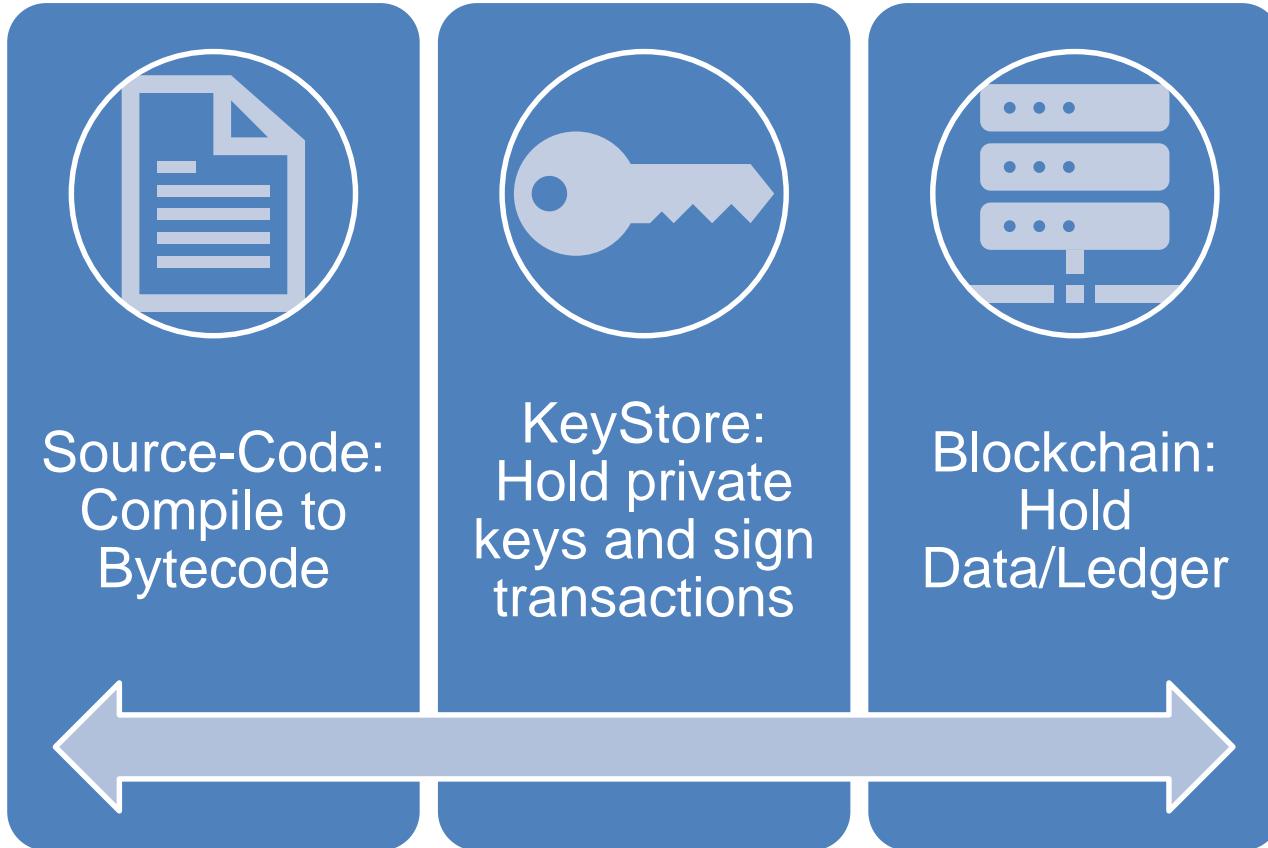
- ❖ Ethereum Blockchain
 - ❖ Solidity/Vyper/... High-Level Language
 - ❖ Compiling to EVM Assembly
 - ❖ Reside on their own address
- ❖ Actionable logic with access to the native token-layer
 - ❖ Smart Contracts can send (and own) Ether
 - ❖ “Self managing funds”



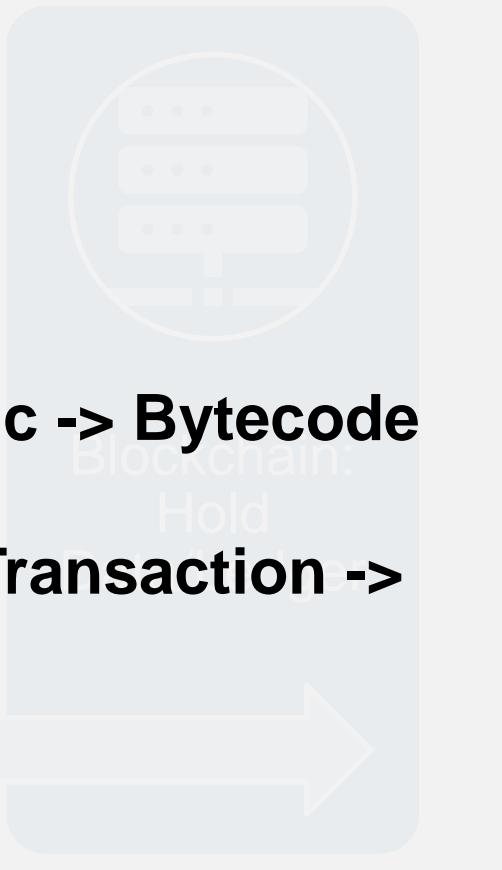
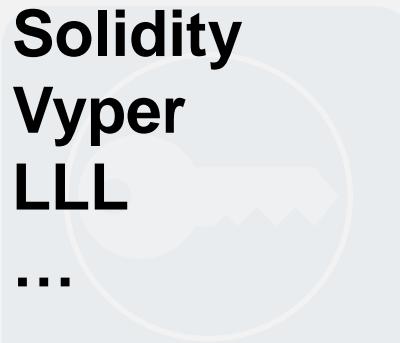
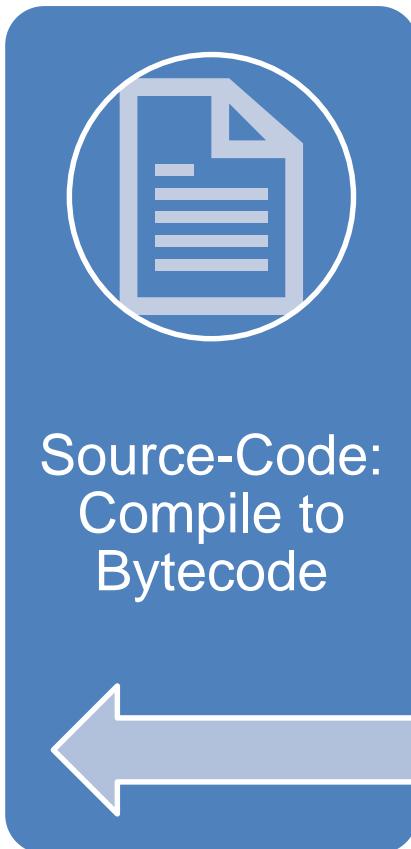
Solidity

- ❖ Solidity
 - ❖ High Level Language
 - ❖ Statically typed
 - ❖ Needs compilation (solc, solc-js)
 - ❖ Very easy to learn
 - ❖ Very hard to *master* (bugs, Gas-consumption)
 - ❖ Very “young” language, constantly evolving (to the better)
- ❖ Alternatives
 - ❖ Vyper: Security related
 - ❖ LLL: Lisp like

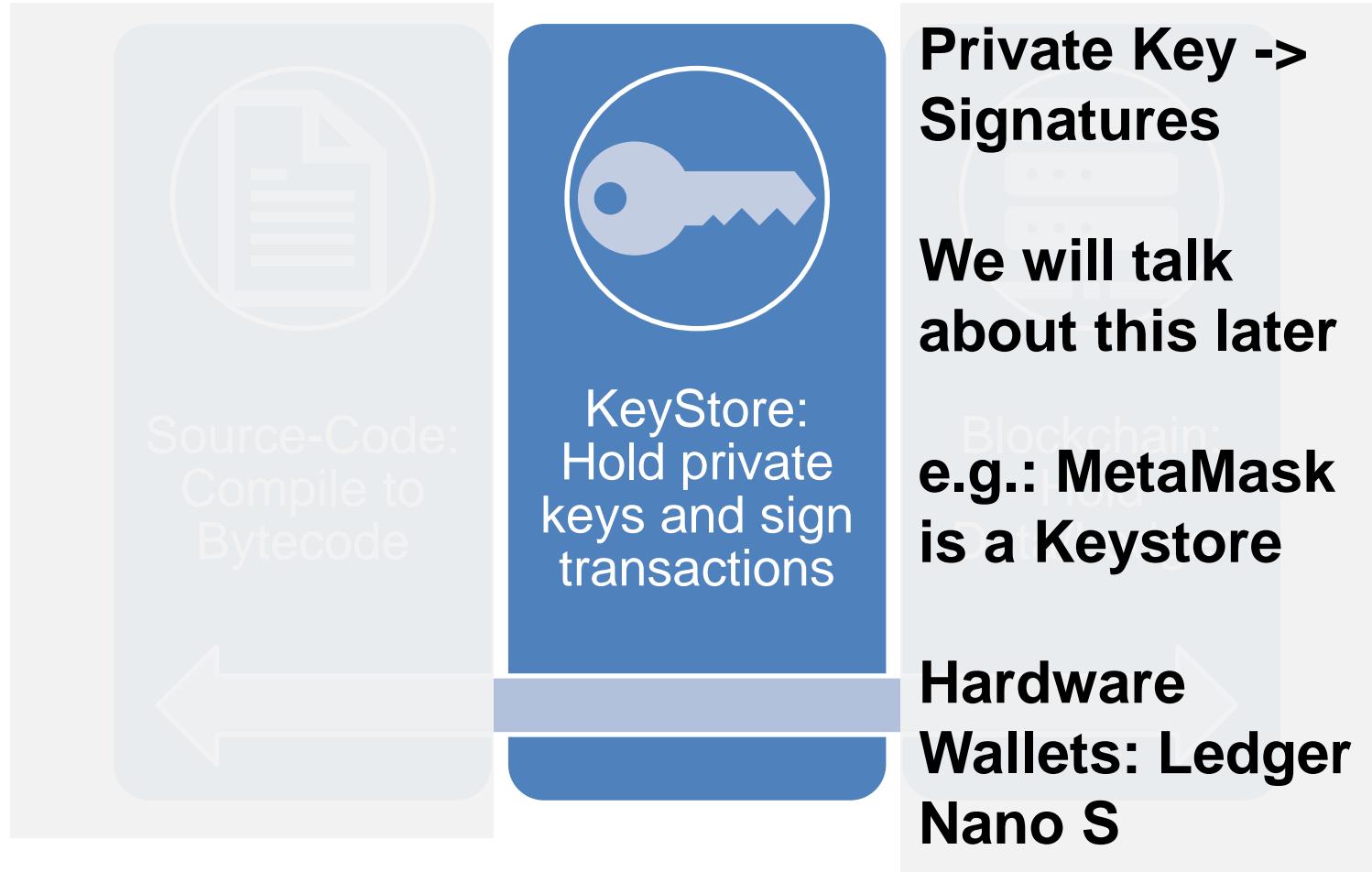
Components



Components



Components





Components

Blockchain holds the data and takes care of consensus. We'll talk about it later!

**Proof of Work: Main-Net,
Ropsten, Geth Private Net**

**Proof-of-Authority: Ganache,
Truffle-Developer-Console,
Geth-Dev Network**

In-Browser-Simulation? Remix





Remix IDE

- ❖ Browser-Based
 - ❖ Pure client-side HTML/JavaScript
 - ❖ Also a NPM package
- ❖ Editor
 - ❖ Can edit Solidity & Vyper files (Text-Editor)
 - ❖ Blockchain Simulation
 - ❖ Can run transactions and test code against an internal Blockchain simulation in the browser
 - ❖ Including a Step-By-Step Transaction Debugger
- ❖ KeyStore
 - ❖ Gives you a few accounts (private keys) for the internal Blockchain

Remix IDE



Remix
IDE

Remix - Solidity IDE

Not secure | remix.ethereum.org/#optimize=false&version=soljson-v0.4.25+commit.59dbf8f1.js

Compile Run Analysis Testing Debugger Settings Support

browser/ballot.sol

```
pragma solidity ^0.4.0;
contract Ballot {
    struct Voter {
        uint weight;
        bool voted;
        uint8 vote;
        address delegate;
    }
    struct Proposal {
        uint voteCount;
    }
    address chairperson;
    mapping(address => Voter) voters;
    Proposal[] proposals;
    /// Create a new ballot with {_numProposals} different
    function Ballot(uint8 _numProposals) public {
        chairperson = msg.sender;
        //t.c.chairperson.weight = 1;
        //c.c.length = _numProposals;
    }
    /// Give $toVoter the right to vote on this ballot
    /// May only be called by $(chairperson),
    function giveRightToVote(address voter) public {
        voters[voter].weight = 1;
    }
}
```

Current version:0.4.25+commit.59dbf8f1.Emscripten clang

Select new compiler version

Auto compile Enable Optimization Hide warnings

Start to compile

Ballot

Details ABI Bytecode

Static Analysis raised 2 warning(s) that requires your attention. Click here to show the warning(s).

browser/ballot.sol:19:5: Warning: Defining constructors as functions was
function Ballot(uint8 _numProposals) public {
^ (Relevant source part starts here and spans across multiple lines)

remix.debugHelp(): Display help message for debugging

- Welcome to Remix v0.7.3 -
You can use this terminal for:

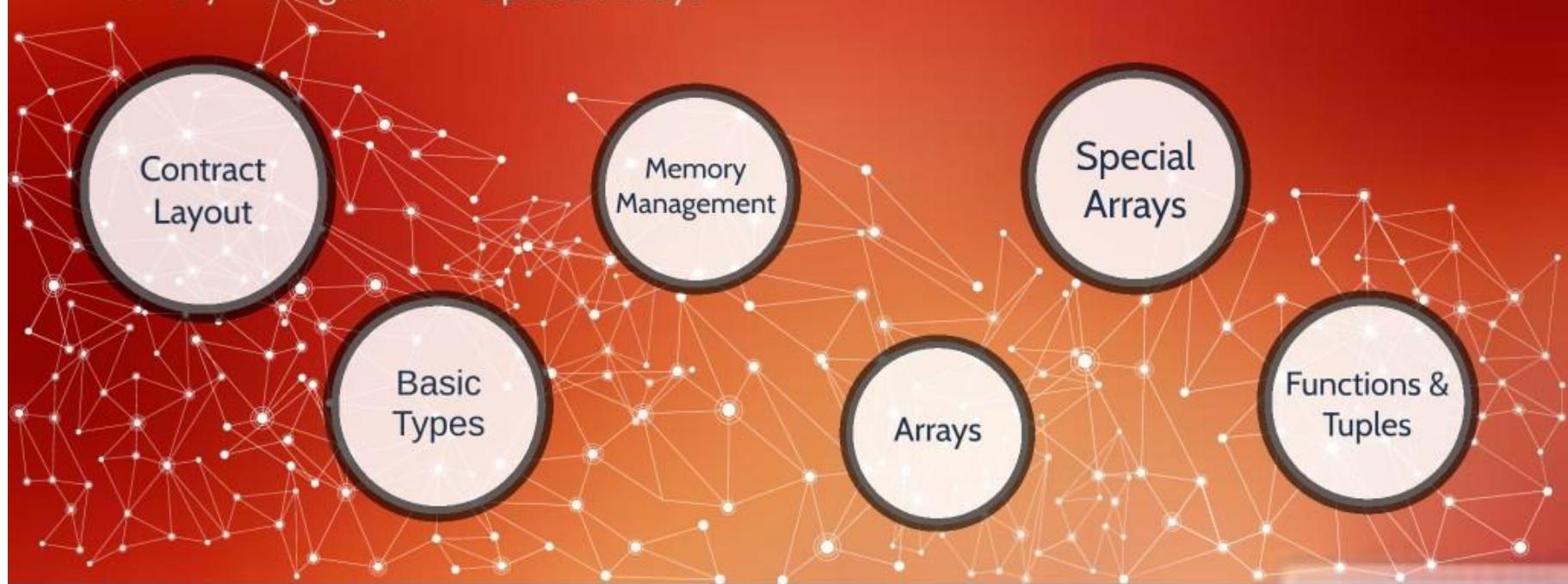
- Checking transactions details and start debugging.
- Running JavaScript scripts. The following libraries are accessible:
 - web3 version 1.0.0
 - ethers.js
 - swarmify
- Executing common command to interact with the Remix interface (see list of commands above). Note that these commands can also be included and run from a JavaScript script.



Solidity Contracts

Solidity Contracts :

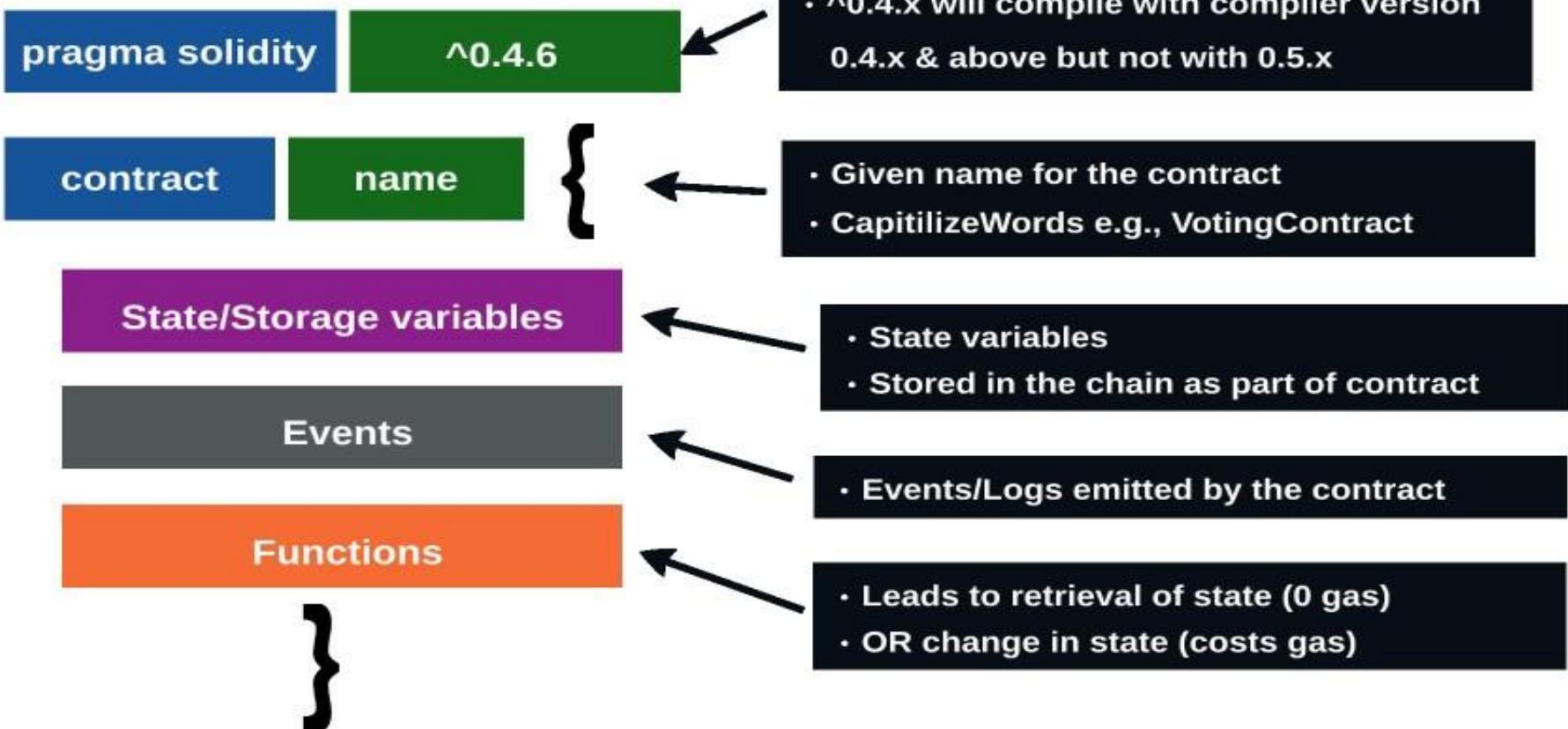
- Contract Layout
- Basic Types
- Memory Management
- Arrays
- Special Arrays
- Special Arrays





Contract Layout

Contract Layout





Boolean & Number

Boolean

- bool
- true / false
- ! && || == !=

Integer

- int & uint
- Size specified in 8 bit increments
- E.g., int8 int16 uint32
- Default: int = int256

```
int num1; // Signed Integer Initialized to 0
uint8 num2; // Unsigned Integer Initialized to 0
bool flag; // Initialized to false
```

Address

- Value types = Always passed by value



Address

Address

- Represents the 20 byte Ethereum address
- Value Type

balance

- `address.balance`

Returns balance in wei

transfer () send ()

- `address.transfer(10)`

Sends 10 Wei from to the `address`

Bytes and Strings



- ❖ Bytes
 - ❖ Array of bytes
 - ❖ Can hold “text”, but not UTF-8
 - ❖ Can be fixed size or dynamic size
 - ❖ Index (myBytes[7]) can be accessed
 - ❖ Can be expanded (push operation)

- ❖ String
 - ❖ Not a “basic” type
 - ❖ Arbitrary Length UTF-8 encoded string
 - ❖ Dynamically sized
 - ❖ No String-Functions (search, replace) – need external libraries for that

Variable Initialization



- ❖ Un-Initialized Variables are automatically set to their default value
 - ❖ integers => 0
 - ❖ Boolean => false
 - ❖ Strings => “” (empty)
 - ❖ ...
 - ❖ **NO ERROR WHEN ACCESSING UNINITIALIZED VARIABLES**
- ❖ There is no “null”



Functions

Functions

```
contract Funcs {  
  
    string ownerName;  
    uint8 ownerAge;  
  
    // Sets the name  
    function setOwnerInfo(string name, uint8 age){  
        ownerName = name;  
        ownerAge = age;  
    }  
  
    // Get the name  
    function getOwnerName() returns (string) {  
        return ownerName;  
    }  
  
    // Get the age  
    function getOwnerAge() returns(uint8 age){  
        // age = ownerAge;  
        return ownerAge;  
    }  
}
```



Output Parameters

Output Parameters

- Use keyword `returns(...)`
- Multiple return parameters
- You may name the return parameters
 - Named local variable available within the function body
 - Initialized to zeros
 - Values assigned to named variable are automatically returned



Output Parameters

Output Parameters

- Declare the arguments with type/names

```
function setData(bytes name, uint8 age){  
    // code for the function  
}
```

- ***But*** may omit argument name if unused

```
function setData(bytes name, uint8 ){  
    // code for the function  
}
```



Local Variables

Local Variables

- Re-declaration of the variable in the function not allowed

```
function someComplexCalculation(uint principle, uint rate) returns(uint){  
  
    for(uint i=0; i < array.length; i++){  
        // do something  
    }  
  
    uint i = 6;  
  
    // Do something  
    return 0;  
}
```

// Compiler will throw an error
// Variable 'i' already declared



Variables Initialization

Variables Initialization

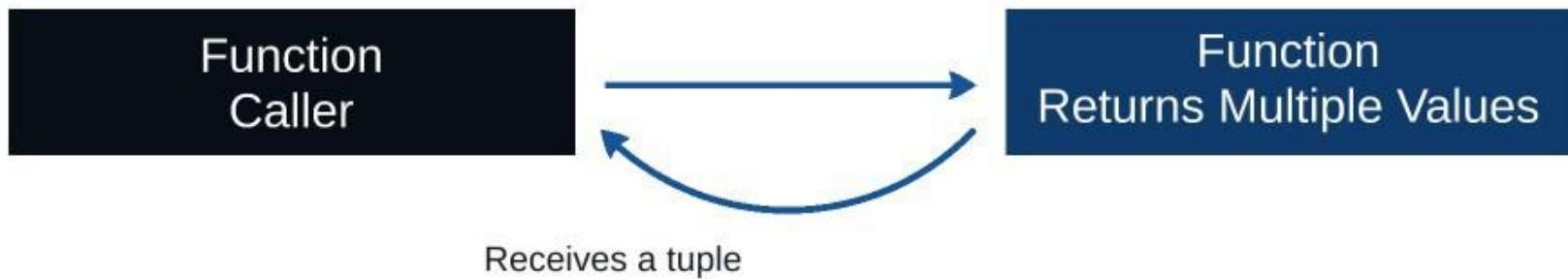
- Bytes initialized to **0s**
- Bool to **false**
- Variables initialized to defaults in the beginning of the function

```
function varScope() returns (uint){  
    uint i = 5;  
  
    uint j = i + k;  
  
    uint k = 10;  
  
    return j;  
}
```





Tuple Types





Tuple Types

Tuple types

- A tuple is a list of objects

```
var(name, age) = getOwnerInfo();
```

- Different types in tuple are OK
- You may **skip** a variable in tuple

```
function multiReturnCaller() returns (string n,uint8 a){  
    // Create a tuple  
    var(name, ) = getOwnerInfo();  
}
```



Type Conversion

Type Conversion

Implicit

- Compiler allows if no loss of information
- If (1) { /** code **/}

Explicit

- Potential loss of information

`uint32 x32 = 20;` `uint24 x24 = x32;`

`uint24 x24 = uint 24(x32);`

Deduction

- Compiler can automatically infer type

~~`var someVar = x32;`~~

Deprecated



Data Location

Data Location

- Default: State variables
- Default: Local variable
- Function(args)

Persistence (it's a database)
• Key-Value Store (256 bit key & value)
• Read/write are costly
• Contract can manage only its own



Temporary
• Arrays & structs
• Addressable at byte level

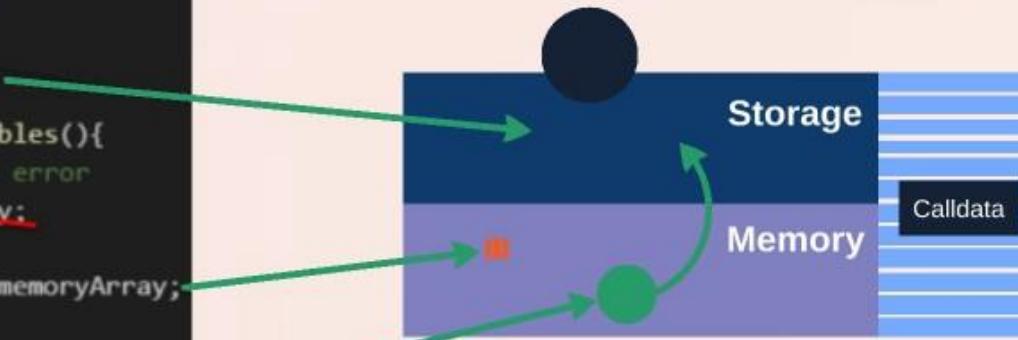
Temporary
• EVM code execution
• Non-modifiable
• Max size 1024, Word 256 bit

Local & Storage Variables



Local & Storage Variables

```
contract DataLocation {  
  
    // Always in storage  
    uint      count;  
    uint[]   allPoints;  
  
    function localVariables(){  
        // This will give error  
        uint[] localArray;  
  
        uint[] memory   memoryArray;  
  
        // Creates a reference  
        uint[] pointer = allPoints;  
  
    }  
}
```

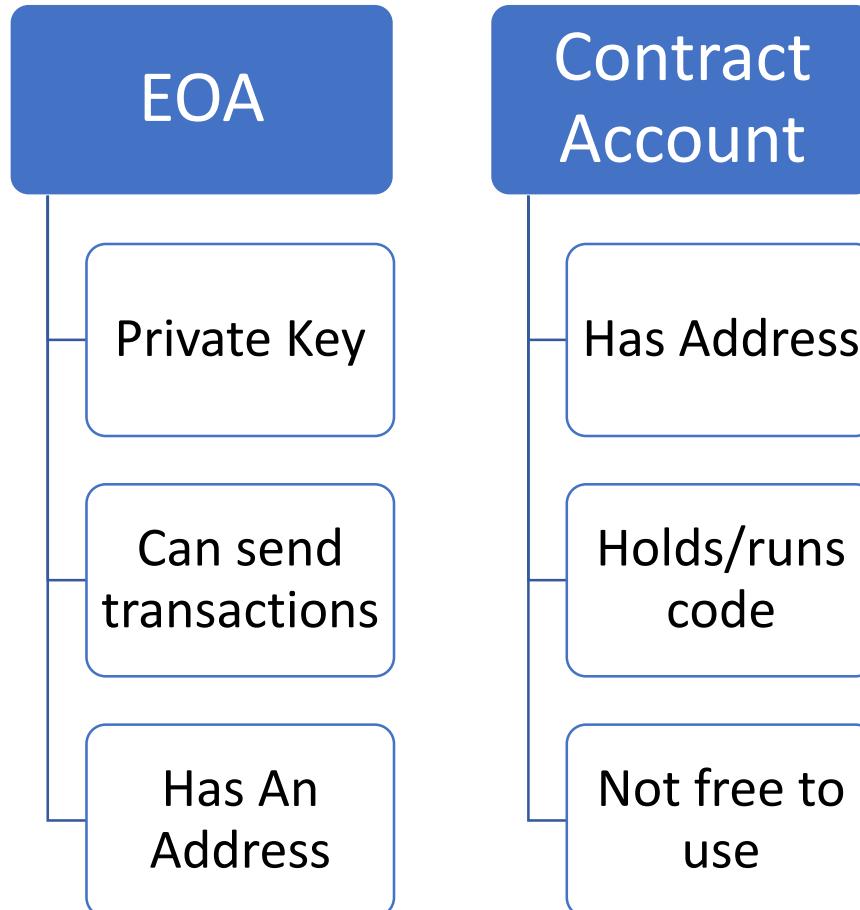


Types of Accounts

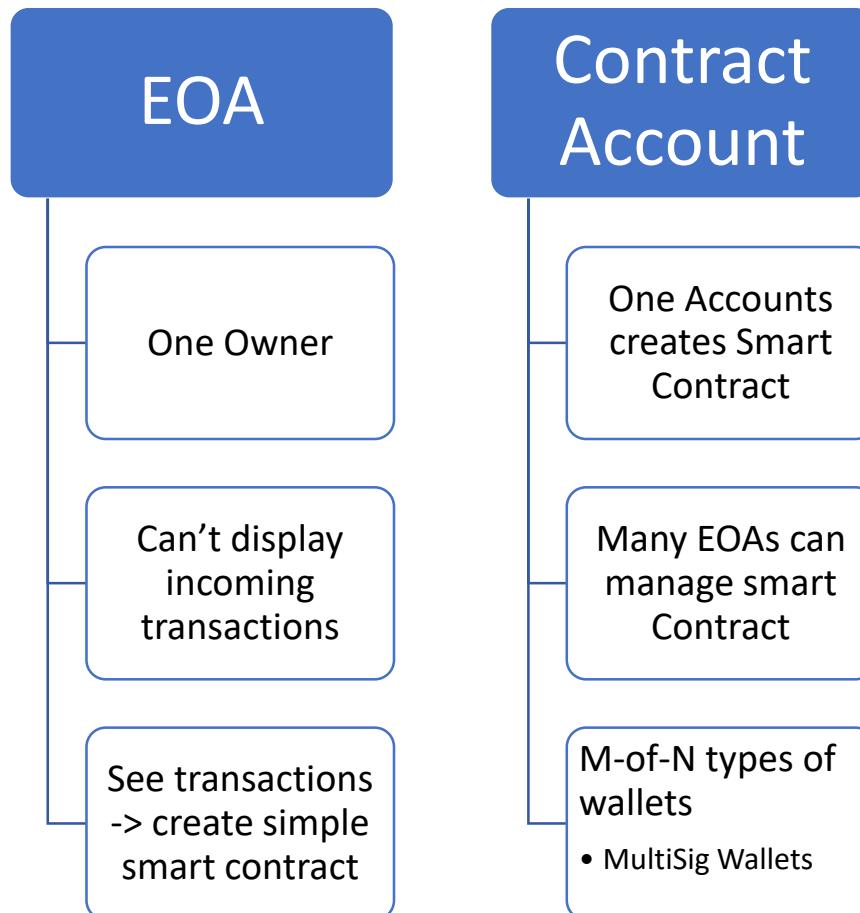


- ❖ Two types of Accounts in Ethereum
 - ❖ Externally Owned Accounts
 - ❖ Contract Accounts
- ❖ Externally Owned Accounts (EOA) are managed by a private Key
 - ❖ Send Transactions
 - ❖ Deploy Smart Contracts
- ❖ Smart Contract Deployment will lead to a new Address for the Smart Contract
 - ❖ Deterministic (based on “from” and “nonce” field in Transaction)
 - ❖ Smart Contract manages itself based on the code – no private key!

Types of Accounts



Types of Accounts



MultiSig Contract



MultiSig Contract





Types of Blockchain

Blockchain as History



- ❖ Blockchain as History
 - ❖ Immutable, cannot be changed
 - ❖ Remember, each block contains the hash of the previous block
 - ❖ Append-only
 - ❖ Data on the Blockchain cannot be deleted or edited, only additions can be made
 - ❖ This provides a detailed history of ALL events, not just a snapshot of the current state!



Types of Blockchains

- ❖ Public vs Private
 - ❖ Who can **write** data to the Blockchain?
 - ❖ Public – everyone can add a record
 - ❖ Private – only certain participants can write data
- ❖ Open vs Closed
 - ❖ Who can **read** data from the Blockchain?
 - ❖ Open – everyone can read Blockchain data
 - ❖ Closed – only certain participants can read data



Public or Private Blockchain

- ❖ Should the solution be a permissioned or permission-less Blockchain
 - ❖ Are all participants considered equal, or should some have abilities that others do not?
 - ❖ Election chairperson can add candidates to an election = permissioned
 - ❖ A digital currency which can exchanged and traded by all = permissionless
- ❖ Do customers understand the tech well enough to trust it with their data?
 - ❖ Great solutions may not be accepted until they have been socially normalized
 - ❖ Credit cards and early e-commerce

Public or Private Blockchain



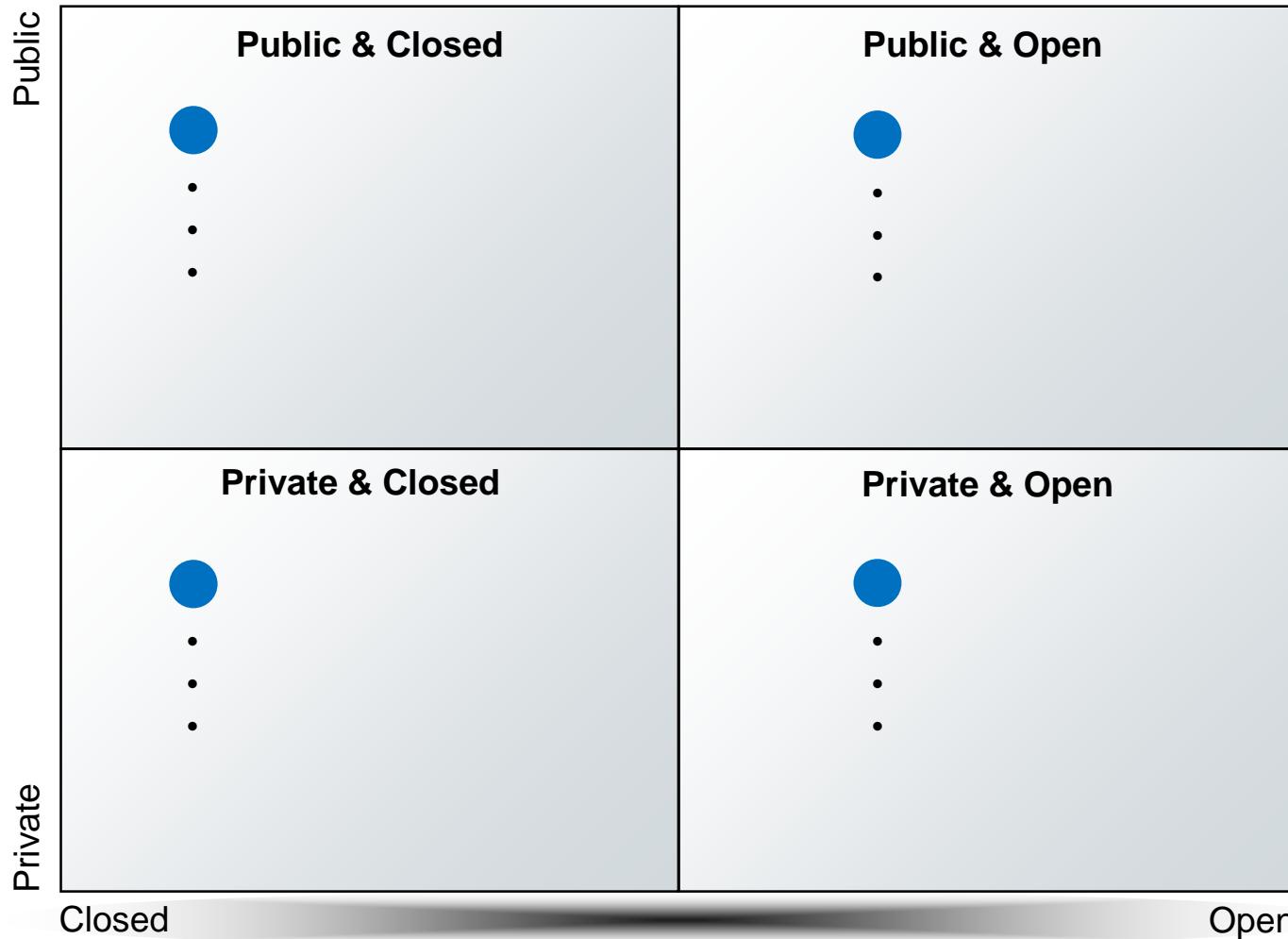
- ❖ Hyperledger vs Ethereum
 - ❖ These are discussion points, NOT absolutes
- ❖ Ethereum
 - ❖ Music and content distribution
 - ❖ Digital currency or asset-backed token
 - ❖ Blockchain-enabled mobile data service
 - ❖ Gambling and on-line gaming
 - ❖ Authoring, editing, and amending a piece of legislation
 - ❖ Group consensus is needed/required



Public or Private Blockchain

- ❖ Hyperledger vs Ethereum
 - ❖ These are discussion points, NOT absolutes
- ❖ Hyperledger
 - ❖ Supply Chain
 - ❖ Supplier / Manufacturer inventory management
 - ❖ Managing internal business processes across geographically distributed locations
 - ❖ Allowing elected officials to vote on initiatives without being present

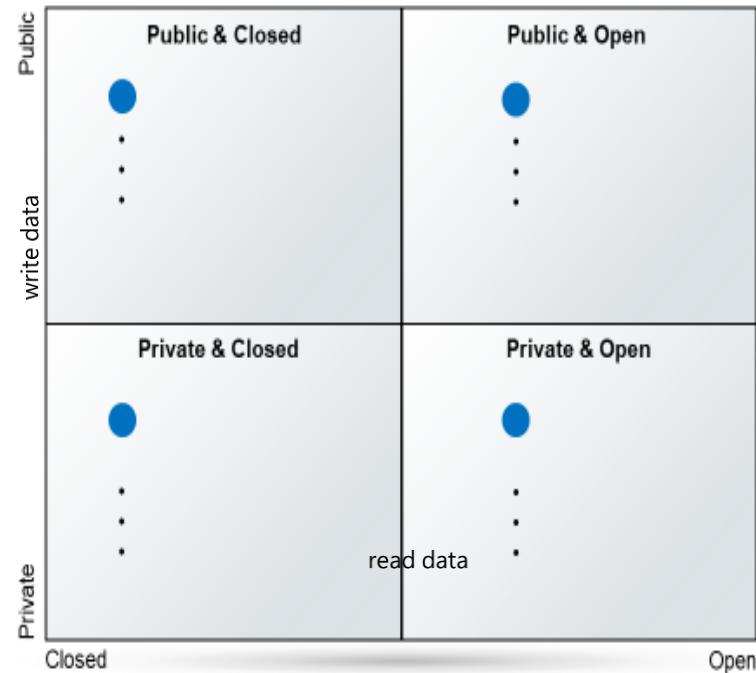
Blockchain Decision Worksheet



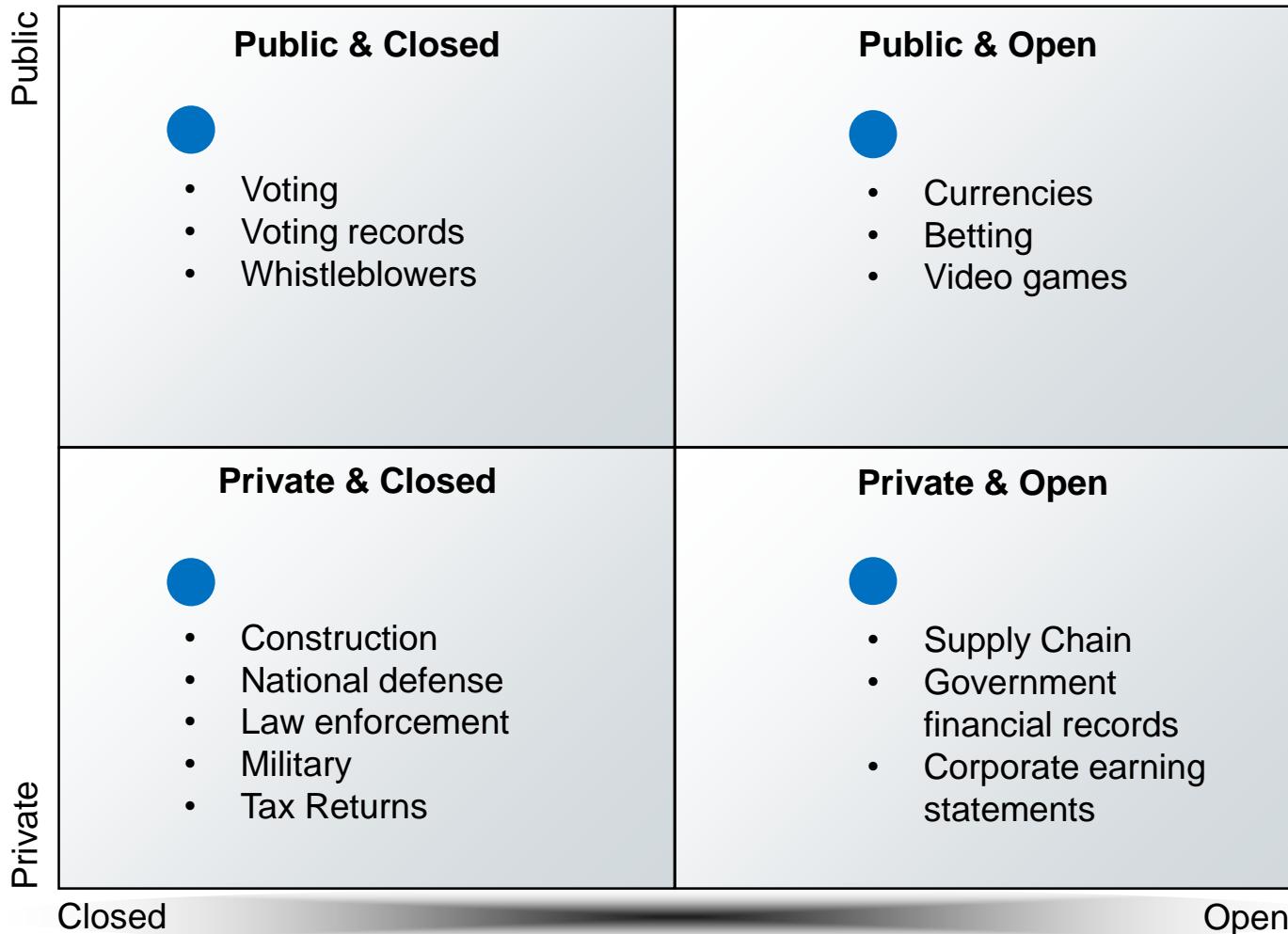


Blockchain Decision Lab

- ❖ Currency
- ❖ Securities exchange
- ❖ Betting
- ❖ Video game
- ❖ Voting records
- ❖ Supply chain data
- ❖ Government financial records
- ❖ Corporate earnings statements
- ❖ Construction tracking
- ❖ Defense programs
- ❖ Law enforcement agencies
- ❖ Others?



Blockchain Decision Matrix



Public Blockchains

- What is blockchain? (public, open, permissionless)
 - Decentralized ledger
 - Can store any type of data
 - Shared ledger
 - Immutable
 - Anonymous
 - Fully-Transparent
 - Group Consensus
 - Nodes only verify data was recorded correctly
 - No ability to verify truth of the data itself
 - Smart Contracts
 - Ability to automate processes
 - Blockchain as workflow / BPM

Blockchain for Business

- Are these properties good or bad?
 - Decentralized ledger
 - Can store any type of data
 - Shared ledger
 - Immutable
 - Anonymous
 - Fully-Transparent
 - Group Consensus
 - Nodes only verify data was recorded correctly
 - No ability to verify truth of the data itself
 - Smart Contracts
 - Ability to automate processes
 - Blockchain as workflow / BPM



TEST Networks

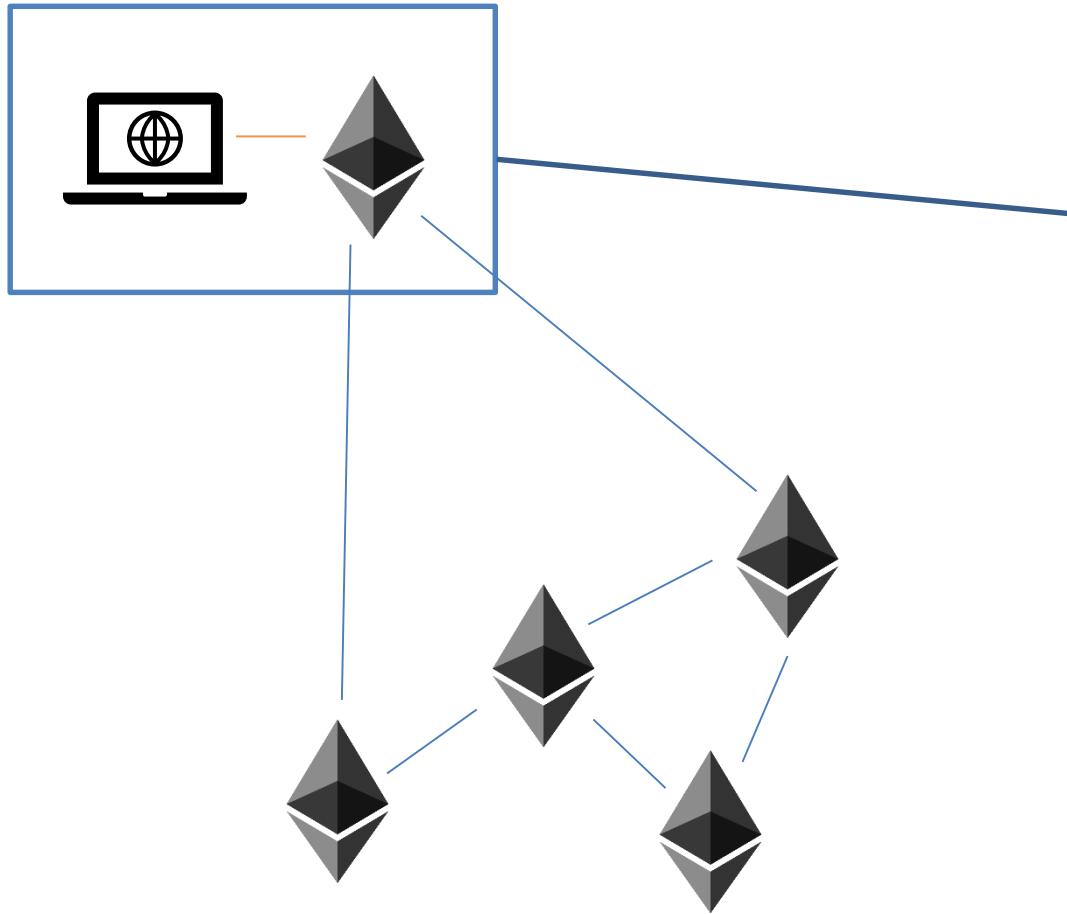
- ❖ What is a Test-Net? (public, open, permissionless)
 - ❖ Same Protocol as a “main” net
 - ❖ Different Data
 - ❖ 1 Ether = 0\$, but not worthless
 - ❖ Used for testing
 - ❖ Dapps
 - ❖ “Give it a try”
 - ❖ Get comfortable
 - ❖ Beta-Customers
- ❖ Ethers on a Test-Net
 - ❖ Faucets



Ethereum Blockchain Tooling Infrastructure



Infrastructure: Local Node



Blockchain Node
Deployed locally *on the client side.* (*Geth, Parity*)

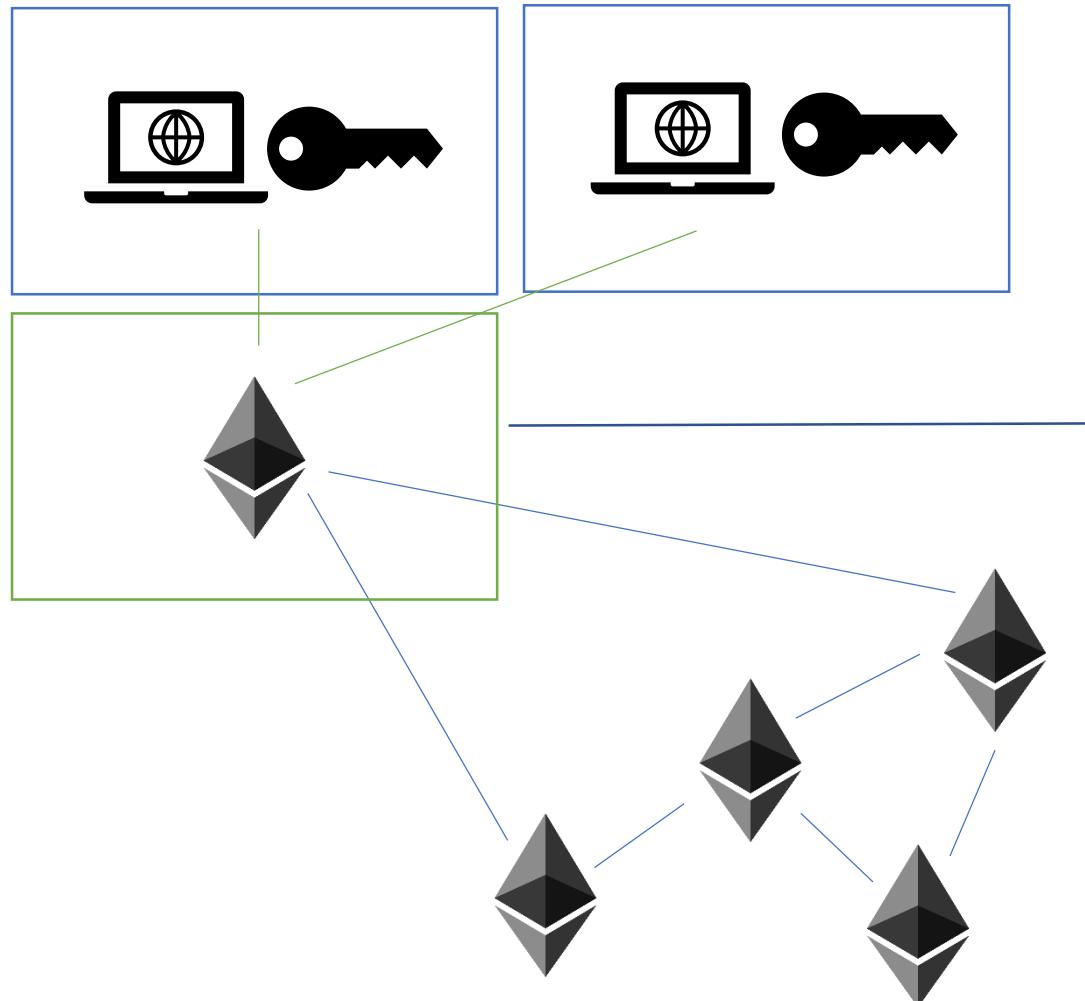
Node needs to “sync” in all blocks from the Network and keep syncing. Can take a long time, but secure.

Node is connected to the rest of the network.

Node can also contain the KeyStore, keys stored locally



Infrastructure: Organization Hosted



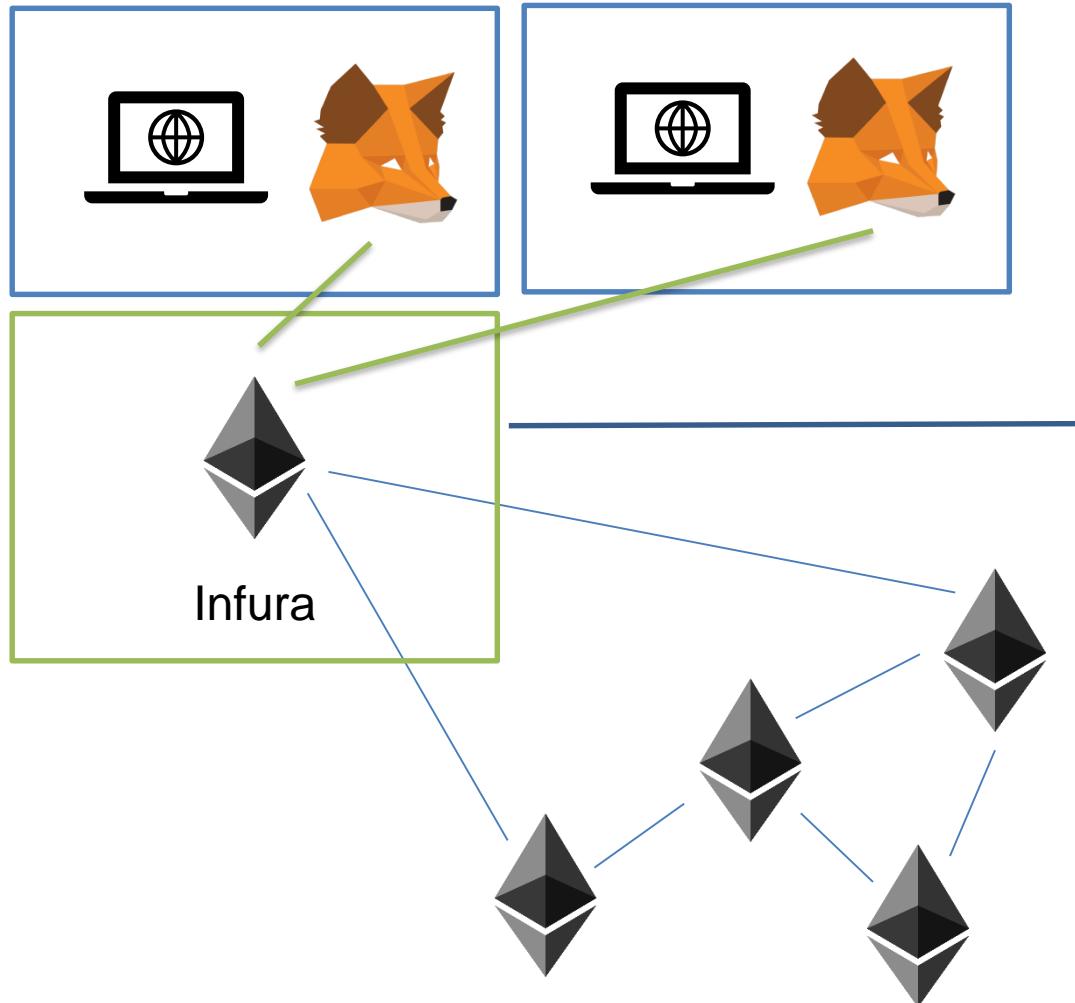
Blockchain Node Deployed
on Organization-Server.
(e.g. Geth, Parity)

Several Clients connect to
one Server directly

Node is connected to the
rest of the network.

Who stores the Private
Keys?

Infrastructure: Externally Hosted



Blockchain Node
Deployed on Infura-
Server.

Browser connects to
MetaMask, MetaMask to
Infura.

Private Keys stored in
MetaMask. Connection
to Dapps very easy!



DAPP Infrastructure



METAMASK



- Manage accounts in a browser vault
 - Export/Import accounts
 - Send Funds
- Exposes web3 object to browser app
 - Single Page Applications
- Supports multiple endpoints
- Does not support mining

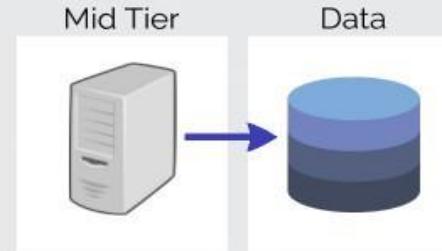


Web App vs DApp

Web App DAPP



Centralized Resources
Owned by the organization



Front end apps

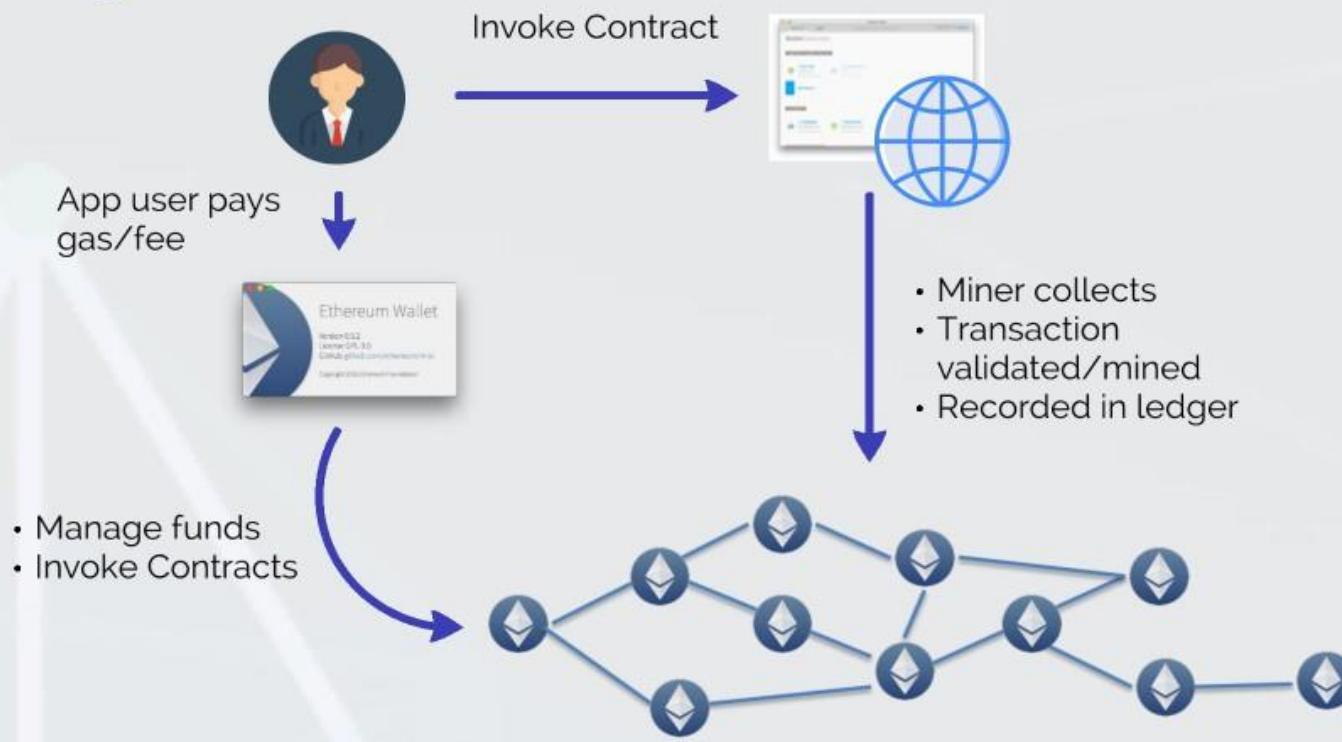


Decentralized Resources
Public domain



DAPP

Working of DAPP



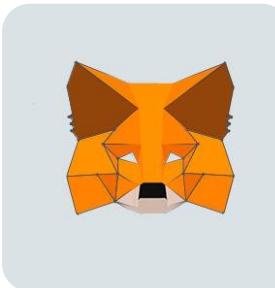


DAPP Bidding

Example DAPP Bidding



Remix



Coding in Remix

- Compilation
- Debugging
- Testing (in local JavaScript VM)



Deployment

- Using MetaMask
- To Infura
- Or Organizational Blockchain

Deployment

- Using direct connection
- Local GO-Ethereum
- JSON RPC Interface

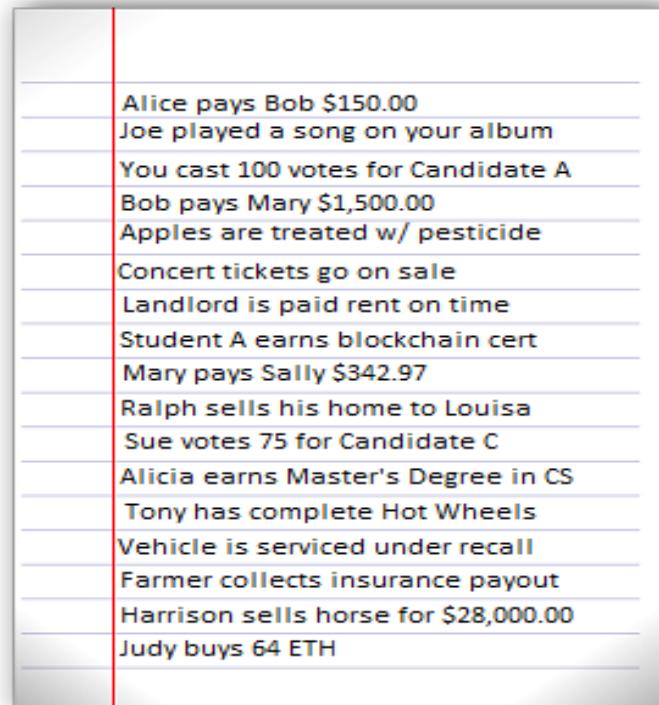


How Blocks are Created

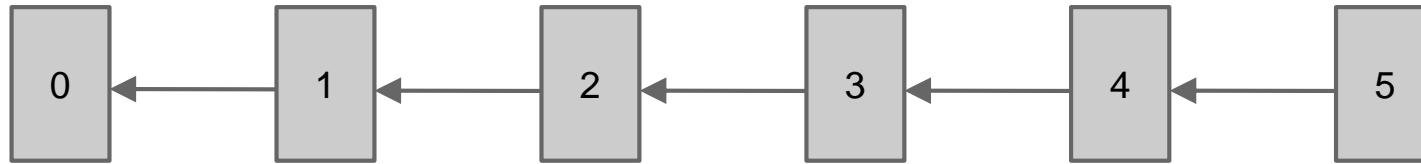


The “Blocks” of a Blockchain

- ❖ Think of a “Block” as a sheet of paper with 25 recorded transactions
 - ❖ Each of the 25 lines has a complete transaction record
 - ❖ Each record is complete with time, data, all transaction details
 - ❖ When a sheet is filled (25 transactions), the Nodes “validate” the transactions on the current page and post it on the Blockchain
 - ❖ All Nodes must agree (have Consensus) on the transaction content



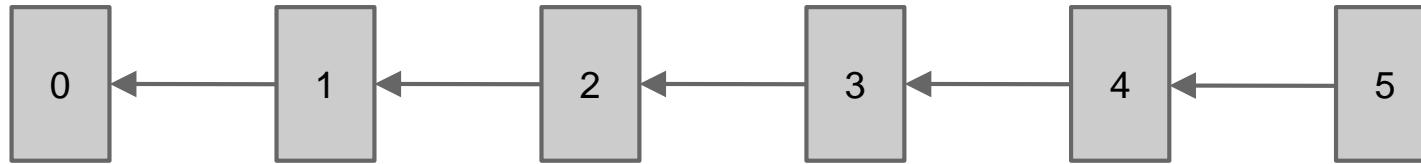
Blockchain Blocks



- ❖ Blocks are numbered in ascending order, 0 is first/oldest
- ❖ The number is the ‘height’ of the block
- ❖ Arrows only go from newer to older blocks - a block only directly links to the one immediately before it
- ❖ Once a block is stored, it’s read-only (which is why it doesn’t link to the ones after it - that would require you to update it)

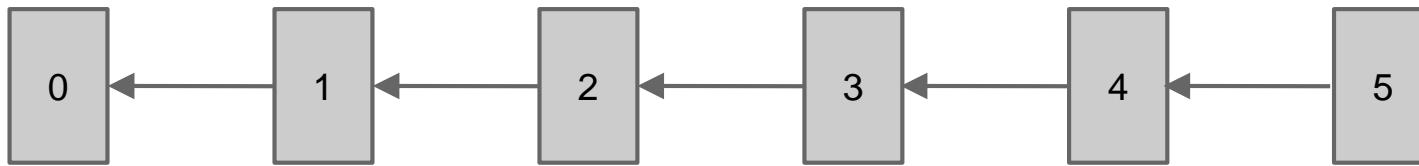


Blockchain Blocks



- ❖ Blocks store data, in Bitcoin, it's the transactions, but it could be any digital data
- ❖ Blocks are created periodically (on average, 10mins for Bitcoin) by a process called 'mining'
- ❖ A block represents a set of events that have occurred over a particular time frame (usually, since the previous block)

Blockchain Blocks



❖ Block id is the hash of the **data** in the block

- 0=000000000019D6689C085AE165831E934FF763AE46A2A6C172B3F1B60A8CE26F
- 1=00000000839A8E6886AB5951D76F411475428AFC90947EE320161BBF18EB6048
- 2=000000006A625F06636B8BB6AC7B960A8D03705D1ACE08B1A19DA3FDCC99DDBD

❖ Block id is a digital fingerprint of that block

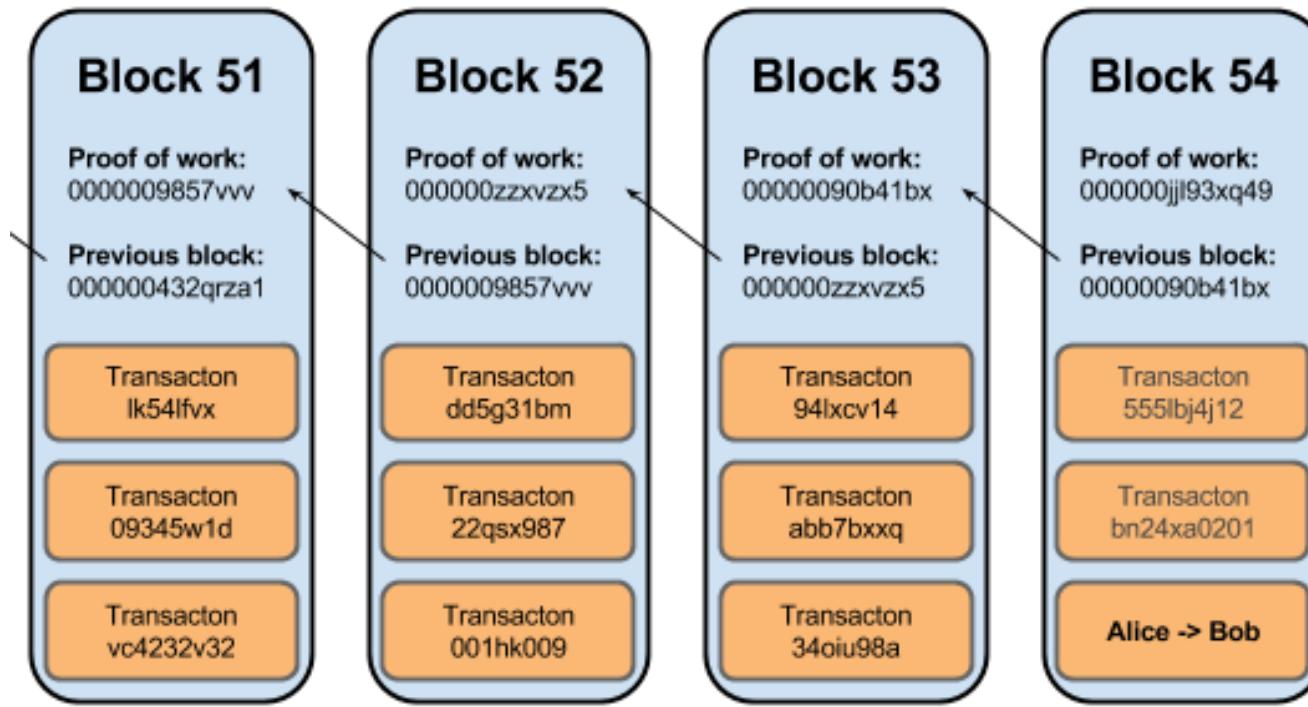
What is in a Block?



- ❖ A size number to specify how much data is contained in the block
- ❖ Some metadata:
 - ❖ A version number of the block format
 - ❖ A link to the previous block that came immediately before it
 - ❖ Merkle root of all the transactions in the block
 - ❖ Timestamp of when the block was created
 - ❖ Mining difficulty (more about this later)
 - ❖Nonce for proof-of-work (more about this later)
 - ❖ All the data of the transactions recorded in this block



What is in a Block?



The connection between blocks means that the Blockchain is much more *tamper-proof* than standard database structures. Since Blockchain is a ledger of records, this tamper-proof record of assets is known as an “Immutable Ledger”.



Cryptography and Hashing



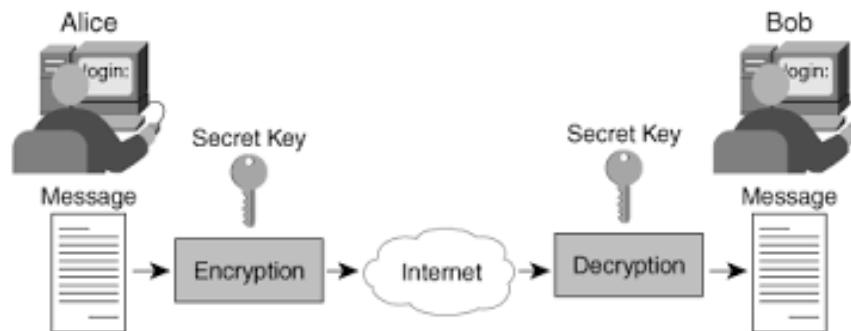
Cryptography Usage

- ❖ Cryptography to Encrypt sensible information and then decrypt it again
 - ❖ Symmetric Cryptography
 - ❖ Public-Key Cryptography (asymmetric)
- ❖ To sign information for proof of authenticity
 - ❖ Digital Signatures
- ❖ To create a non-reversible unique “fingerprint” of Data
 - ❖ Hashing



Cryptography

- ❖ Cryptography can be used to address the issue of privacy
- ❖ What is cryptography?
 - ❖ The study of how to send information back and forth securely in the presence of adversaries



Cryptographic Function



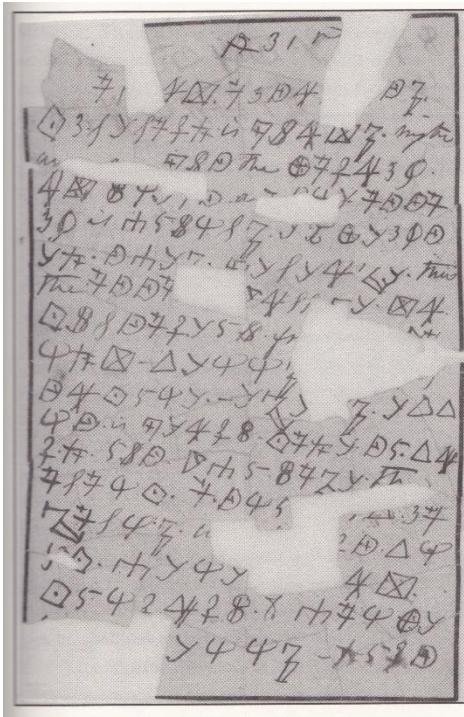
What is a cryptographic function?

- ❖ A function for encoding or encrypting data to protect the contents from adversaries
- ❖ Simple example function:
 - ❖ The Secret - “Blockchain Training Alliance”
 - ❖ The Function – Swap each letter in the secret with a new letter according to the Key
 - ❖ The Key - “+2”
 - ❖ The Cipher = “Dnqemejckp Vtckpcpi Cnnkcpeg”

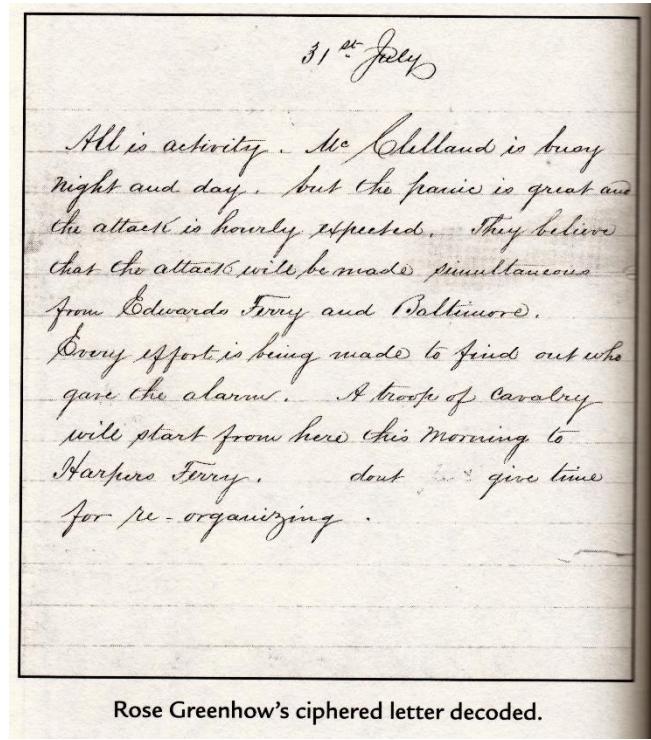


Cryptographic Function

- ❖ Real World Example: Rose Greenhow
 - ❖ Renowned confederate spy during US Civil War
 - ❖ Socialite in Washington D.C.
 - ❖ Used cryptography to communicate



Rose Greenhow's seized ciphered letter.



Rose Greenhow's ciphered letter decoded.



Digital Signatures

- ❖ “A digital signature is meant to prove a message came from a particular sender; neither can anyone impersonate the sender nor can the sender deny having sent the message.”
source: Wikipedia

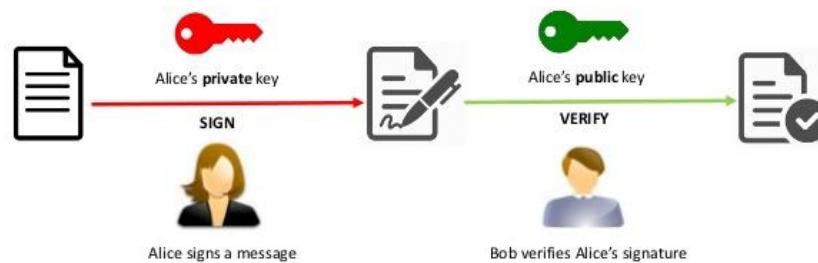


Digital Signatures

Public Key Cryptography

- ❖ Provides identity & transaction approval
- ❖ Public Key
 - ❖ Verify the digital signature of a given key pair
- ❖ Private Key
 - ❖ Sign/approve any transaction/action that might be made by the holder of the key pair

Digital Signatures

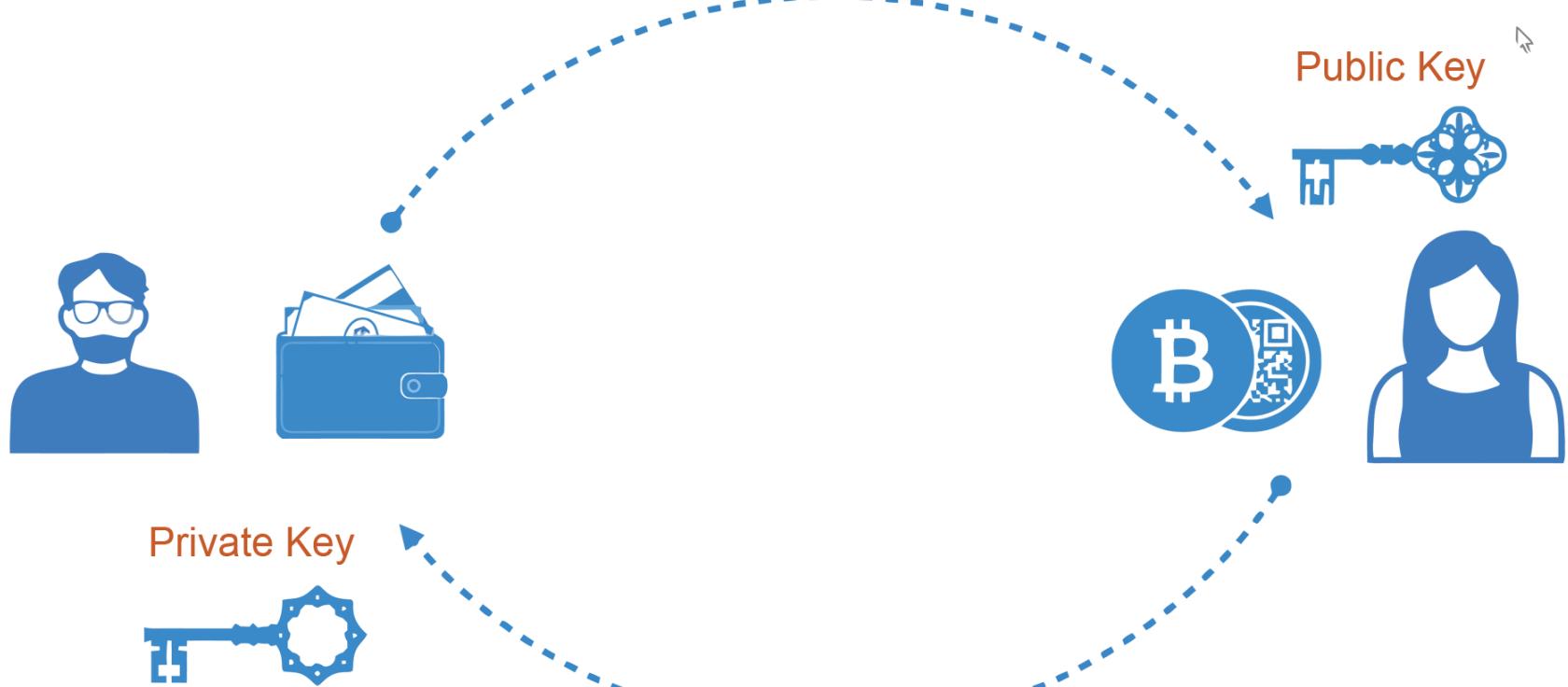


PUBLIC/PRIVATE KEY CRYPTO



- ❖ 2 uniquely related cryptographic keys
- ❖ Data encrypted with the public key can only decrypted with the private one (and vice versa)
- ❖ Main aim is confidentiality (in messaging)
- ❖ Also used for digital signatures

Public and Private Keys





Important Terms

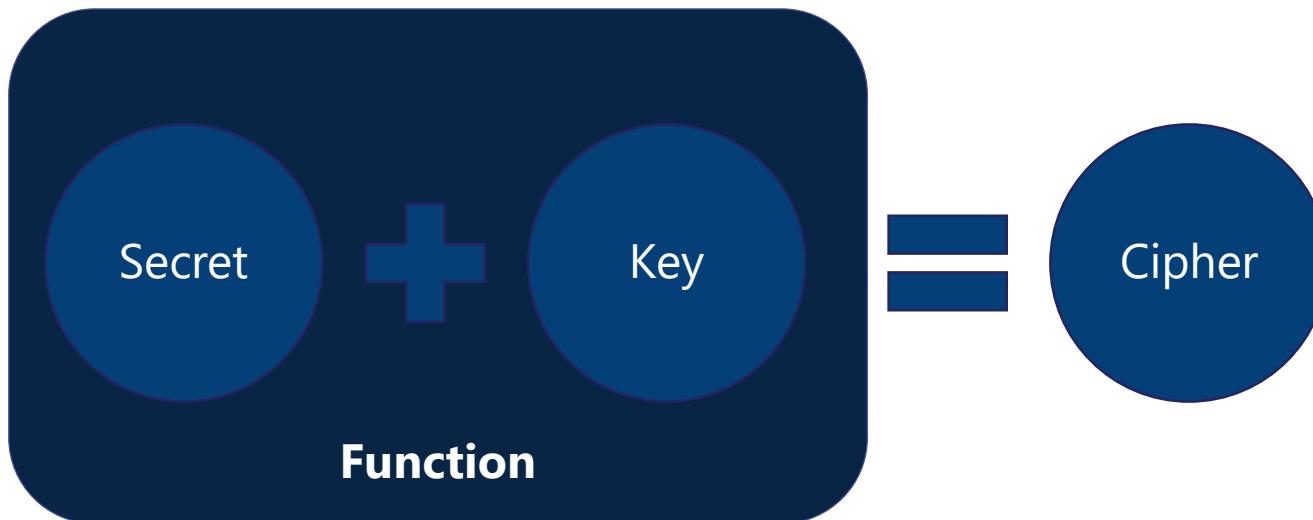
Terms:

- ❖ The Secret – The data which we are trying to protect
- ❖ The Key – A piece of data used for encrypting and decrypting the secret
- ❖ The Function – The process or function used to encrypt the secret
- ❖ The Cipher – The encrypted secret data, output of the function

Cipher



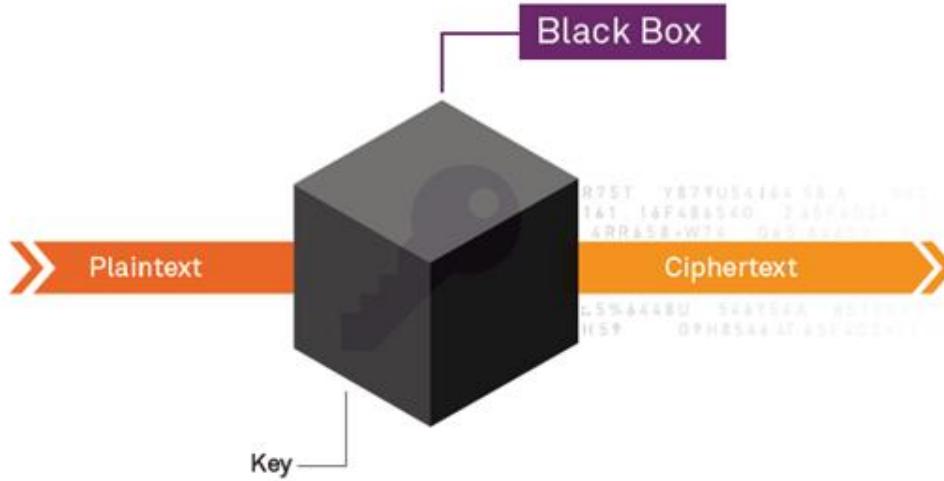
- ❖ The Secret and the Key are passed into the Function to create the Cipher





Cryptographic Hash

- ❖ What is a cryptographic hash function?
 - ❖ A hash is a one-way function, encrypted information CANNOT be decrypted
 - ❖ Each unique input generates a unique output



Cryptographic Hash



- ❖ Why would I want to use a hash?
 - ❖ Address privacy concerns by making net worth private with a “digital thumbprint”
 - ❖ This would NOT be acceptable:
 - ❖ Sally - \$418,013.45
 - ❖ John - \$93,247.89
 - ❖ Mary - \$9,423.11
 - ❖ This WOULD be acceptable:
 - ❖ 0376189a740845f75bde8260416b3812ab6d4377 - \$418,013.45
 - ❖ 5753A498f025464d72e088a9d5d6e872592d5f91 - \$93,247.89
 - ❖ 94F85995c7492eec546c321821aa4beca9a3e2b1 - \$9,423.11
 - ❖ Nobody knows Sally’s net worth, but Sally can always prove which account is hers



Cryptography Example

- ❖ Cryptographic Hashing example
 - ❖ Try it out (using SHA256)
 - ❖ [Go to: www.anders.com/blockchain/hash.html](http://www.anders.com/blockchain/hash.html)
(or: bit.ly/2HnJS6O)
- ❖ Demo:
 - ❖ Let's eat, Grandma
 - ❖ 45b09eea07b7896d836308e89d01986ea92227ea41d5239dc42650c66393cc01
 - ❖ Let's eat Grandma
 - ❖ aab9185e406446c4700be80ae8c274778a15e9f2914303ae803ecfa2eca19b5a

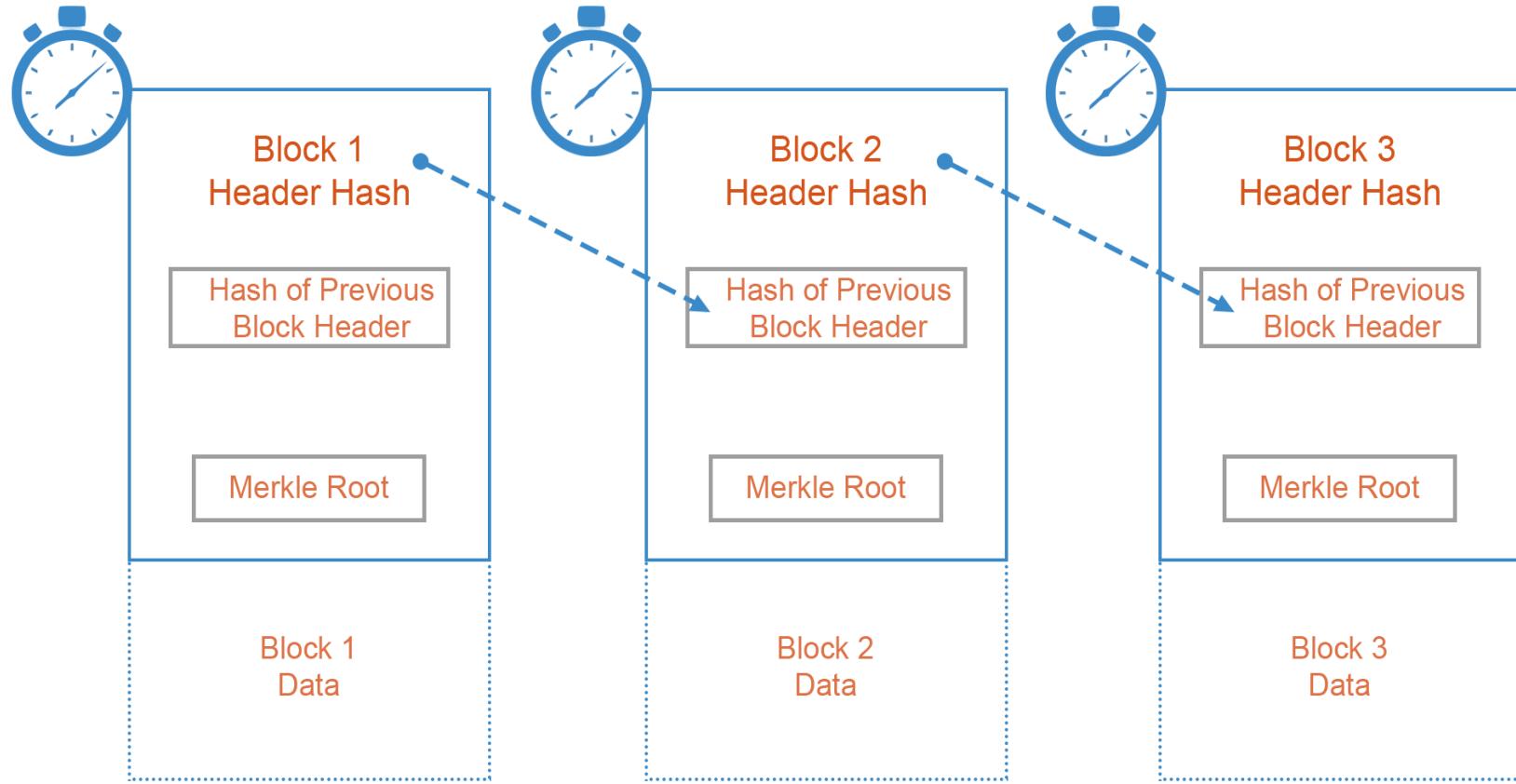


Cryptographic Hash

- ❖ Why would I want to use a hash?
 - ❖ Landlord and tenant can compare lease documents
 - ❖ Verification of software
 - ❖ If there's ANY difference between what should be and what is, it's easy to identify
 - ❖ Malware which makes slight changes to the original codebase can be easily detected
 - ❖ Important for self-driving cars, automation, IoT, etc..
 - ❖ Instantly compare two or more LARGE volumes of data to ensure they're the same
 - ❖ Has 1 bit been flipped in a 100TB file?



Blockchain Basics - recap





Blockchain Nodes And Web3



Nodes

- ❖ Different Nodes
 - ❖ Geth, Parity, Ganache, Truffle Developer Console
- ❖ Different Purpose
 - ❖ Developer-Network for Rapid Development
 - ❖ Unit Testing
- ❖ Libraries (Web3.js, Web3j, Web3PHP,...)
 - ❖ Connect to a Blockchain Node
 - ❖ “Speak” the right language (Data encoding/decoding)



- ❖ Go-Ethereum
- ❖ “GETH”
- ❖ Written in GOlang
- ❖ Can connect to Public Networks and Test-Networks
- ❖ Integrated “Developer” Mode
- ❖ Integrated Keystore
- ❖ Interactive JavaScript console



Ganache

- ❖ “Developer” Blockchain
- ❖ Exposes an HTTP-RPC Interface
- ❖ Proof of Authority (only one node)
- ❖ For Unit-Testing and Development
- ❖ Non-Persistent In-Memory Datastore



Remix Built In Blockchain

- ❖ IDE Blockchain Simulation
- ❖ Doesn't expose an HTTP-RPC Interface
 - ❖ Only running within Remix
- ❖ Not persistent



Web3.js

- ❖ JavaScript library to encode and decode data
- ❖ Connects to a blockchain node
- ❖ Usually using Ajax (Async) HTTP Requests or WebSockets
- ❖ Either directly (e.g. <http://localhost:8545>) or via MetaMask/Mist for example

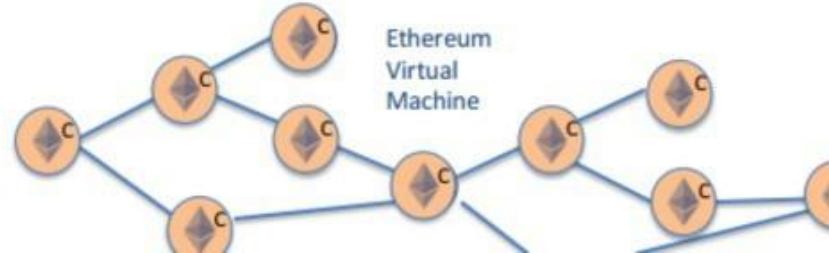
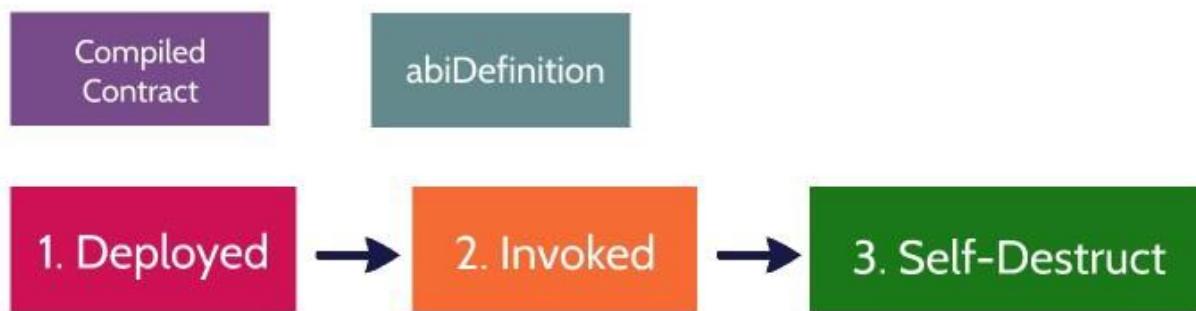


Smart Contract Life Cycle



Contract Lifecycle

Contract Lifecycle

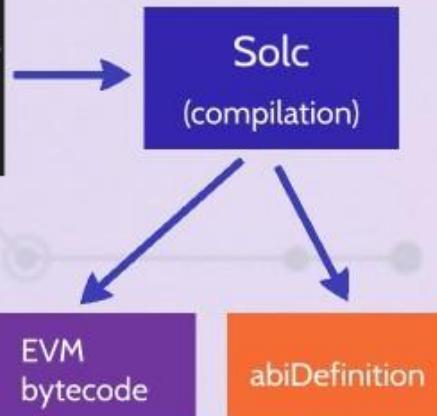




Compilation

Compilation

```
pragma solidity >=0.4.25;
contract MyContract {
    ...
}
```



"Application Binary Interface"

EVM
bytecode

abiDefinition

Contract
Lifecycle



EVM bytecode

EVM bytecode

- EVM Bytecode | Op codes
 - 140+ op codes

We talk about
this later again!

0x00	STOP	Halts execution
0x01	ADD	Addition operation
0x02	MUL	Multiplication operation
0x03	SUB	Subtraction operation
0x04	DIV	Integer division operation
0x05	SDIV	Signed integer
0x06	MOD	Modulo
0x07	SMOD	Signed modulo
0x08	ADDMOD	Modulo
0x09	MULMOD	Modulo
0x0a	EXP	Exponential operation
0x0b	SIGNEXTEND	Extend length of two's complement signed integer



AbiDefinition

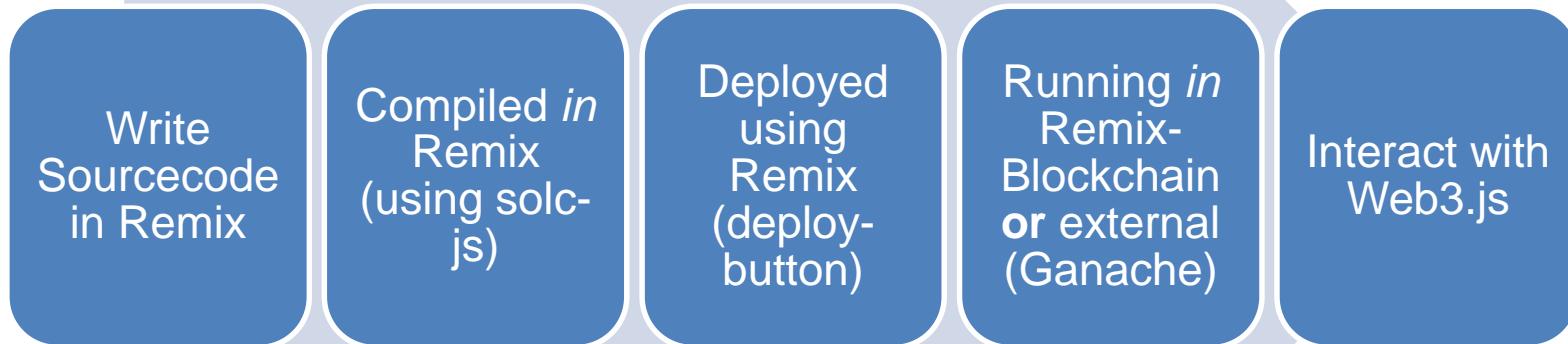
abiDefinition

```
[-] JSON
  [-] 0
  [-] 1
  [-] 2
  [-] 3
    anonymous : false
  [-] inputs
    [-] 0
      indexed : true
      name : "caller"
      type : "address"
    [-] 1
      indexed : false
      name : "oldNum"
      type : "uint256"
    [-] 2
      indexed : false
      name : "newNum"
      type : "uint256"
    name : "NumberSetEvent"
    type : "event"
```

- Contains function & event metadata
- **Needed for contract invocation & deployment**



Deployment using Remix (manual)





Deployment using Truffle





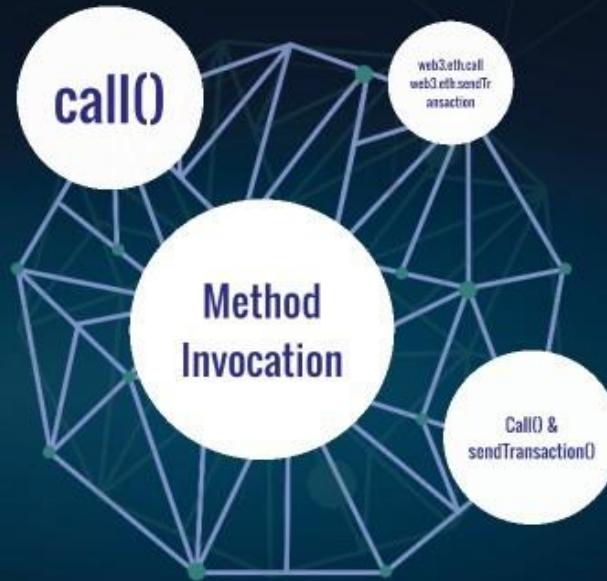
Smart Contract Concurrency and Events



Web3 JS API

WEB3 JS API

- Call()
- sendTransaction()





Web3 JS API

Method. call(...)

- Executed locally on the node
- Value= Return value from function
- No state changes in contract
- o execution fee

Method. sendTransaction(...)

- Executed on miner nodes
- Value= Transaction hash
- State changes in contracts
- Gas paid by caller

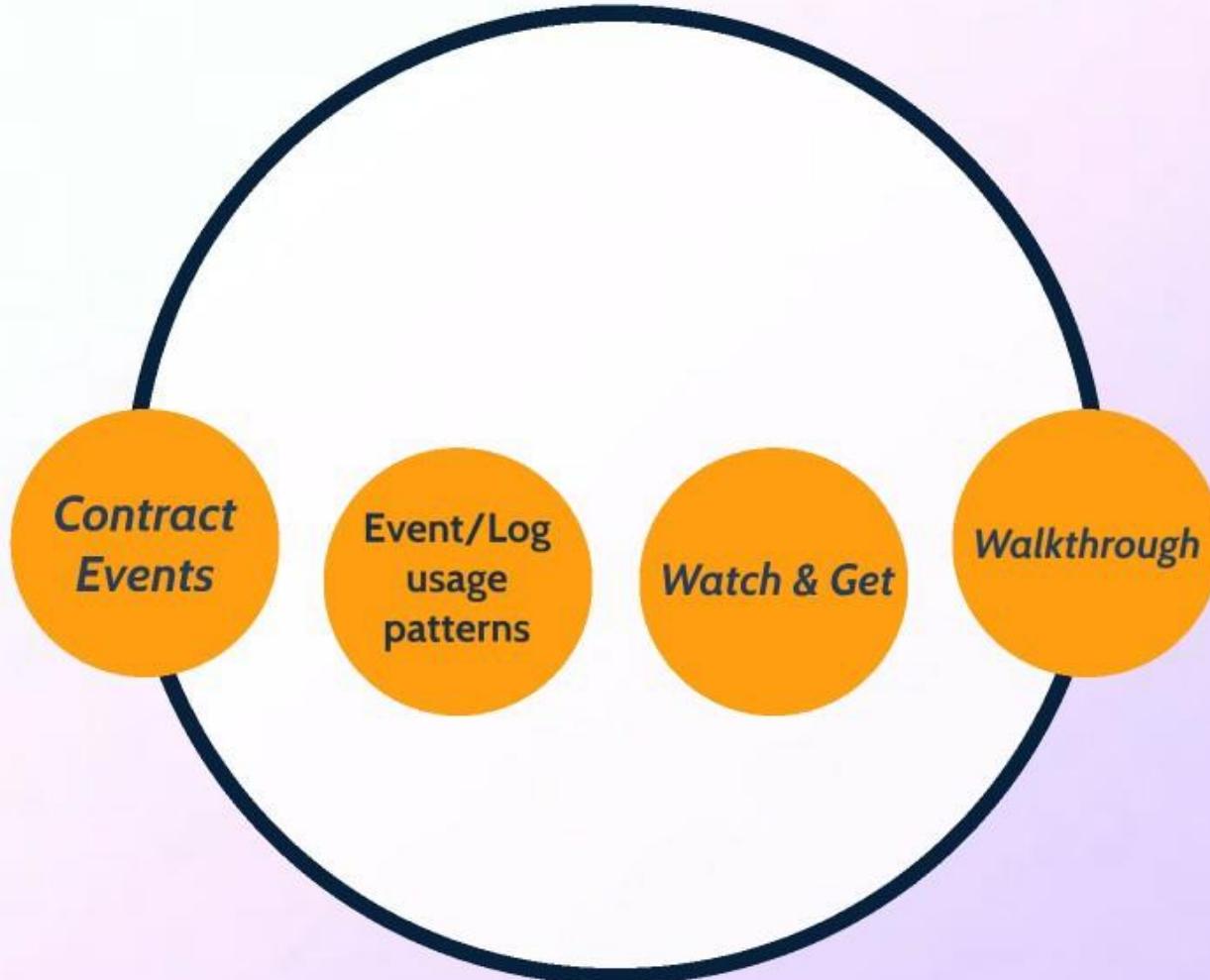


Call vs. Transaction

- ❖ “Call”
 - ❖ Reading operation(view/pure functions)
 - ❖ No need to reach consensus
- ❖ “Transaction”
 - ❖ Writing operation
 - ❖ Concurrent operation, Mining needs to happen, consensus has to be reached
 - ❖ Doesn’t return a value, returns a transaction hash
 - ❖ Events used to “react” to this



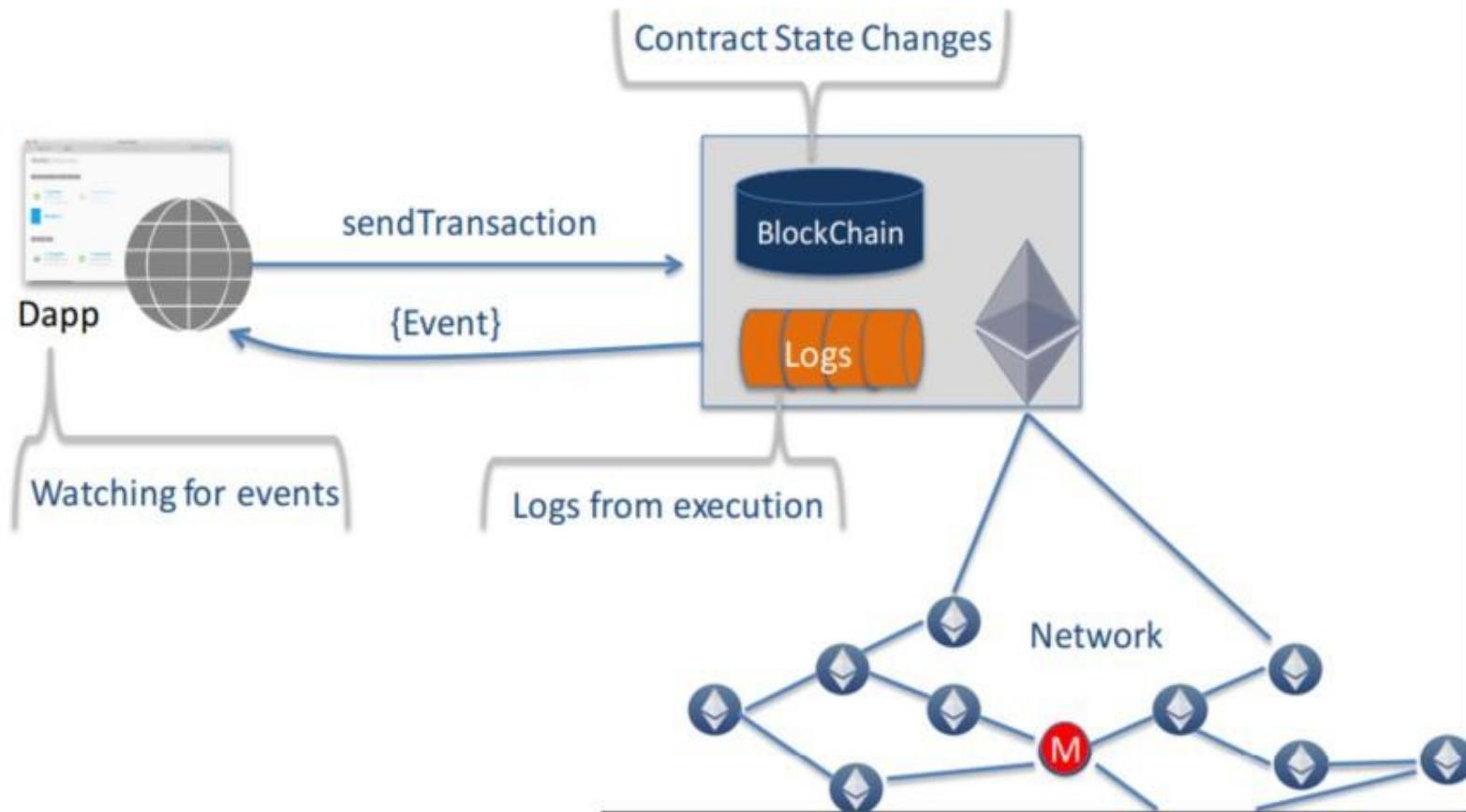
• Contract Events





• Contract Events

Ethereum Logs & Events





Logs / Events

- ❖ Events reside on a “Side Chain”
 - ❖ Cheap data storage
- ❖ Event Listener
 - ❖ Setup “on” the blockchain node, it’s not simple polling
- ❖ Smart Contracts can’t access Events, only emit



• Contract Events

Event/Log usage patterns

1. Receive data for transaction
2. Asynchronous trigger
3. Cheap data storage

1. Receives data for transaction

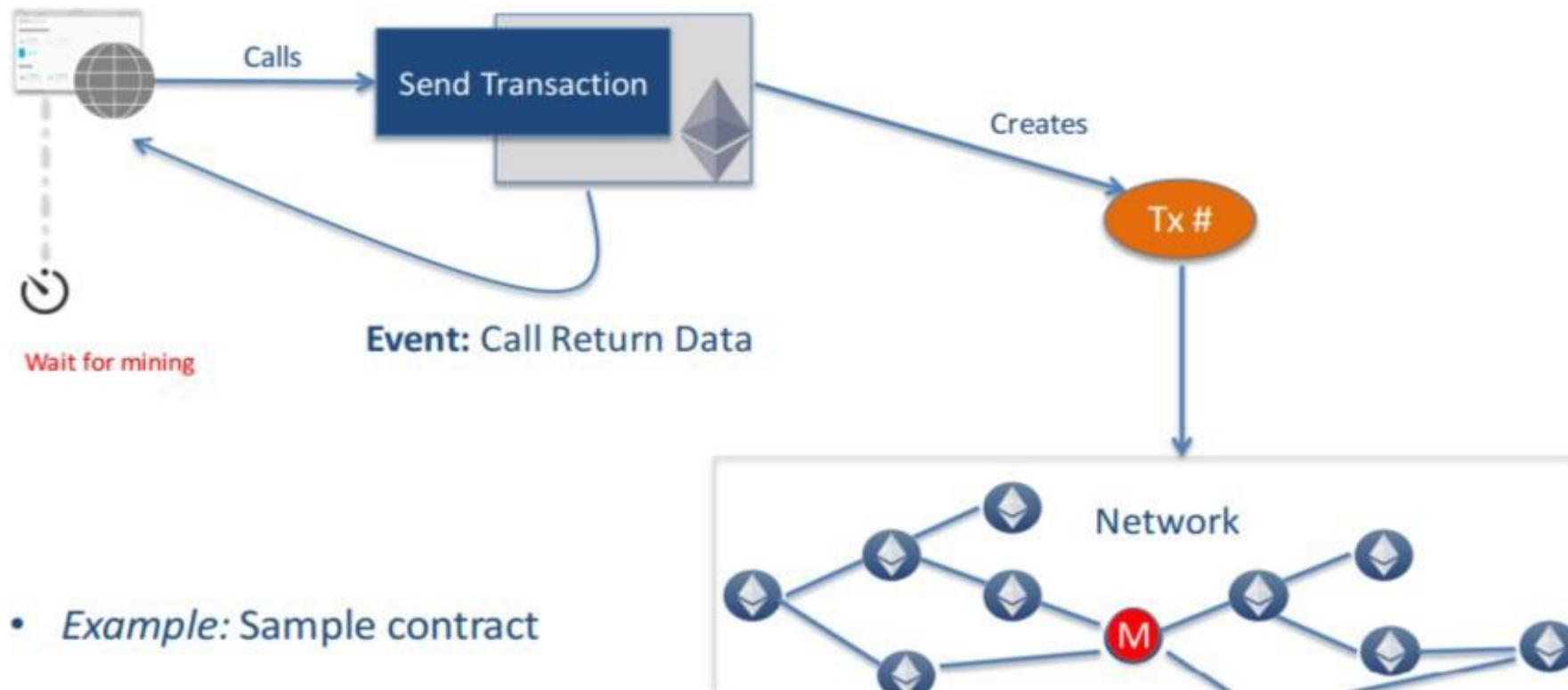
2. Asynchronous trigger

3. Data storage



• Contract Events

1. Receives data for transaction

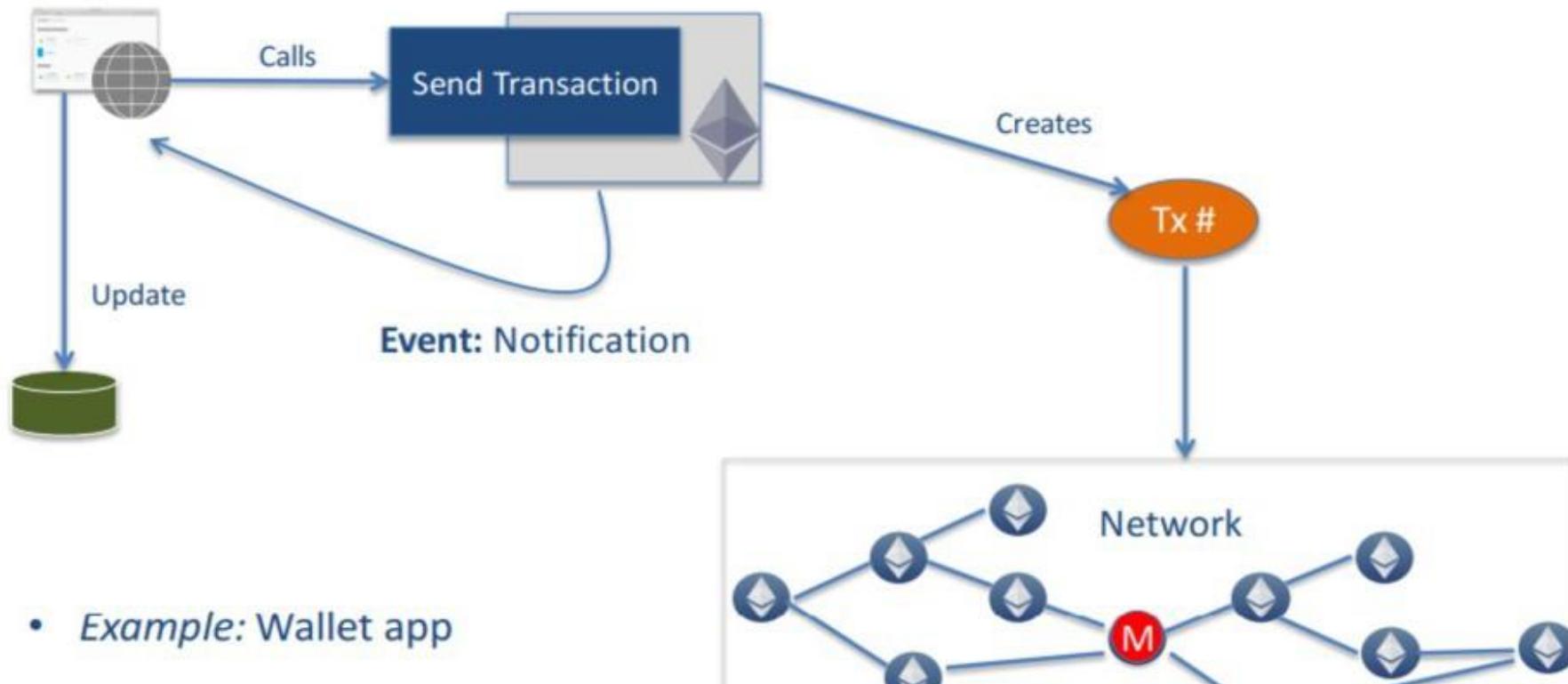


- *Example: Sample contract*



• Contract Events

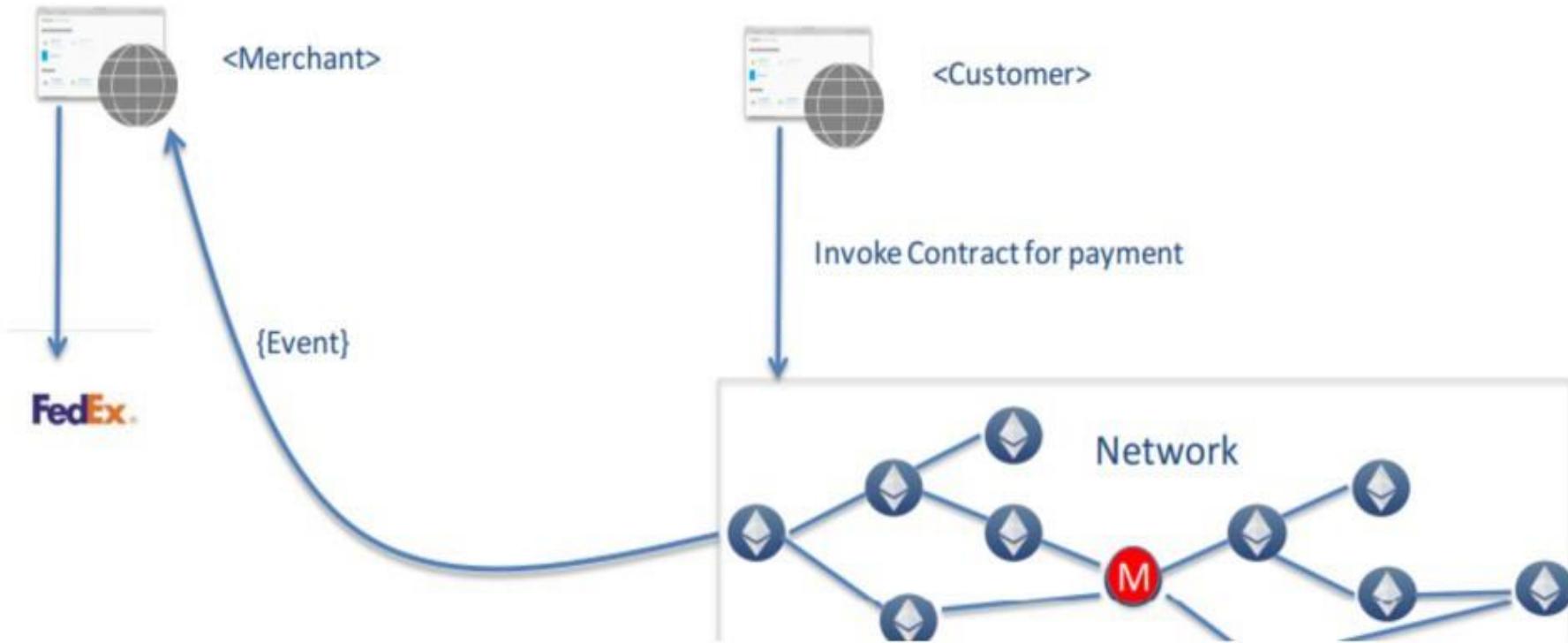
2. Asynchronous Notifications





• Contract Events

2. Asynchronous Notifications





• Contract Events

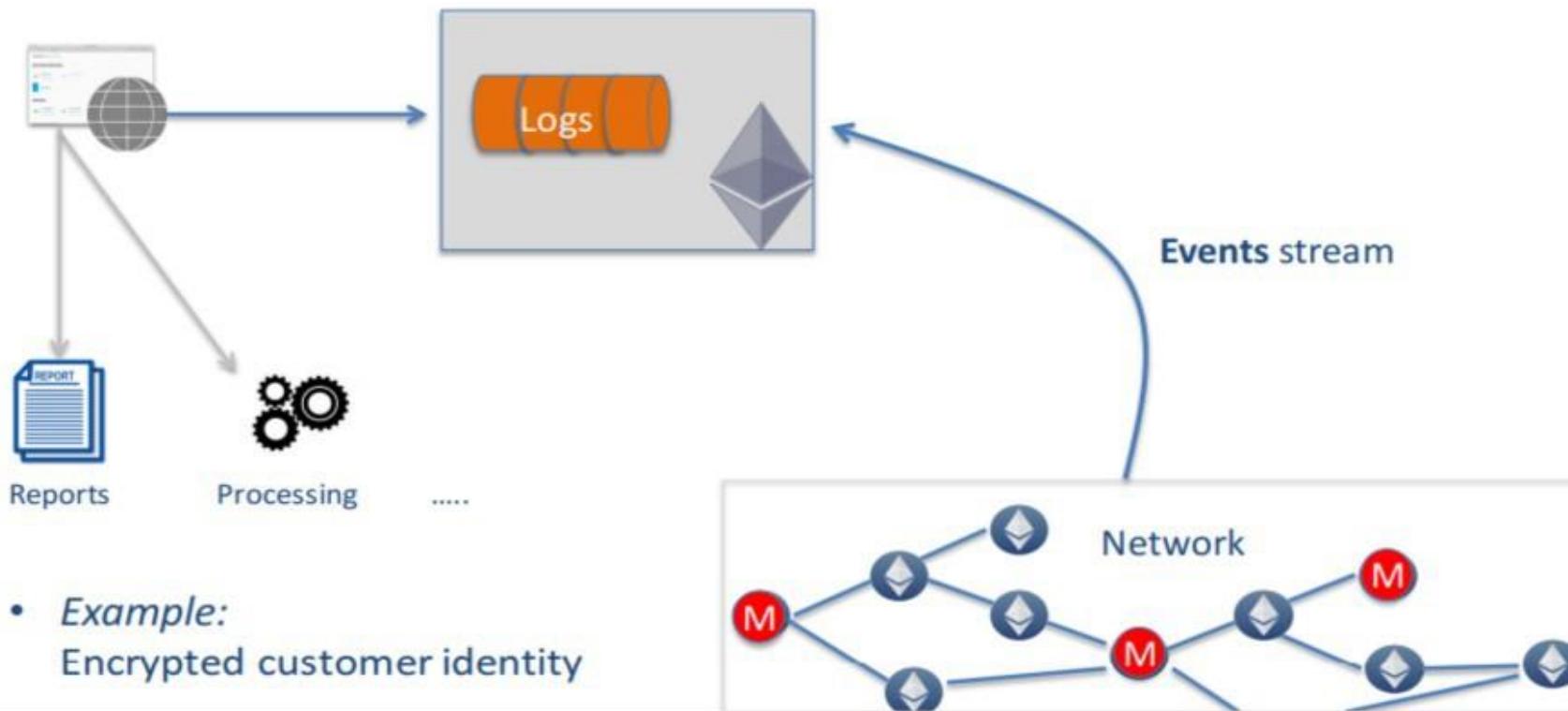
2. Asynchronous processing

- Front end (Dapp) can watch for events of interest
 - Example: Wallet app receives notification on receiving ethers
 - Example: Multisig contract shows transactions waiting for approval



• Data Storage

3. Data storage





• Data Storage

3. Data storage

- Cheaper than contract storage
 - Log data storage cost 8 Gas/byte
 - Contract data storage cost 20,000 Gas/32-byte
- Logs are **NOT** accessible from contracts



Events vs. Filter

❖ Event API

- ❖ High-Level
- ❖ Web3js 1.0.0: “web3.eth.contract => events.EventName...”
- ❖ Can filter for specific indexed parameters from one Contract
- ❖ Every Solidity Event can have up to 3 indexed parameters
- ❖ Used quite often
- ❖ Defined by the ABI Array

❖ Filter-API

- ❖ Low-Level
- ❖ Filter for anything anywhere
- ❖ `web3.eth.subscribe('logs', options [, callback]);`



Watch vs. Get

❖ Watch

- ❖ Listen for events continuously
- ❖ React to events
- ❖ Web3js 0.20.6:
 - ❖ Var event = myContract.MyEvent([options][,callback]);
- ❖ Web3js 1.0.0 beta:
 - ❖ Var event = myContract.events.MyEvent([options][, callback])
- ❖ Polling the blockchain node for event changes
 - ❖ event.stopWatching(); //web3@0.20.6
 - ❖ event.unsubscribe(); //web3@1.0.0-beta subject to change



Watch vs. Get

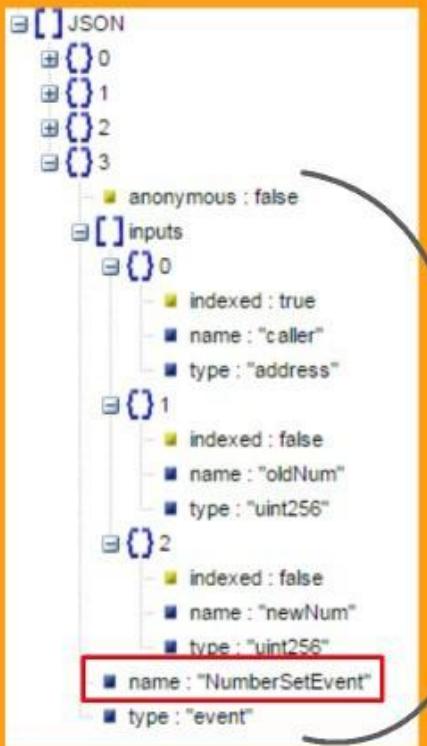
❖ Get

- ❖ Get a list of past events
 - ❖ Doesn't listen continuously
 - ❖ Returns an array, not an object
 - ❖ No need to stop
-
- ❖ `myContract.getPastEvents(event[, options][, callback])`
//web3@1.0.0 beta
 - ❖ `myEvent.get(function(error, logs){ ... });` //web3@0.20.6



Contract Events

Contract Events



Like methods, events are part of abiDefinition

```
1 pragma solidity ^0.4.6;
2 contract MyContract {
3
4     uint num;
5
6     event NumberSetEvent(address indexed caller, bytes32 indexed oldNum, bytes32 indexed newNum);
7
8     function getNum() returns (uint n) {
9         return num;
10    }
11
12    function setNum(uint n) {
13        uint old = num;
14        num=n;
15        NumberSetEvent(msg.sender,bytes32(old),bytes32(num));
16    }
17
18    function MyContract(uint x){num=x;}
19 }
```



Solidity

Functions and Modifiers



Functions

- ❖ Keyword “function”
 - ❖ Followed by arguments
 - ❖ Static typed
 - ❖ Solidity 5: Storage location must be given
- ❖ View/Pure
 - ❖ Reading functions
- ❖ Function Visibility
 - ❖ Public/Private/Internal/External
- ❖ Payable?
- ❖ Return
 - ❖ Statically typed

```
function f(uint a, uint b) public view returns (uint) {  
    return a * (b + 42) + now;  
}
```



View/Pure

❖ View Functions

- ❖ Read Only
- ❖ Read from Storage
- ❖ Call other View or Pure functions

❖ Pure

- ❖ Read Only
- ❖ Can't read from Storage
- ❖ Call only other Pure functions



Getter Functions

- ❖ compiler automatically creates getter functions for all public state variables

```
pragma solidity >=0.4.0 <0.6.0;

contract C {
    uint public data = 42;
}

contract Caller {
    C c = new C();
    function f() public view returns (uint) {
        return c.data();
    }
}
```



Fallback Function

- ❖ A contract can have exactly one unnamed function.
- ❖ Cannot have arguments
- ❖ Cannot return anything
- ❖ Has to have “external” visibility
- ❖ Executed on a call to the contract if none of the other functions match
- ❖ Better only rely on 2300 gas



Function Overloading

- ❖ multiple functions of the same name but with different parameter types
- ❖ also applies to inherited functions

```
pragma solidity >=0.4.16 <0.6.0;

contract A {
    function f(uint _in) public pure returns (uint out) {
        out = _in;
    }

    function f(uint _in, bool _really) public pure returns (uint out) {
        if (_really)
            out = _in;
    }
}
```



Function Modifiers

- ❖ used to easily change the behaviour of functions
- ❖ can automatically check a condition prior to executing the function
- ❖ inheritable properties of contracts and may be overridden by derived contracts

```
modifier costs(uint price) {
    if (msg.value >= price) {
        _;
    }
}
```



Solidity

Mappings and Structs



Mappings / Structs

- ❖ (u)int, Boolean, string very simple
- ❖ How can we store more advanced data structures
 - ❖ Tokens => Balance per Address
 - ❖ Supply Chain?
 - ❖ Training Certificates
 - ❖ Bonus Points
- ❖ We need better data structures!



Mappings

- ❖ Like Arrays in usage
- ❖ Arrays are actually disadvantageous in Solidity
 - ❖ But famous
- ❖ Iterating through large datasets costs Gas
 - ❖ Easy to run out of Gas. “for each” not recommended!
- ❖ Better use Mappings
 - ❖ Like hash-maps



Mappings

- ❖ `mapping(_KeyType => _ValueType)`
- ❖ The `_KeyType` can be any elementary type.
- ❖ `_ValueType` can be any type, including mappings.

- ❖ Access like arrays:
 - ❖ `balances[msg.sender] = newBalance;`



Struct

- ❖ define new types

```
struct Funder {  
    address addr;  
    uint amount;  
}
```

- ❖ ...and then use them in mappings:

```
mapping(address => Funder) myFunder;
```



Solidity

Files and Inheritance



Project Management

- ❖ Solidity can “import” files
- ❖ Truffle can import local files
 - ❖ And from eth-pm (like npm)
- ❖ Remix can import from GitHub



Project Management

- ❖ More than one contract in one file possible
 - ❖ There is no recommendation if that's good or bad
- ❖ Solc understands inheritance
- ❖ Multiple Inheritance possible
 - ❖ contract MyContract extends Owned,Transfer,Bookable
{...}



Understanding Transactions And Gas



Transactions

- ❖ “Plain-Text” Json Object
- ❖ With a “data” field
 - ❖ {from: 0xabc..., to: 0x123..., value: 0, **data**: “0xab3f...”}
- ❖ Data field = hex encoded function signature + arguments
- ❖ Storage location: Calldata
- ❖ Function myFunction(uint _arg) ...
 - ❖ Becomes: bytes4(keccak256('myFunction(uint256)'))
- ❖ The rest is the argument
- ❖ Example on next slide!

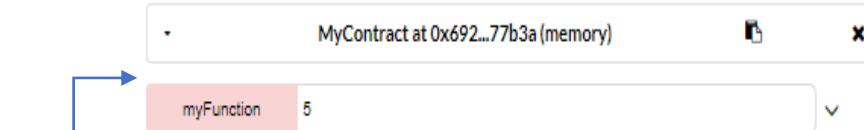
Transactions – Signature



```
1 pragma solidity >=0.4.24 <0.6.0;
2
3 contract MyContract {
4
5     uint myStorageVar;
6
7     function myFunction(uint _arg) public {
8         myStorageVar = _arg;
9     }
10 }
```

0x50628c960000000000000000
00000000000000000000000000
00000000000000000000000000
05

0x50628c96 =>
byte4(keccak256('myFunction
(uint256)'))



[vm]	from:0xca3...a733c to:MyContract.myFunction(uint256) 0x692...77b3a value:0 wei	Debug
status	0x1 Transaction mined and execution succeed	
transaction hash	0x44b641108a6335a7cdc9ed2bbc4313037172ff3a4a005e8c46082e980806a66d	
from	0xca35b7d915458ef540ade6068dfe2f44e8fa733c	
to	MyContract.myFunction(uint256) 0x692a70d2e424a56d2c6c27aa97d1a86395877b3a	
gas	3000000 gas	
transaction cost	41669 gas	
execution cost	20205 gas	
hash	0x44b641108a6335a7cdc9ed2bbc4313037172ff3a4a005e8c46082e980806a66d	
input	0x506...00005	
decoded input	{ "uint256 _arg": "5" }	
decoded output	{}	
logs	[]	
value	0 wei	



Ethereum Gas

Gas

- User invoking the transaction pays for the execution



Measures: kWh used



Measures: Gallons of water used

- Gas is the unit in which EVM resource usage is measured



Gas

- ❖ Gas is detached from Ether
 - ❖ Gas stays constant
 - ❖ Ether is volatile
- ❖ Gas is used when contract is executed
 - ❖ Contract Gas Usage depending on Code
 - ❖ *Gas Price* is defined by User
- ❖ Contract -> compile -> bytecode
 - ❖ EVM Assembly Instruction bytecode representation
- ❖ EVM Assembly Instructions cost Gas



Example

The screenshot shows the Remix IDE interface. On the left, there are two tabs: "browser/C07Arrays.sol" and "browser/myFun.sol". The "myFun.sol" tab is active, displaying the following Solidity code:

```
pragma solidity >=0.4.24 <0.6.0;
contract MyContract {
    uint myStorageVar;
    function myFunction(uint _arg) public {
        myStorageVar = _arg;
    }
}
```

The right side of the interface has several sections:

- Environment:** Set to "JavaScript VM".
 - Account:** 0xca3...a733c (99.99999999999999)
 - Gas limit:** 3000000
 - Value:** 0
- Deploy:** A red circle highlights the "Debug" button next to the "Deploy" button.
- Transactions recorded:** Shows two transactions:
 - [vm] from:0xca3...a733c to:MyContract.(constructor) value:0 wei Debug hash:0x9bb...5ef68
 - [vm] from:0xca3...a733c to:MyContract.myFunction(uint256) 0x92...77b3a value:0 wei data:0x506...00005 logs:0 Debug
- Deployed Contracts:** Shows "MyContract at 0x692...77b3a (memory)".

The screenshot shows the Remix IDE in "Debugger" mode. The top bar includes "Compile", "Run", "Analysis", "Testing", "Debugger", and "Settings".

The main area displays an assembly trace:

- Block number: 111 DUP1
- Transaction index or hash: 112 PUSH1 00
- 114 DUP2
- 115 SWAP1
- 116 SSTORE
- 117 POP
- 118 POP
- 119 JUMP

On the right, a sidebar provides details about the current step:

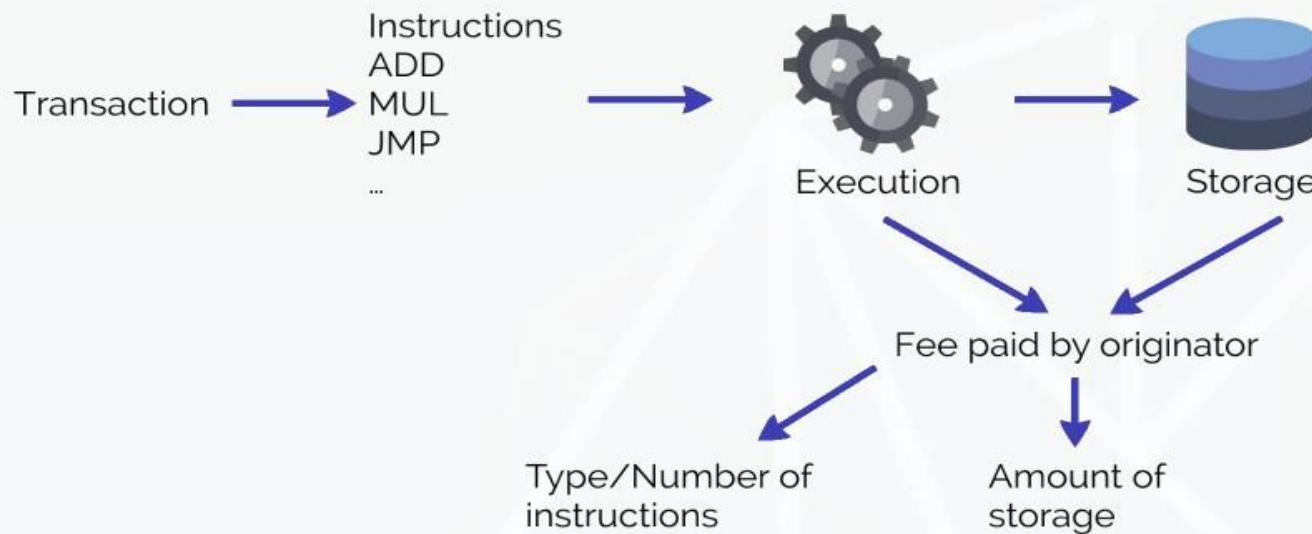
- vm trace step: 52
- execution step: 52
- add memory:
- gas: 3
- remaining gas: 2978356
- loaded address: 0x692a70d2e424a56d2c6c
- 27aa97d1a86395877b3a

A red bracket highlights the assembly steps 111 through 119, and another red bracket highlights the sidebar details.



Ethereum Gas

Gas Calculation





Ethereum Gas

Opcodes & Gas

Gas cost	QUICKSTEP	FASTEESTSTEP	FASTSTEP	MIDSTEP	SLOWSTEP	EXTSTEP	
	2	3	5	8	10	20	
ADDRESS	DUP	MUL		ADDMOD	JUMPI	BLOCKHASH	
ORIGIN	SWAP	DIV		MULMOD	EXPBASE	BALANCE	
CALLER	PUSH	MOD		JUMP		EXTCODESIZE	
CALLVALUE	ADD	SDIV				EXTCODECOPYBASE	
CALLDATASIZE	SUB	SMOD					
CODESIZE	LT	SIGNEXTEND					
GASPRICE	GT						
COINBASE	SLT						
TIMESTAMP	SGT						
NUMBER	EQ						
DIFFICULTY	AND						
GASLIMIT	OR						
POP	XOR						
PC	NOT						
MSIZE	BYTE						
GAS	CALLDATALOAD						
	CALLDATACOPY						
	CODECOPY						
	MILOAD						
	MSTORE						
	MSTORE8						



Ethereum Gas

Fee Calculation

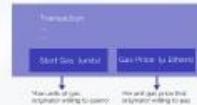
gasUsed =

- Instructions executed (summed up gas)

gasPrice =

- User specified in the transaction
- Miners decides the minimal acceptable price

$$\text{Transaction Fee} = \text{gasUsed} * \text{gasPrice}$$





Gas Price

- ❖ Gas amount fixed, but sometimes not known upfront
 - ❖ Provide more gas than necessary
 - ❖ Rest will be refunded
- ❖ Gas Price determines how *fast* the transaction will be mined
- ❖ Two types of exceptions
 - ❖ Failed require => Rest of Gas will be refunded
 - ❖ Failed assertion => All provided gas will be consumed.
- ❖ Goal: Write Low-Gas Consuming Contracts
 - ❖ There is also a “block gas limit”. A Blockchain is no mainframe.



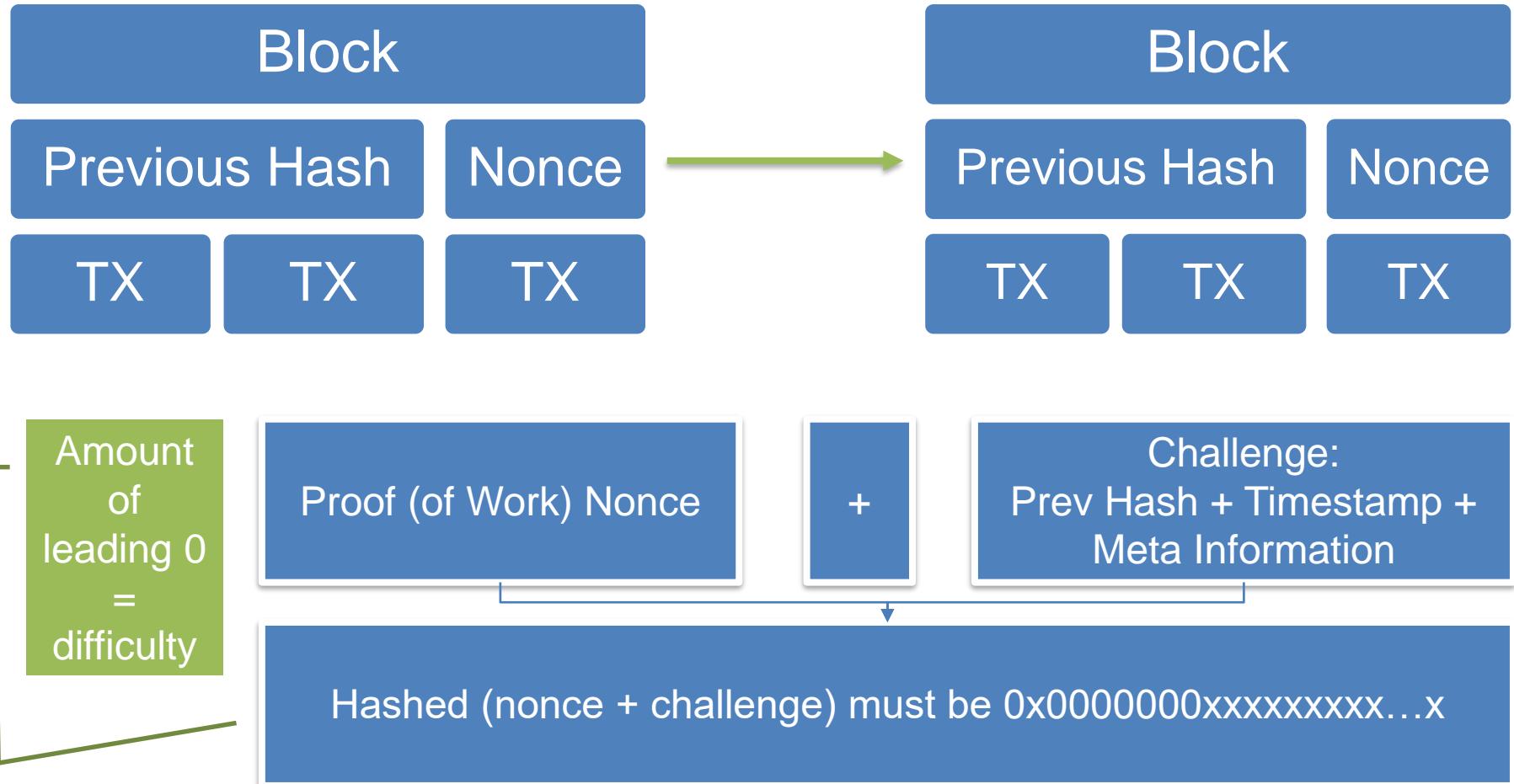
Types of Consensus

Consensus PoW



- ❖ Proof of Work Consensus
 - ❖ When a block is full, each node competes to solve a guessing game problem
 - ❖ This problem requires computational resources to quickly guess the “Nonce” of the transaction block
 - ❖ Miners try to guess the “nonce”
 - ❖ All block data plus the current guess (nonce) are run through a cryptographic hash
 - ❖ If the result matches the current level of “difficulty”, the miner has guessed the right answer
 - ❖ The miner with the answer shares it with all other miners, Miners will confirm the answer is correct by using the nonce with their block data to try and get the correct result. When 51% of the miners confirm the nonce is correct, the transaction is added to the Blockchain
 - ❖ The result is Proof of Work Consensus

Proof of Work



Proof of Stake (PoS)



- ❖ Proof of Stake Consensus
 - ❖ Proposed as an alternative to Proof of Work
 - ❖ Attempt to overcome scalability concerns imposed by PoW consensus
 - ❖ Removes the guessing game from consensus
 - ❖ Mining no longer requires specialized and powerful hardware
 - ❖ Many feel that specialized hardware requirements lead to centralization
 - ❖ Blockchain is about de-centralization
 - ❖ Less energy intensive form of consensus
 - ❖ Addresses concerns about “green” mining



Proof of Stake (PoS)

- ❖ Proof of Stake Consensus
 - ❖ Ok, how does it work?
 - ❖ Block of transactions created
 - ❖ When it's time for group consensus, all who wish to participate lock up funds in a stake
 - ❖ A random 'player' is selected
 - ❖ That player's block data is shown to all other participants
 - ❖ Other players stake on the validity of the block transactions



Proof of Stake (PoS)

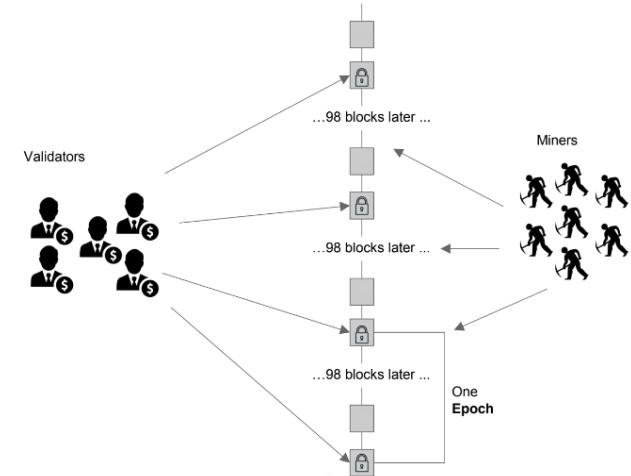
❖ Proof of Stake Consensus

- ❖ If the majority agree with the proposed block, the random player is rewarded, as are all who staked on that player
- ❖ If the majority disagree, the random player gets no reward AND loses their stake!
- ❖ Then a new player is randomly selected to share their block data



PoW vs PoS

- ❖ PoW vs PoS
 - ❖ Work for a reward vs make a safe bet for a reward
 - ❖ Security vs Speed
 - ❖ Centralization vs Decentralization
 - ❖ Proven vs New
 - ❖ Capital spent on hardware vs capital spent on staking funds
- ❖ Ethereum – quickly moving to PoS
 - ❖ Serenity (Casper)
- ❖ 0.1.0 Released May 2018





Ethereum 2.0

- ❖ Beacon Chain
 - ❖ Proof of Stake
- ❖ 1000x higher scalability
- ❖ EWASM instead of EVM
 - ❖ EWASM = Ethereum WebAssembly
- ❖ Account Abstraction?

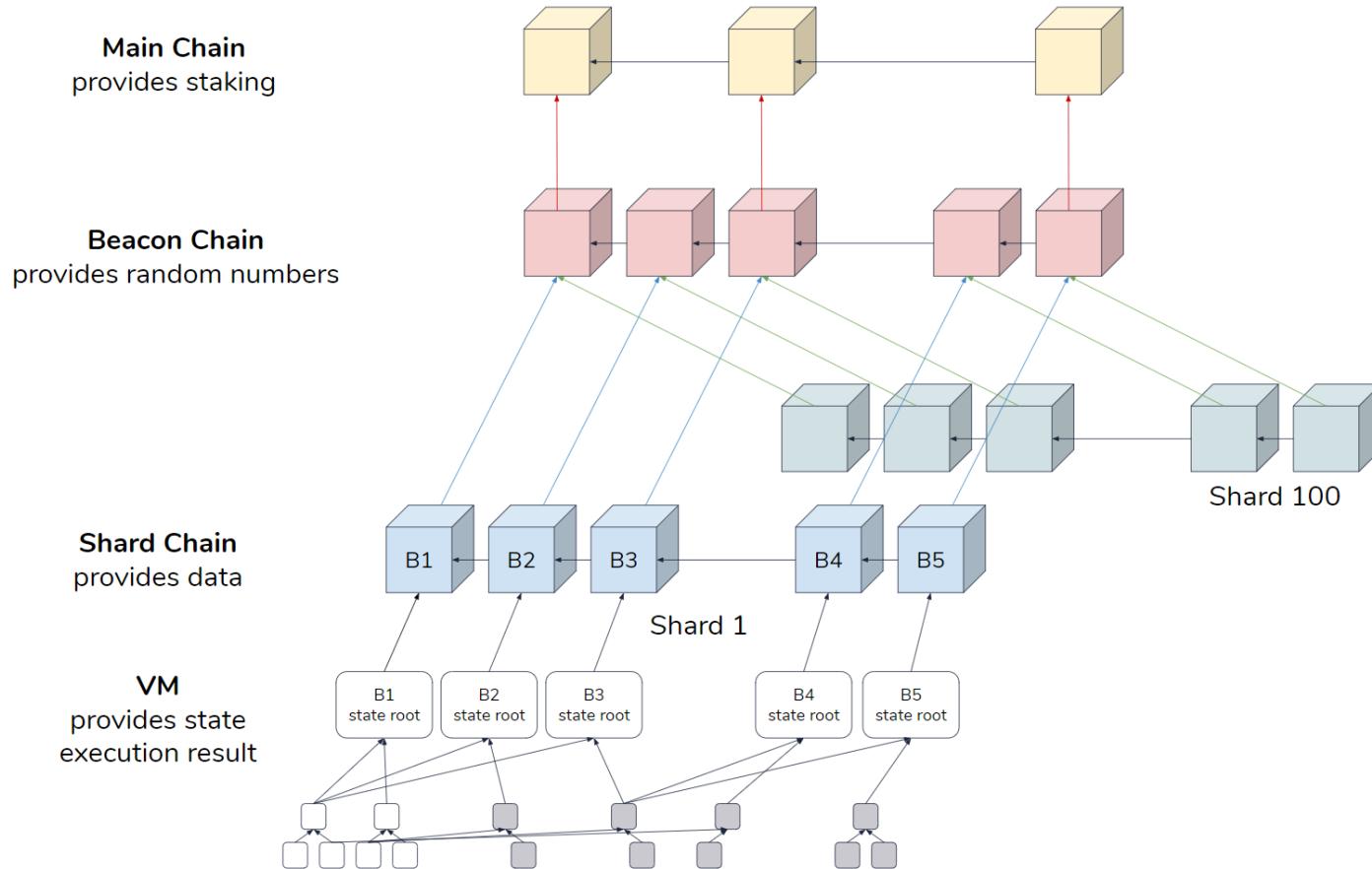


Beacon Chain

- ❖ Pure PoS
- ❖ Deposit 32 Ether
 - ❖ Become a Validator
 - ❖ Anyone can be a Validator
- ❖ Main PoS Chain
- ❖ Base Layer of Sharding Solution



Ethereum 2.0



Credit: Hsiao-Wei Wang



Beacon Chain

- ❖ Managing Validators
 - ❖ 32 Ether into a Smart Contract on the PoW Chain
 - ❖ Event is emitted, picked up by beacon chain clients
 - ❖ Validator is then active
 - ❖ Exit after 97 days (might change), refund only to Shard Chain, not on the old Main-Net
- ❖ Provide Randomness
- ❖ Block Proposers
- ❖ Committee



Beacon Chain

- ❖ Managing Validators
- ❖ Provide Randomness
 - ❖ RANDAO => Many contributors with random numbers will lead to one single random output number
 - ❖ Commit-Reveal Scheme
 - ❖ Sufficiently Robust
- ❖ Block Proposers
- ❖ Committee



Beacon Chain

- ❖ Managing Validators
- ❖ Provide Randomness
- ❖ Block Proposers
 - ❖ Eth2.0 has a 16 seconds heartbeat
 - ❖ Called “slots”
 - ❖ Proposers chosen randomly will propose a new block based on attestations from validators
 - ❖ A committee votes on the block to be added
- ❖ Committee



Beacon Chain

- ❖ Managing Validators
- ❖ Provide Randomness
- ❖ Block Proposers
- ❖ Committee
 - ❖ Critical for security
 - ❖ Vote on the true history of the chain
 - ❖ Ideally *all* validators
 - ❖ Or as much validators which are quick enough
 - ❖ Validating each shards proposers



Beacon Chain

❖ Rewards

- ❖ Unknown yet, but probably 1-5 ETH?
- ❖ Subject to change

❖ Penalties

- ❖ Slashing: Some Ethers are removed for bad behavior
- ❖ Ejecting: They are removed as validators
- ❖ Being Absent (“quadratic leak”): small penalty
- ❖ If <16 Ether automatically ejected

❖ Crosslinks

- ❖ Combine all Shards state into the beacon-chain and finalize it



Simple Terms

- ❖ Put 32 Ether in PoW Chain Smart Contract
 - ❖ Contract emits event
 - ❖ 32 Ether are locked *forever*
- ❖ 32 Ether then pop up on the Beacon Chain
 - ❖ You become a validator
- ❖ The rest is abstracted away from you as developer
 - ❖ The only noticeably good thing: It will be much faster



Proof of Stake (PoS)

❖ Proof of Stake Consensus

- ❖ No “computing” is ever performed during consensus, only staking/wagering
- ❖ Any kind of device can stake, regardless of computing power

Other Consensus Mechanisms



- ❖ Other consensus mechanisms
 - ❖ Proof of Activity
 - ❖ Hybrid of PoW and PoS
 - ❖ Empty template blocks are mined (PoW), then filled with transactions which are validated via PoS
 - ❖ Proof of Burn
 - ❖ Coins are “burned” by sending them to an address where they cannot be retrieved
 - ❖ The more coins you burn, the better your chances of being selected to mine the next block
 - ❖ Eventually, you must stake more by burning more coins

Other Consensus Mechanisms



- ❖ Other consensus mechanisms
 - ❖ Proof of Capacity (aka Space)
 - ❖ Pay to play with hard drive space or memory
 - ❖ The most space you ‘stake’ the better your odds of being selected to mine the next block
 - ❖ Consensus algorithm generates large data sets called ‘plots’ which consume storage
 - ❖ Major criticism – this method has no real deterrent for bad actors

Nonce



Other Consensus Mechanisms



- ❖ Other consensus mechanisms
 - ❖ Proof of Elapsed Time
 - ❖ Created by Intel to run on their trusted execution environment
 - ❖ Similar to PoW, far more energy efficient
 - ❖ Major criticism – requires trust in Intel, places power back in the hands of a central authority
 - ❖ Proof of Authority
 - ❖ Uses a set of “authorities” – nodes that are explicitly allowed to create new blocks and secure the blockchain
 - ❖ Replacement for PoW - Private blockchains only
 - ❖ Earn the right to become a validator/authority



Consensus Mechanisms

- ❖ Consensus protocols are the key to Blockchain!
 - ❖ Blockchain consensus mechanisms are the nuts and bolts of validation.
 - ❖ Think Internet & TCP/IP – transmission of bytes of data across the Internet
- ❖ PoW is the only tried and true (9+ years in use). PoS is coming, rolling out now.
- ❖ The rest are concepts and development ideas that different developers are working on. Some are in a test trial phase.



Mining a Block



Transaction Implications

- ❖ Transactions take time to ‘confirm’
- ❖ Each transaction, once it’s in an accepted block has a height
 - The height of a block is the **number of blocks in the chain between it and the genesis block**
- ❖ Each increase in blockchain height is called a confirmation
- ❖ A transaction 5 blocks below the top of the chain is said to have ‘6 confirmations’
- ❖ Wait for 6 confirmations for anything of value



Blockchain Forks

- ❖ Upgrades to the protocol can cause problems – but can be managed
- ❖ Blocks that are created have a version number
- ❖ New blocks using the new protocol use a different version number
- ❖ If the upgrade is backward compatible, it's a soft fork
- ❖ If the upgrade isn't backward compatible, it's a hard fork
- ❖ Hard forks are much harder and we try to avoid them



Ethereum Tooling



Overview

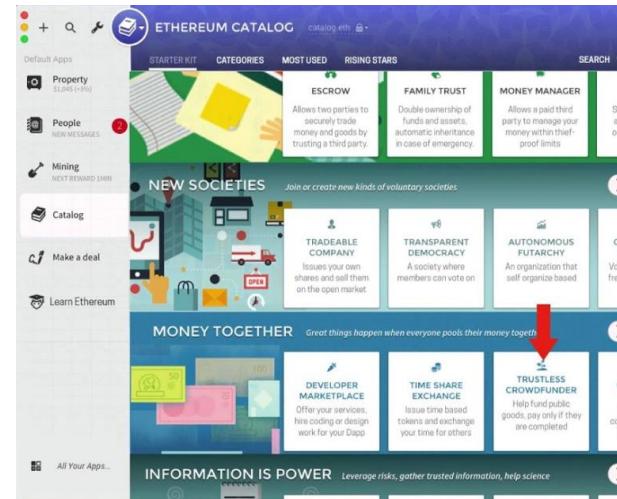
Browsers	Frameworks	KeyStore (Wallet)	Networks
<ul style="list-style-type: none">• MIST• Status.im• MetaMask (to some extend)	<ul style="list-style-type: none">• Truffle• Embark• DApp (yes, it's called that way!)• Populus• Etherlime• ...	<ul style="list-style-type: none">• In-Node (Geth)• MetaMask• MIST• Hardware Wallet• Paper?• ...	<ul style="list-style-type: none">• Main (1)• Ropsten (3)• Rinkeby (4)• Kovan (42)• Sokol (77)• Start your own!• ...
Languages	Consensus Algorithms	Blockchain Nodes	IDEs
<ul style="list-style-type: none">• Solidity• Vyper• Bamboo• ...	<ul style="list-style-type: none">• PoW• PoS• PoA• ...	<ul style="list-style-type: none">• Geth• Parity• Ganache• Kaleido• Ethereum on Azure• ...	<ul style="list-style-type: none">• Remix• Visual Studio Code (with Solidity Plugin)• Atom• Superblocks Lab



Browsers - MIST

❖ MIST

- ❖ Is an Ethereum Browser which was built using electron
- ❖ Runs a full “GETH” Node underneath
- ❖ Including Keystore
- ❖ And a chromium browser on top
- ❖ Like Chrome with MetaMask but with a full local Ethereum node



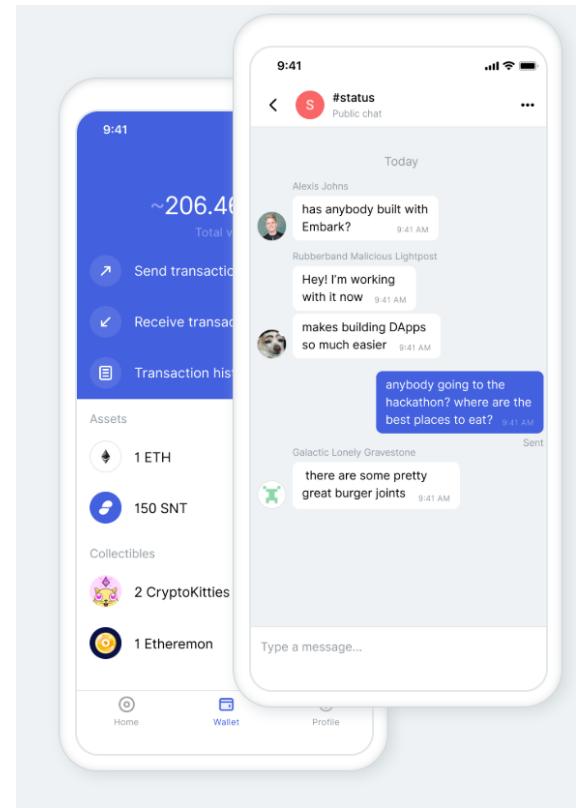
Credit: Ethereum

Browsers - Status.im



- ❖ Chat/Browser for Mobile
- ❖ Connects to a Blockchain node
- ❖ KeyStore integrated

- ❖ Can connect to Dapps
- ❖ Can “browse” Dapps like with MetaMask+Chrome on PC





Browsers - MetaMask

- ❖ Browser Plugin
- ❖ Connects to a Blockchain (Infura most likely)
- ❖ KeyStore
- ❖ Intercepts Transactions
 - ❖ And shows confirmation window



Frameworks - Truffle

- ❖ “A world class development environment, testing framework and asset pipeline for blockchains using the Ethereum Virtual Machine (EVM), aiming to make life as a developer easier.”

- ❖ Lifecycle Management
- ❖ Unit Testing
- ❖ Scriptable Deployments
- ❖ + Truffle Boxes!

- ❖ Good for local development and teams



Frameworks - Embark

- ❖ “Embark is a framework that allows you to easily develop and deploy Decentralized Applications (DApps).”

- ❖ Test Driven Development
- ❖ Manage Different Chains
- ❖ Work with Decentralized Storage
- ❖ Work with Decentralized Communication



Frameworks - Populus

- ❖ Populus is a smart contract development framework for the Ethereum blockchain.
- ❖ Python based test environment
- ❖ Very similar to the previous frameworks



Frameworks - Etherlime

- ❖ etherlime is an ethereum development and deployment framework based on ethers.js.
- ❖ allows for ultimate control by the developer
- ❖ adds verboseness in the deployment process



Networks

- ❖ There is the Main-Net
 - ❖ Network ID 1
- ❖ But new things might need tests on protocol level
 - ❖ Ropsten, Rinkeby, Kovan, Sokol
 - ❖ Faucets (free Ether)
 - ❖ Test-Net to try Smart Contracts (Beta Customers)
- ❖ Can start your own Blockchain based on the Ethereum Protocol (with a unique network ID)



Language - Solidity

- ❖ contract-oriented programming language
- ❖ initially proposed in August 2014 by Gavin Wood
- ❖ is designed around the ECMAScript syntax to make it familiar for existing web developers
- ❖ Support for
 - ❖ Complex member variables
 - ❖ arbitrarily hierarchical mappings and structs
 - ❖ inheritance, including multiple inheritance with C3 linearization
 - ❖ An application binary interface (ABI) facilitating multiple type-safe functions within a single contract



Language - Other

- ❖ Vyper (IDE: <https://vyper.online/>) :
 - ❖ Vyper is a smart contract development language built with the following goals:
 - ❖ **Security** - it should be possible and natural to build secure smart contracts in Vyper.
 - ❖ Language and compiler **simplicity** - the language and the compiler implementation should strive to be simple.
 - ❖ **Auditability** - Vyper code should be maximally human-readable
- ❖ Bamboo: a language for morphing smart contracts
 - ❖ state transition explicit and avoids reentrance problems by default



Consensus Algorithms

- ❖ Proof of Work
- ❖ Proof of Stake
- ❖ Proof of Authority
- ❖ Proof of ...

- ❖ We talked about it.

- ❖ Ideally Nodes can “understand” more than one Consensus algorithm for Enterprise usage in private or consortium networks



Blockchain Nodes

❖ Geth

- ❖ Go-Ethereum
- ❖ Written in GO
- ❖ Keystore + Ethereum Node + Web3js Interface

❖ Parity

- ❖ Rust
- ❖ Web-Interface

❖ Ganache

- ❖ Developer Node
- ❖ In-Memory “private” blockchain
- ❖ Only One Node possible

❖ Kaleido

- ❖ full-stack platform
- ❖ Made Radically Simple
- ❖ RAFT, IBFT, POA



IDEs

- ❖ Remix
 - ❖ Browser Based
 - ❖ Integrated Debugger
 - ❖ Integrated Blockchain (simulation)
 - ❖ Very Handy
- ❖ Local Development
 - ❖ Using VS Code or Atom

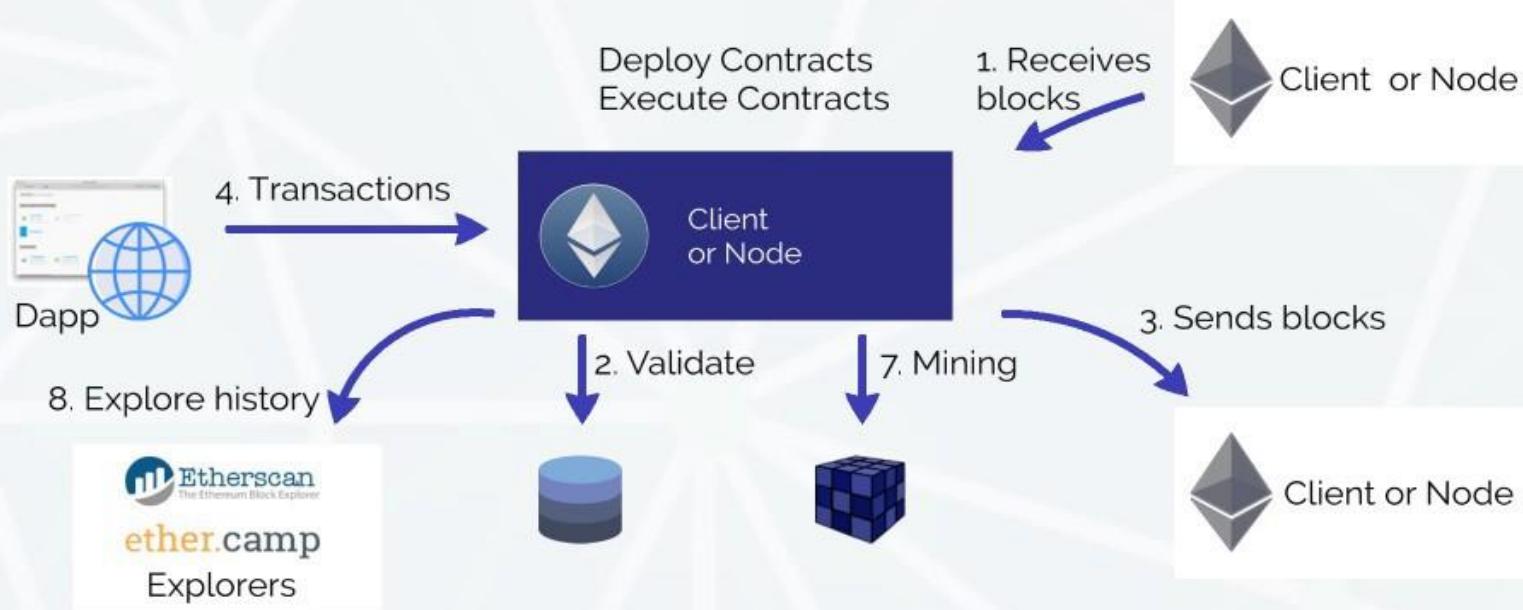


Go-Ethereum In-Depth



Ethereum Client

Ethereum Client





Clients

- ❖ Go-Ethereum
 - ❖ Written in Go
- ❖ Parity
 - ❖ Rust
- ❖ Pantheon
 - ❖ Java
- ❖ CPP-Ethereum
 - ❖ C++



Go-Ethereum

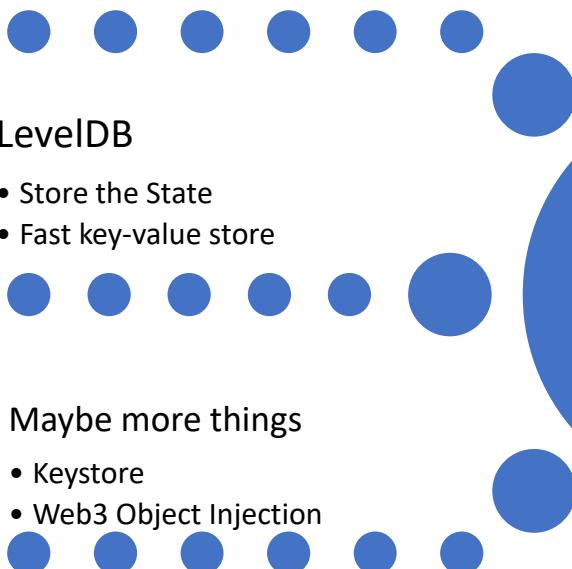
- ❖ Blockchain Node
- ❖ KeyStore
- ❖ Interactive JavaScript console
 - ❖ Might change
- ❖ Regular updates
 - ❖ But no update mechanism
- ❖ Several “Sync Modes”
- ❖ Several Consensus Protocols
- ❖ Create your own Private Network



Client Flow

Local Connectivity

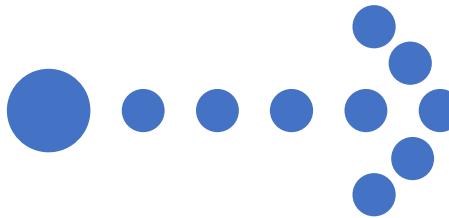
- JSON-RPC (8545)
- WS-RPC
- IPC



LevelDB

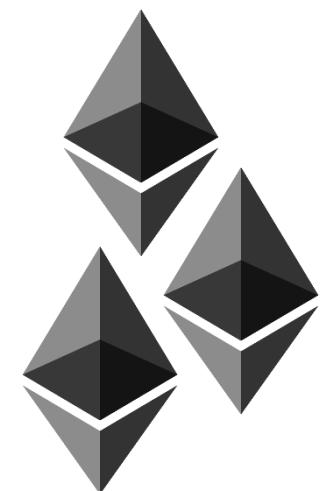
- Store the State
- Fast key-value store

DevP2P Connection to the Network



Maybe more things

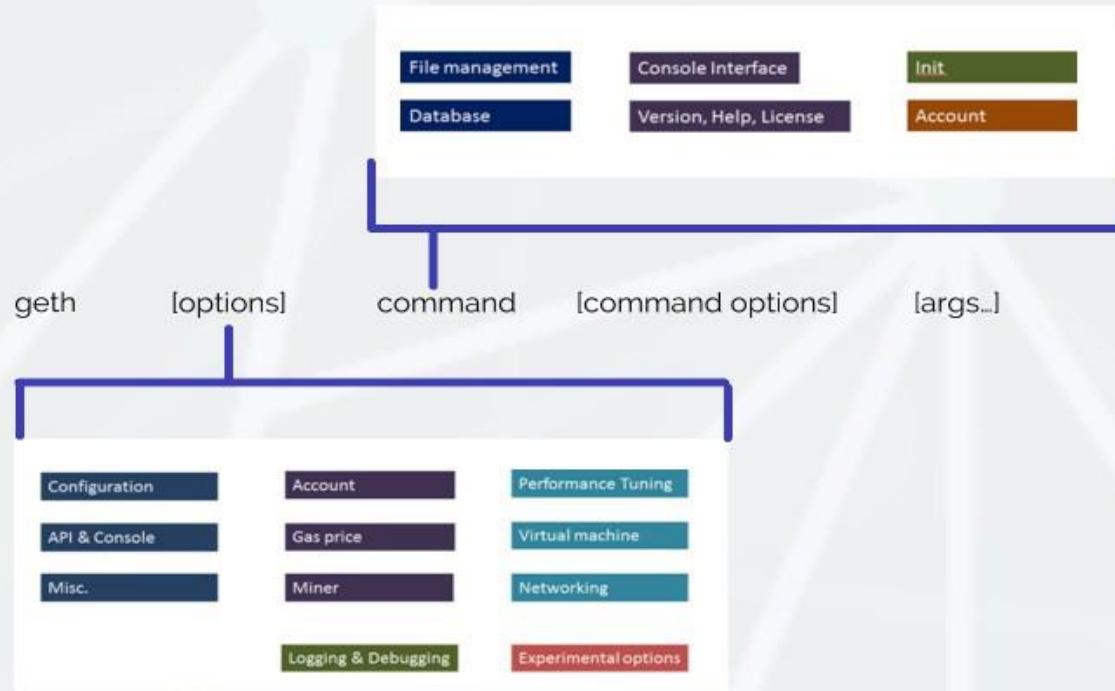
- Keystore
- Web3 Object Injection





Geth

Geth CLI : Usage





Geth

geth [options] command [command options] [args...]

Configuration

API & Console

Misc.

--identity	"MyTestNode"	Custom identity
--datadir	"/data-dir"	Directory for writing the chain data
--keystore	"/key-dir"	Keystore location
--networkid	"value"	Default=1 Live, Testnet=3, Private = User assigned
--testnet		Connect the node to the testnet(ROPSEN)
--fast	(use with --cache)	Fast synching through statedownloads
--dev		Developer mode; private network; multiple debug flags



Private Networks

Private Dev Network

Isolated network with 1 node

- Fast transactions
- Extremely low storage needs

Suitable for experimentation & learning

Mining needs to be ON for: (memory intensive)

- Trying out transactions
- Getting ethers

How it -dev
work?



Javascript Runtime

Javascript Runtime

Geth implements a Javascript Runtime Environment

- Execute the javascript code
- Invoke the Management API
- web3, eth, personal, admin, miner, personal ...

Supports interactive as well as non-interactive mode



Geth Javascript

Geth Javascript API

- Supports web3 Dapp API

Go-Ethereum supports additional API

- Available over JSON-RPC





Geth Console API

Geth console API

<https://github.com/ethereum/wiki/wiki/JavaScript-API>



Ethereum blockchain related methods

Node's network status

Get/put for local *LevelDB*

P2P messaging using *Whisper*



Management API

Management API

<https://github.com/ethereum/go-ethereum/wiki/Management-API>
<https://github.com/ethereum/go-ethereum/wiki/JavaScript-Consoles>

admin Node management

personal Account management

miner Miner management

txpool Transaction pool

debug Node debugging



Account Keyfiles

Account Keyfiles

On account creation a key file gets added to the geth client

- <DATADIR>/keystore
- UTC--<created_at UTC ISO8601>--<address hex>

```
keystore  
UTC--2017-01-14T11:57:39.276261600Z--8b4180f38fb42f42c4d6b0cbcc3e01b8a5f521
```

Ok to copy keystore file to a different instance of client

- Switching clients e.g., Mac to PC *or* Geth to Wallet

Keyfiles Practices



Account Keyfiles

Keyfiles Practices

Good practices:

- Periodically backup all your key files
- Use strong passwords; DO NOT forget the passwords.
- If managing passwords in a file make sure they are hidden and accessible to only you



Miner

Miner management

geth CLI Miner options

--mine

--minerthreads

--etherbase

--gasprice "value"

--extradata "somedata"



miner.start(num_threads)
miner.stop()



miner.setEtherbase(addr)



miner.setGasPrice(num)



miner.setExtra(data)



Truffle and Local Development

Truffle Framework



- ❖ Local Environment
- ❖ Can be checked into GitHub
- ❖ Unit Testing
 - ❖ Using JavaScript and Solidity
- ❖ Scripted Deployment & Migrations
- ❖ Command Line Based



Truffle Commands

- ❖ “truffle init”
 - ❖ Initialized an empty directory as truffle project
 - ❖ Will download the “truffle-init” repository which contains some smart contracts
 - ❖ And runs “npm install” which looks into package.json

- ❖ “truffle develop”
 - ❖ Opens developer console
 - ❖ From there are “test” or “migration” possible
 - ❖ On the Developer-Blockchain



Truffle Commands

- ❖ “truffle migrate” or “truffle migrate --network abc”
 - ❖ Looks in the migrations folder for migration-scripts
 - ❖ Deploys to the “default” network or to “abc” network
 - ❖ Based on truffle.js config file in root directory
- ❖ “truffle test”
 - ❖ Runs the tests according to the “test/” directory
- ❖ “truffle compile”
 - ❖ Compiles the contracts
 - ❖ Generates so called contract artifacts – contain ABI, and meta-information
- ❖ That’s all the magic you really need.
- ❖ Where is the HTML part?

Truffle – HTML5



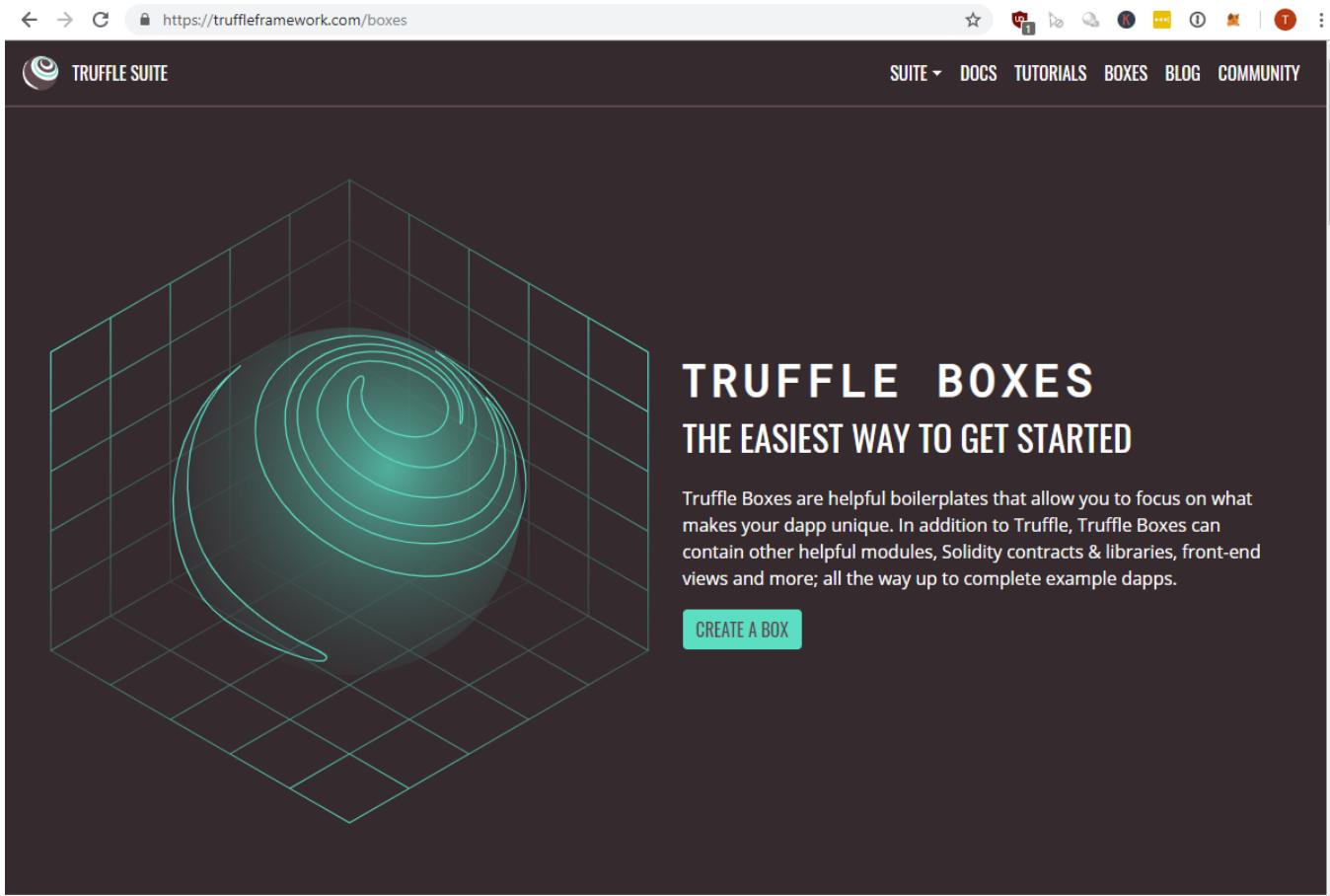
- ❖ Truffle *itself* is really *only* for Smart Contract Development
- ❖ Truffle is a npm package
- ❖ Any other npm package can be integrated.
- ❖ truffle-contract is an abstraction of web3.js
 - ❖ Can be used almost like web3.js
 - ❖ Uses Promises
 - ❖ And used in HTML/JS

Truffle - HTML5



- ❖ Webpack or React or Vue or any other Web-Dev Framework can be used in combination with Truffle and Truffle-Contract
- ❖ Truffle has “ready to go” Templates
 - ❖ Calls it “Truffle Boxes”
- ❖ We can install a “ready to go” box via “truffle unbox”
 - ❖ A list is here: <https://truffleframework.com/boxes>

Truffle Boxes



The screenshot shows a web browser window for the URL <https://truffleframework.com/boxes>. The page has a dark background. On the left, there is a large, semi-transparent 3D wireframe cube containing a glowing teal sphere that rotates slowly. At the top left is the "TRUFFLE SUITE" logo. At the top right are navigation links: "SUITE", "DOCS", "TUTORIALS", "BOXES" (which is highlighted in red), "BLOG", and "COMMUNITY". Below the navigation, the main heading reads "TRUFFLE BOXES" in large white letters, followed by "THE EASIEST WAY TO GET STARTED" in smaller white letters. A paragraph of text explains what Truffle Boxes are: "Truffle Boxes are helpful boilerplates that allow you to focus on what makes your dapp unique. In addition to Truffle, Truffle Boxes can contain other helpful modules, Solidity contracts & libraries, front-end views and more; all the way up to complete example dapps." At the bottom left of this section is a green button labeled "CREATE A BOX".



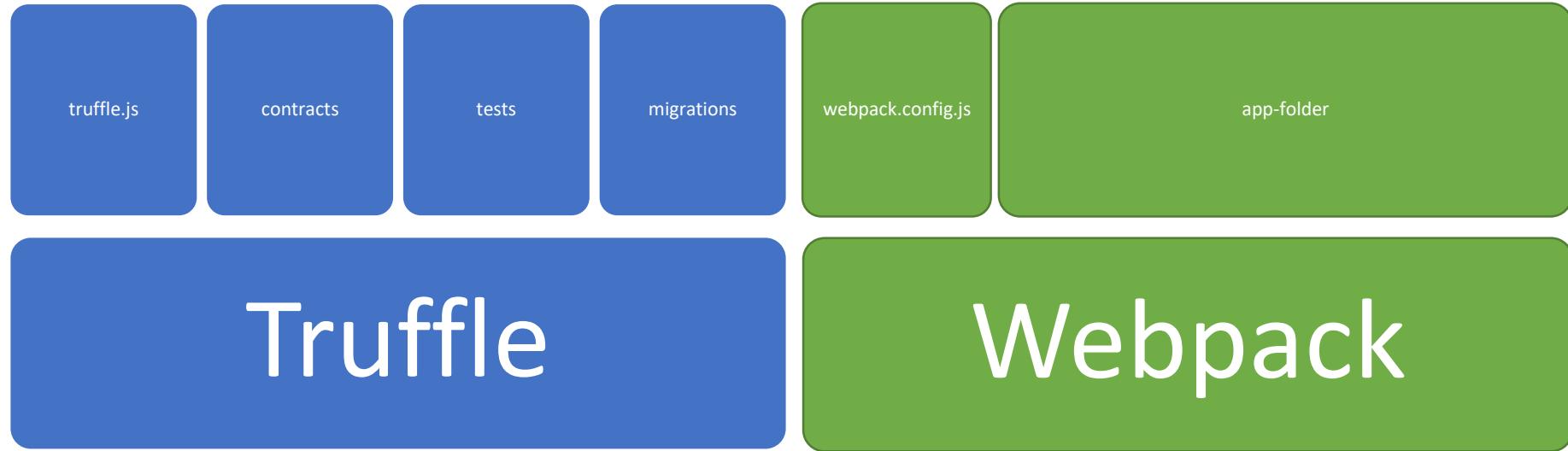
Truffle Boxes

- ❖ Truffle Boxes are most likely community created
- ❖ It's just a repository
- ❖ “truffle unbox” will
 - ❖ Download the repository
 - ❖ Run “npm install” to install all dependencies
- ❖ Virtually a boilerplate box for all modern Frontend Languages



Truffle and Webpack

- ❖ “truffle unbox webpack” stack



Root directory



Truffle

Deployment/Migration

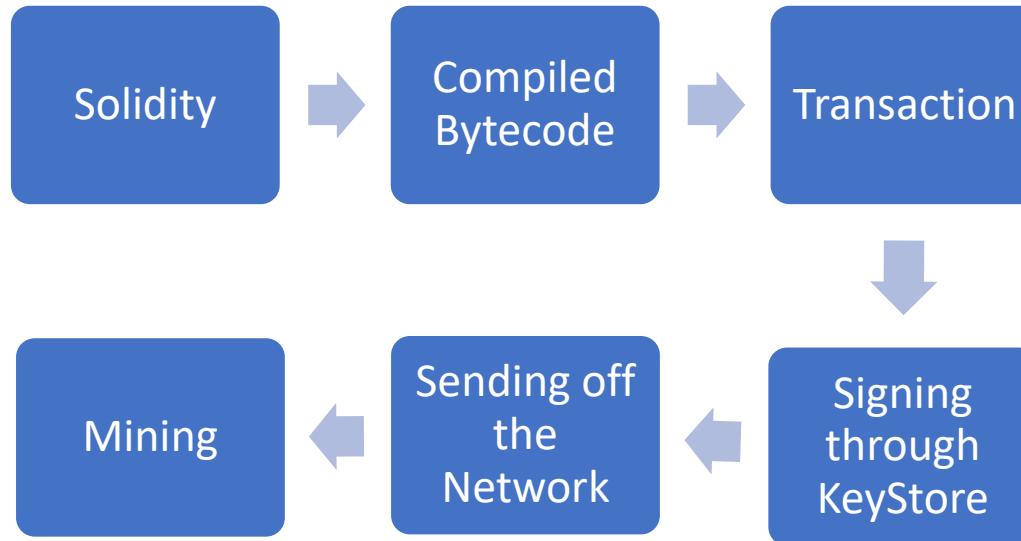


Migrations

- ❖ Totally scriptable
- ❖ More than one network
 - ❖ Maybe a developer blockchain
 - ❖ Plus a Corporate Network
 - ❖ Plus a Main Production Network
- ❖ No “upgradeable” contracts by default
 - ❖ Upgrade or migration plan in place?



Components





Solidity -> Compile

- ❖ Write your code in Solidity
- ❖ /contracts/ folder
- ❖ Truffle takes solc-js for compilation
 - ❖ Usually 1-2 version numbers behind
 - ❖ Comes pre-packaged
- ❖ As of writing: version 0.5.0 is newest, truffle comes with 0.4.24



Compilation -> Bytecode

- ❖ Truffle produces “artifacts”
- ❖ Json-Files
- ❖ Contain the ABI
- ❖ Bytecode
- ❖ Plenty of Meta-Information
- ❖ /build/contracts/ folder



Bytecode -> Transaction

- ❖ The Bytecode needs to be sent off using a transaction
 - ❖ Transactions can only be started using EOAs
 - ❖ EOA is managed by Private Key
-
- ❖ So: where does the Private Key come from for signing the transaction?



KeyStore / Signature

- ❖ KeyStore can be built into a node offering web3 access through JSON RPC
 - ❖ Ganache/Geth for example
 - ❖ web3.eth.accounts array is filled
 - ❖ maybe “personal.unlockAccount(...)”
- ❖ Keys can also be generated using HD Wallet
 - ❖ Hierarchical Deterministic (HD) key creation
 - ❖ A “seed phrase” always leads to the same private keys

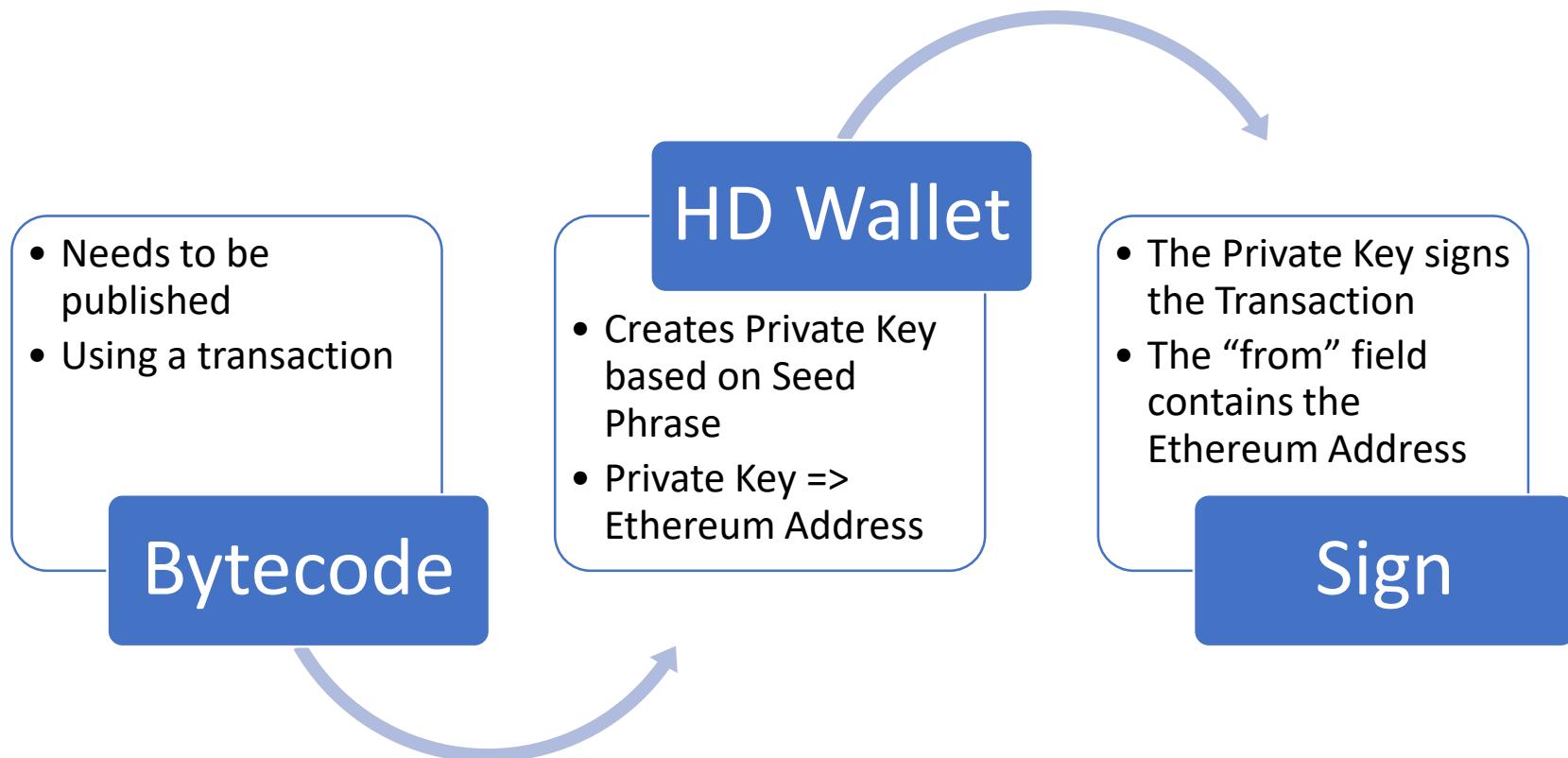


HD Wallet

- ❖ Known as “mnemonic” or “seed phrase” used to initialize your wallet
- ❖ Same 12 words lead to the same private keys
 - ❖ Plural: Seed+Index => key
 - ❖ Seed+1 => Private Key1
 - ❖ Seed+2 => Private Key2
 - ❖ ...
- ❖ Truffle can do that do
 - ❖ Don’t publish your seed-phrase!
 - ❖ Also don’t check it into your repository

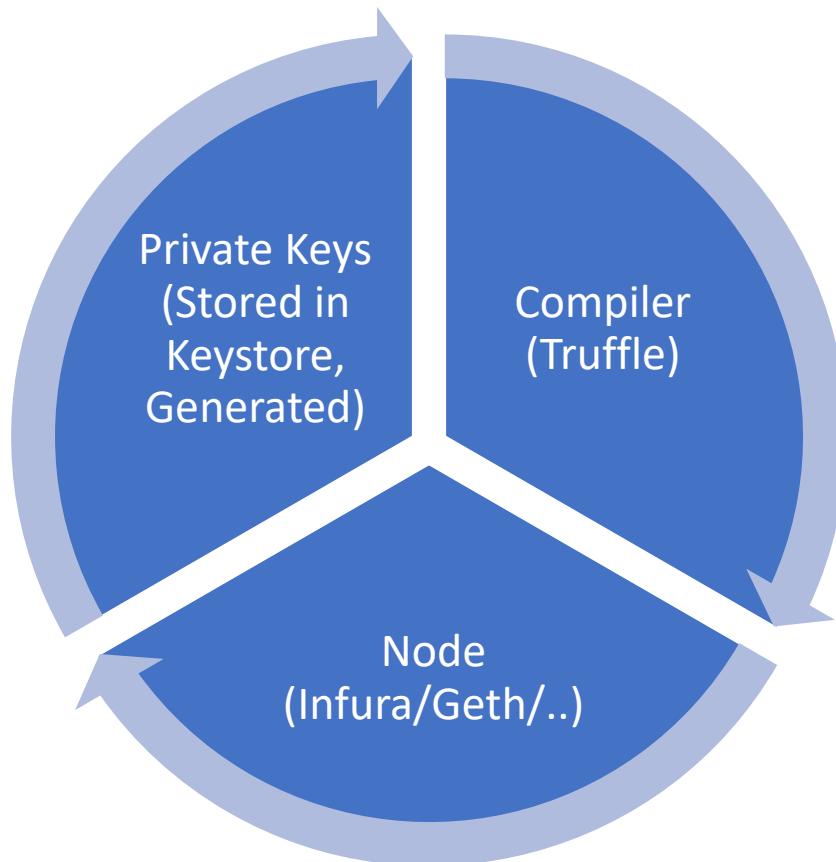


Publish using HD Wallet





Components





Truffle

Unit Testing



Unit Testing

- ❖ Test all aspects of a software
- ❖ A Unit: smallest testable part of an application

- ❖ Smart Contract
 - ❖ That's a function
 - ❖ Write as simple as possible



Testing with Truffle

- ❖ Smart Contract testing
 - ❖ Not the “JavaScript” app
 - ❖ But Tests can be written in JavaScript
- ❖ Clean Room Environment
 - ❖ advanced snapshotting features to ensure your test files don't share state with each other.
 - ❖ re-deploy all of your migrations at the beginning of every test file to ensure you have a fresh set of contracts to test against
- ❖ Ganache or Truffle Develop during normal development and testing recommended (speed)



Tests using JavaScript

- ❖ Mocha testing framework
- ❖ Chai for assertions
- ❖ Structurally, tests very similar to Mocha
 - ❖ If you know Mocha, Use contract() instead of describe()
- ❖ Contract Abstraction via artifacts.require()
- ❖ A web3 instance is available in each test file, configured to the correct provider.
 - ❖ So calling web3.eth.getBalance just works!



Tests using JavaScript

- ❖ Very easy if you know *some* JavaScript
- ❖ Good for continuous integration
- ❖ Good when working in Teams
 - ❖ Avoid Regression Bugs
- ❖ Quite fast when working with Ganache/Truffle Develop



Tests using JavaScript

```
var MetaCoin = artifacts.require("MetaCoin");

contract('MetaCoin', function(accounts) {
  it("should put 10000 MetaCoin in the first account", function() {
    return MetaCoin.deployed().then(function(instance) {
      return instance.getBalance.call(accounts[0]);
    }).then(function(balance) {
      assert.equal(balance.valueOf(), 10000, "10000 wasn't in the first account");
    });
  });
});
```



Tests using Solidity

- ❖ Clean Room Environment
- ❖ Solidity tests shouldn't extend from any contract
 - ❖ makes your tests as minimal as possible
 - ❖ gives you complete control over the contracts you write.
- ❖ Don't use your own assertion library.
 - ❖ Truffle provides an assertion library
- ❖ You should be able to run your Solidity tests against any Ethereum client.



Tests using Solidity

```
import "truffle/Assert.sol";
import "truffle/DeployedAddresses.sol";
import "../contracts/MetaCoin.sol";

contract TestMetacoin {
    function testInitialBalanceUsingDeployedContract() {
        MetaCoin meta = MetaCoin(DeployedAddresses.MetaCoin());

        uint expected = 10000;

        Assert.equal(meta.getBalance(tx.origin), expected, "Owner should have 10000 MetaCoin initially");
    }

    function testInitialBalanceWithNewMetaCoin() {
        MetaCoin meta = new MetaCoin();

        uint expected = 10000;

        Assert.equal(meta.getBalance(tx.origin), expected, "Owner should have 10000 MetaCoin initially");
    }
}
```



Security & Upgradeable Contracts

- ❖ Private Information and Randomness
- ❖ Re-Entrancy
 - ❖ Checks-Effects-Interactions Pattern
- ❖ Gas Limit and Loops
- ❖ Sending and Receiving Ether
- ❖ Callstack Depth
- ❖ tx.origin vs. msg.sender
- ❖ Underflows / Overflows
- ❖ General Recommendations

- ❖ Private Information and Randomness
 - ❖ Everything publicly visible, even local variables and state variables marked private.
 - ❖ random numbers in smart contracts is quite tricky
 - ❖ Randomness on Deterministic Systems hard to achieve
 - ❖ RANDAO
 - ❖ Smart Contract
 - ❖ Uses “many people” to generate randomness

- ❖ Re-Entrancy
 - ❖ The DAO attack
 - ❖ Contract A hands over control to Contract B
 - ❖ Contract B can “call back” to A
 - ❖ When does this become a problem?
 - ❖ If we can withdraw the same amount again
- ❖ Checks-Effects-Interactions Pattern
 - ❖ Update variable before calling external contract
 - ❖ “balance = 0” before “someAddress.call.value(...)(...)" not after
 - ❖ Check for return variables from low-level functions

❖ Gas Limit and Loops

- ❖ Block gas limit limits the maximum runtime of smart contracts
- ❖ Loops with “open end” quickly run out of gas
- ❖ Avoid loops altogether
 - ❖ Unless you really know what you are doing
- ❖ Never use loops on “scaling” variables
 - ❖ Such as arrays that grow with the number of users

❖ Sending and Receiving Ether

- ❖ You can never prevent the reception of ether
 - ❖ Set as coinbase or set as beneficiary on “selfdestruct(...)”
- ❖ Fallback function is called by default (if no function is defined by caller). Transaction fails if no fallback function is called
- ❖ `addr.call.value(x)("")` forwards all gas, letting the called contract potentially call back again. -> re-entrancy
- ❖ Never rely on more than 2300 gas in a fallback function!

- ❖ Callstack Depth
 - ❖ Maximum callstack of 1024
 - ❖ Throws an exception
- ❖ Malicious actors can force Exception
- ❖ Be careful with low-level functions and always check the return value!

- ❖ tx.origin vs. msg.sender
 - ❖ Confusing difference between tx.origin and msg.sender
- ❖ Consider a Contract A and that contract calls another Contract B
 - ❖ In contract A tx.origin and msg.sender are the same (address of EOA)
 - ❖ In contract B tx.origin is the address of the EOA and msg.sender is the address of the calling contract A

- ❖ Underflows / Overflows
- ❖ $\text{uint8}(255) + \text{uint8}(1) == 0$
- ❖ $\text{uint8}(0) - \text{uint8}(1) == 255$

- ❖ No warning
- ❖ Try to use require to limit the size of inputs to a reasonable range

❖ General Recommendations

- ❖ Take Warnings Seriously
- ❖ Restrict the Amount of Ether
- ❖ Keep it Small and Modular
- ❖ Use the Checks-Effects-Interactions Pattern
- ❖ Include a Fail-Safe Mode
- ❖ Ask for Peer Review



Upgrades: Why Upgrade?

- ❖ An “unchangeable” ledger
 - ❖ Things can only be added
 - ❖ Immutability!
- ❖ Smart Contracts?
 - ❖ unchangeable logic
 - ❖ *immutable*
 - ❖ Looking at the logic now (which doesn’t do anything bad) should guarantee that it doesn’t do anything bad anytime in the future
 - ❖ Nice, but not *convenient* especially when bugs happen where contracts manage several M\$.



Ways around that

- ❖ Immutability stays
 - ❖ Have a migration path
 - ❖ Maybe means to release a new smart contract on a *new address* and then move all data over
 - ❖ Also tell users about the new address
 - ❖ Pause smart contracts?
- ❖ Make your Smart Contracts as *simple as possible* and *test everything!*



Ways around that

- ❖ Separate Logic from Data
 - ❖ Proxy-Delegate Pattern
 - ❖ Eternal Storage Pattern
 - ❖ <https://zeppelinos.org/>
-
- ❖ Governance?
 - ❖ Who is allowed to do that? Does every user have to agree to the upgrade? Or is there an authority?



Proxy-Delegate Pattern

- ❖ Makes use of “delegatecall” low level function
- ❖ One Smart Contract as entry point
- ❖ Connects to another smart contract via an Address
- ❖ Relays the initial request

- ❖ More you don't have to know
- ❖ A first functional version of this pattern was introduced in 2016.
- ❖ Removes the basic promise – Governance question:
 - ❖ Would you – as end-user – entrust all your savings a piece of code where the logic can be changed by a person?



Eternal Storage Pattern

- ❖ Idea: Keep contract storage after a smart contract upgrade.
- ❖ Moving storage is the most expensive operation on Ethereum
- ❖ slightly more complex syntax for storing data.

- ❖ Users might be able to choose which Version of Smart Contract they want to use
 - ❖ Let Users “upgrade” manually

- ❖ RocketPool uses this
 - ❖ Sidenote: Proof of Stake infrastructure service and pool

ZeppelinOS



- ❖ Develop, deploy and operate any smart contract project securely
 - ❖ “ZeppelinOS is a development platform designed specifically for smart contracts projects. It allows for seamless upgrades, and provides economic incentives to create a healthy ecosystem of secure applications.”
- ❖ Have upgradeable smart contracts
 - ❖ which follows the immutability rules guaranteed by the Ethereum blockchain
- ❖ Try to achieve decentralized governance
 - ❖ Not one person alone can upgrade a smart contract



Smart Contracts Best Practices



Best Practices

- ❖ The Ecosystem is very young
 - ❖ Many have made costly mistakes
- ❖ Good writeups of security best practices are available
 - ❖ ConsenSys does one at
<https://consensys.github.io/smart-contract-best-practices/>
- ❖ Let's discuss this in more detail!
 - ❖ Don't make costly mistakes!



Prepare for failure

- ❖ There is no coding without errors
 - ❖ It's a wish to code error-free, so, how can we address this?
- ❖ Pause contracts when errors are discovered
 - ❖ Or have sanity checks
 - ❖ “Circuit breaker”
- ❖ Manage amount of money at risk
 - ❖ Rate limiting, maximum usage
- ❖ Have an “upgrade path” for bugfixes and improvements



Rollout strategy

- ❖ Test test test
 - ❖ Unit Testing
 - ❖ Alpha Testing
 - ❖ Beta Testing
- ❖ Bug bounties
- ❖ Rollout in phases
 - ❖ slowly, if possible



Simple and stupid

- ❖ The simpler the contract the better
- ❖ Modularize code, keep functions small
- ❖ Use libraries and tested code if possible
 - ❖ Don't try to re-invent the wheel
 - ❖ EthPM for example (truffle and zeppelinOS can do that)
- ❖ Clarity over performance
- ❖ **Only use the blockchain where the blockchain is needed!**



Stay up to date

- ❖ Checkout Niche News
 - ❖ Ethereum in a Week newsletter
 - ❖ Consensys newsletter
 - ❖ Reddit
 - ❖ ...
- ❖ New Bugs discovered in the industry?
 - ❖ Make sure your contracts are safe then!
- ❖ Upgrade libraries and tooling
- ❖ Adopt new security best practices if possible



Blockchain Specifics

- ❖ Attention at external contract calls
 - ❖ Might execute malicious code
 - ❖ Can change the control flow
- ❖ Public functions can be called by anyone, so prepare for that
- ❖ Private variables can be viewed by anyone easily
- ❖ Keep gas costs low and don't forget the block gas limit!

Simplicity vs. Complexity



- ❖ Ideally for a software developer:
 - ❖ Modular
 - ❖ code reuse
 - ❖ Upgradeable
- ❖ Blockchain Security sometimes not aligned with these ideals
 - ❖ Rigid vs. Upgradable
 - ❖ Monolithic vs. Modular
 - ❖ Duplication vs. Reuse



Security Best Practices

- ❖ Be careful with external calls
- ❖ Mark untrusted contracts in your code clearly
- ❖ Checks-effects-interactions pattern
 - ❖ Avoid state changes after external calls
- ❖ Pull over Push Method
 - ❖ Also called “withdraw” method
 - ❖ Let a user “pull” his money out, rather than just sending it straight away
 - ❖ Can lead to dangerous DoS attacks
- ❖ Be careful with checking invariants concerning the balance
 - ❖ It is possible to forcefully send ether to a smart contract
 - ❖ Assert might break functionality completely



Assert and Require

- ❖ Use Assert to check for invariants
 - ❖ Such as the cost per token
 - ❖ Use it as “circuit breaker”
 - ❖ Pause the contract if things fail
- ❖ Use Require to do input validation
- ❖ Keep this pattern to help with formal validation
 - ❖ “assert” should *never* fail if your code is correct



General Best Practices

- ❖ Use only assertions in modifiers
 - ❖ Code is executed before the function body
 - ❖ Check for invariants only, don't do input validation
 - ❖ Use modifiers only for error handling!
- ❖ Integer rounds always off!
 - ❖ Cuts off the decimal
 - ❖ $5/2 = 2$



General Best Practices

- ❖ Keep fallback functions simple
 - ❖ Don't assume that there is more than 2300 gas available
 - ❖ That is usually just enough to emit an event
- ❖ Check the data-length in fallback functions
 - ❖ Are called when no other function matches
 - ❖ Throw an exception, or caller will not know they mistakenly called the fallback function
- ❖ Make your version pragma specific
 - ❖ ^0.4.25 can also be used for the nightly compiler



General Best Practices

- ❖ Be aware that the block timestamp can be manipulated by a miner
- ❖ Understand the inheritance graph if you use multiple inheritance
- ❖ Read https://consensys.github.io/smart-contract-best-practices/known_attacks/ and understand known attacks



Blockchains of Storage



Blockchains for Data Storage

- ❖ Ethereum is not good at storing large amounts of data
 - ❖ Storage cost is extreme
 - ❖ \$0.076/KB or \$76,000/GB (Feb 2016)
 - ❖ More than \$10M/GB in Nov 2017
 - ❖ Not meant for storing large data (pictures, videos...)
 - ❖ Data is redundantly distributed across all nodes
- ❖ Other Blockchain solutions are better for this
 - ❖ IPFS
 - ❖ SWARM



IPFS and Swarm

- ❖ Peer to Peer
- ❖ File Content = Hash
- ❖ low-latency retrieval reliable, fault-tolerant operation, resistant to node's disconnections, intermittent availability
- ❖ zero-downtime
- ❖ censorship-resistant
- ❖ content-addressed block storage model, with content-addressed hyperlinks
- ❖ Beta (or even alpha)



- ❖ InterPlanetaryFileSystem
- ❖ Generic DHT
- ❖ proof of retrievability as part of mining
 - ❖ FileCoin
- ❖ already serves as a working solution for real-world businesses
- ❖ is supported by an enthusiastic user-base



Swarm

- ❖ swarm's core storage component as an immutable content addressed chunkstore rather than a generic DHT
- ❖ swarm has deep integration with the Ethereum blockchain and the incentive system benefits from both smart contracts as well as the semi-stable peerpool
- ❖ Similar to BitTorrent

Architecture



Ethereum

Host a “pointer”
to data

Be the
lightweight
database (lookup
table?)

IPFS/Swarm

Host large files

Host the
“website”



Compilation of Smart Contracts



Compilation

- ❖ Usually done within a Development Tool
 - ❖ Abstracted away from developer
 - ❖ Remix -> “deploy” will magically take the binary and put it in a transaction
 - ❖ Truffle -> “compile” or “migrate” will magically create the artifact and deploy it
- ❖ What really happens during compilation?



Different Compilers

- ❖ There are different solidity compilers and versions
 - ❖ Old *compiled* solidity code is not “insecure”
 - ❖ New language features makes it harder to write buggy code
- ❖ NPM package: *solcjs*
 - ❖ The *solc-js* project is derived from the C++ *solc*
 - ❖ Uses the same source
- ❖ *Solc* is the c++ project
 - ❖ The commandline options of *solcjs* are not compatible with *solc* and tools (such as *geth*) expecting the behaviour of *solc* will not work with *solcjs*.



Idea to reality

- ❖ 2016: Geth can compile Solidity code
 - ❖ `web3.eth.compile.solidity(source_string, callback_func)`
- ❖ 2017: Compiler Removed, but can be plugged in
 - ❖ But only solc (needs compilation)
 - ❖ Not solcjs (can't just do "npm install solcjs")
- ❖ 2018: Nobody uses geth to compile solidity anymore, better use a standalone compiler



When using a compiler

- ❖ Large Project might need specific compiler version
 - ❖ Truffle comes “pre-packaged”
- ❖ Specific compiler options
- ❖ Maybe a custom compiler for low-level optimization?
 - ❖ Better not – “don’t change the library”: Upgrade-hell



Compiler Output

- ❖ ABI Array

- ❖ Compiler automatically creates Application Binary Interface
- ❖ Needed for contract deployment
- ❖ Needed for invoking contracts

- ❖ Bytecode / EVM Code / EWASM

- ❖ Code that's deployed on the blockchain
- ❖ Transaction field “data”



Important?

- ❖ Is it important to know how to compile manually?
 - ❖ No
- ❖ Unless...
 - ❖ You want to do a deep-dive into the intrinsic details of the compiler and language
 - ❖ You need specific compiler options
 - ❖ Truffle Contract Management won't work for you
- ❖ Better...
 - ❖ Understand the bytecode / EVM Assembly or EWASM
 - ❖ Optimize from there



Blockchain Use Cases

Just a Few Use Cases



- ❖ Background checks: education credentials, criminal records
- ❖ Secure document storage: home deed, auto title
- ❖ Birth registries
- ❖ Land registries
- ❖ Financial services: securities clearing, syndicated loans
- ❖ Global supply chain: automotive recalls and counterfeit airbags
- ❖ Healthcare: EMRs, insurance claims, genome research
- ❖ Airlines: registration, re-booking, vouchers, loyalty
- ❖ Tokenized economy: Tech Coworking space 1 token = 1 seat
- ❖ Payment channels: Starbucks or for bandwidth consumption



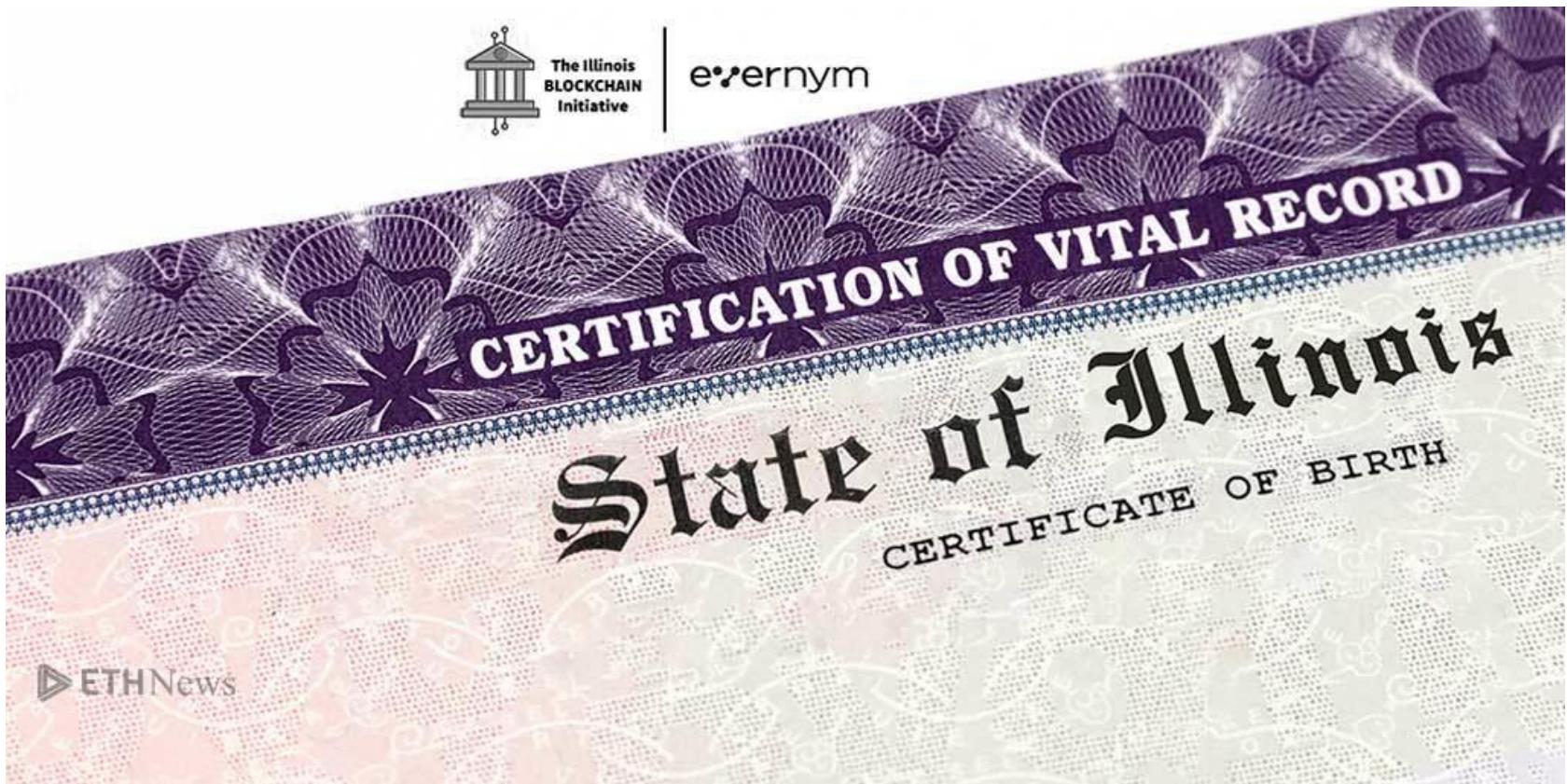
Background Checks

MIT Digital Certificates (diplomas); Criminal Records

The image displays a composite of three digital documents:

- Digital Diploma (Left):** A mobile phone screen shows a digital diploma for "Master of Finance" from "MASSACHUSETTS INSTITUTE OF TECHNOLOGY". The student is listed as "Sample Student". The degree is "MASTER OF FINANCE". The document includes a signature of "R. Gregory Meyer" and "SECRETARY" above "Rafael Reif" and "PRESIDENT". It also features the official seal of the Massachusetts Institute of Technology.
- Background Check Progress (Top Right):** A vertical progress bar titled "Step 1 of 5" through "Step 5 of 5" with status "DONE" for each step: "Computing local hash", "Fetching remote hash", "Comparing local and remote hashes", "Checking Merkle root", and "Checking receipt".
- Digital Resume (Bottom Right):** A resume for "Loryann M. Hoofnagle" featuring:
 - Contact Information:** 830 Harmony Street, Simi Valley, CA 94588
 - Career Overview:** I have 10 years of experience in web design with a range of clients including small private companies, medium-sized e-commerce shops, and large online magazines. My main focus is on design, usability and SEO optimization.
 - Details:** home: 805-960-6116, work: 805-990-1747, email: demo@opresume.com, website: http://OPResume.com/demoA QR code is also present on the right side of the resume.

Birth Registry





Global Supply Chain

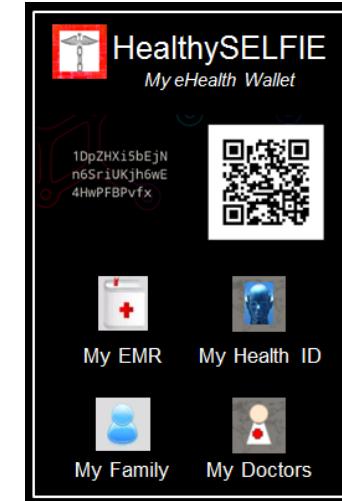
- ❖ Automotive Industry Recalls and Counterfeit Airbags
- ❖ Business case: 30% global airbags sold and installed are counterfeit
- ❖ Solution: Single shared process for airbag registration and lookup



Healthcare



- ❖ Electronic Medical Records (EMRs)
 - ❖ Digital health wallet
 - ❖ Identity credentials + EMR + health insurance + payment information
- ❖ Health insurance claims
 - ❖ Automated claims billing, validation, payment, and settlement
 - ❖ Multi-party value chain: patient, service provider, billing agent, insurance company, payor, government, collections
- ❖ Genomic research
 - ❖ Files too large (20-40 Gb) for centralized research repositories
 - ❖ Need secure validated access

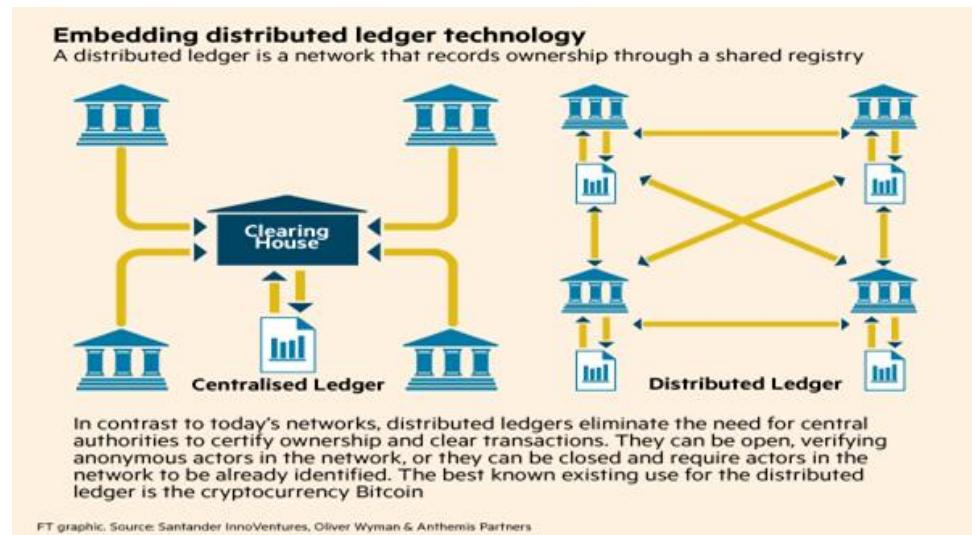


Digital health wallet

Financial Services

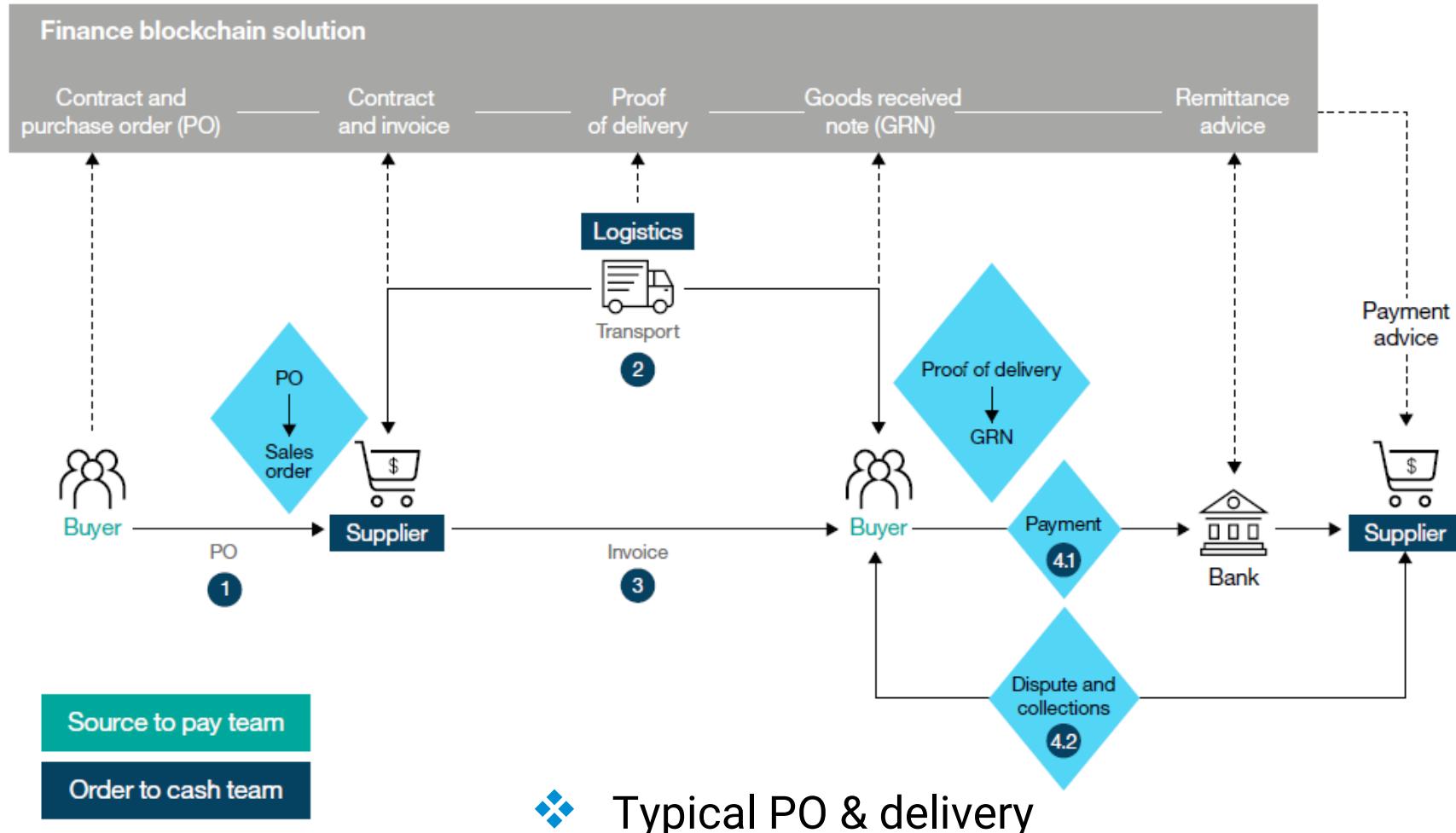


- ❖ Clearing and Settlement
- ❖ Issuance, Ownership, and Transfer of Financial Instruments
- ❖ Servicing of Instruments
- ❖ Payments and Remittance
- ❖ KYC/AML Compliance
- ❖ Regulatory Reporting
- ❖ Audit and QA
- ❖ Back-office Reconciliation





Financial Services - PO



Airlines



- ❖ Registration System (like SABRE)
- ❖ Cross-airline flight re-booking
- ❖ Loyalty programs
- ❖ Flight cancellation smart contract vouchers



Smart contracts automatically issue vouchers to customers when flights are delayed or cancelled



Blockchain and Home Purchases

- ❖ Save by eliminating transaction fees for buyer/seller agent, banks and any other intermediaries
- ❖ Minimize turnaround time
- ❖ Eliminate escrow as Blockchain is only a source of trust
- ❖ Eliminate manual process for requesting mortgages

Blockchain and Political Systems



- ❖ Elections and Voting
- ❖ Liquid Democracy / Delegative Democracy
 - ❖ Voters can transitively delegate their votes
- ❖ Futarchy and Voting Prediction Markets
 - ❖ 2-tier voting for project area and approach
- ❖ Participatory Budgeting
 - ❖ Residents collectively decide how to spend their local government's budget
- ❖ Self-directed Public Finance
 - ❖ \$500/\$1000 personal bond offerings
- ❖ Virtual Democracy
 - ❖ Instant elections among machine learning models of real voters to address the challenge of ethical decision making



Blockchain Adoption

One of the fastest-moving technology adoptions



Blockchain Adoption

- ❖ Blockchain (distributed ledger technology) is being considered by more than half of the world's big corporations, according to a Juniper market research survey released Jul 2017
 - ❖ 57 percent of large corporations – defined as any company with more than 20,000 employees – were either actively considering or in the process of deploying blockchain
 - ❖ Two-thirds of companies surveyed by Juniper said that they expected the technology to be integrated into their systems by the end of 2018
- ❖ IDC: \$2.1 billion estimated global blockchain spend 2018

The Future of Blockchain



Transforming Society

- ❖ Blockchain technology is bringing us the Internet of value: a new platform to reshape the world of business
- ❖ It transcends all physical and geographical barriers and uses math and cryptography to enable transactions globally
- ❖ The uniqueness of blockchain lies in its capacity to store and retain person-to-person transactional history, so that chances of fraud, hacking, and third-party interference are greatly reduced

Many Blockchains/Crypto currencies



- ❖ It's easy to create your own, and there are many.
-



- ❖ Each is separate and runs its own blockchain
- ❖ The value transferred in each blockchain is primarily in its own cryptocurrency



Parallels to the Internet

Blockchains today have been likened to the Internet in 90s.

- ❖ Only faster growing WW investment occurring!
- ❖ Touching a larger scope of business and society
- ❖ Exploiting Web 3.0 with unlimited uses (IoT)

History doesn't repeat, but it rhymes: We expect similar...but

- ❖ Faster path to maturity – continued expansion
- ❖ Wider – Faster Adoption curve
- ❖ Evolution of protocol and applications touching the lives of people previously unreached by the Internet



Parallels to the Internet

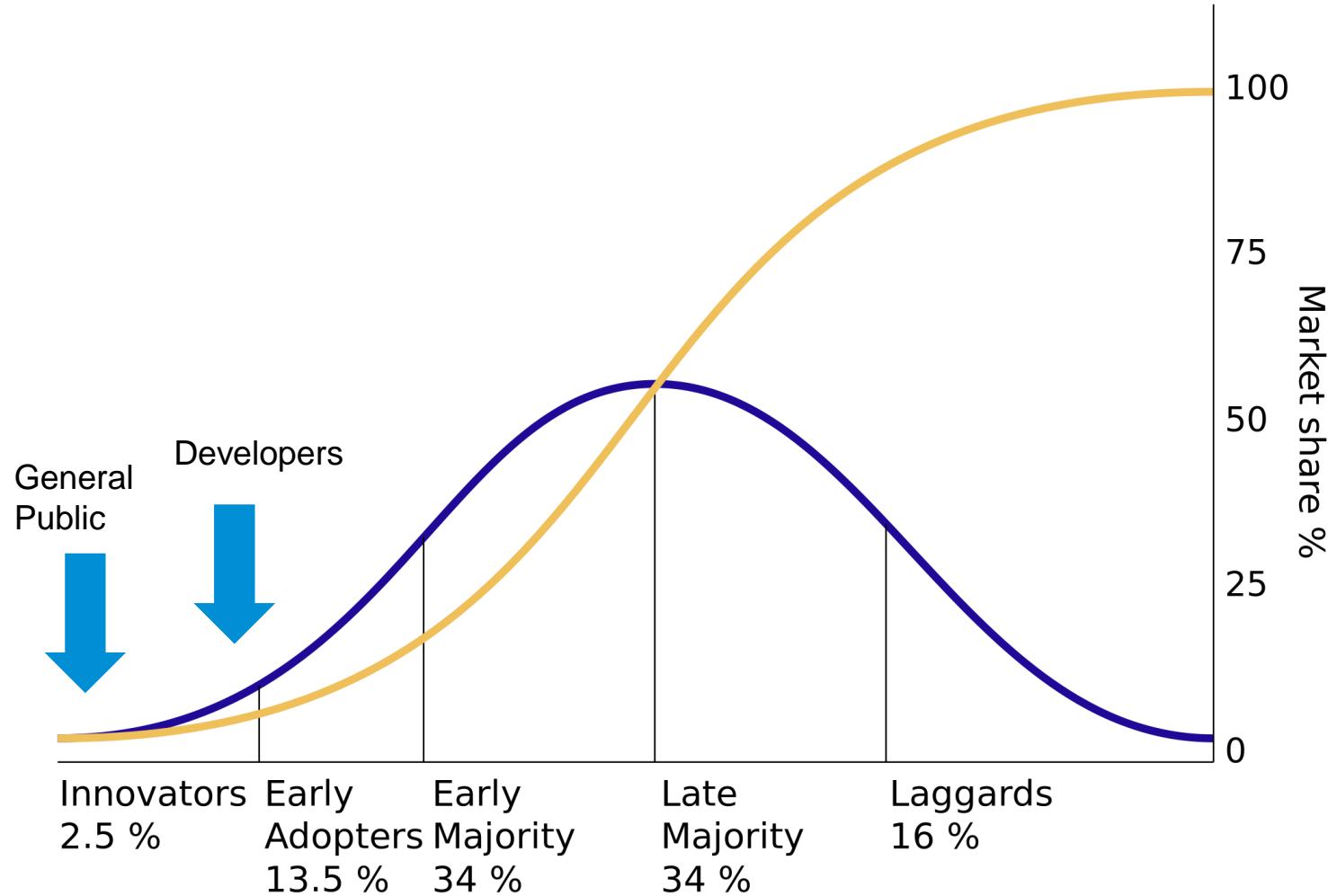
Just as the internet 1.0 revolutionized access to information, Blockchain is doing the same to multiple industrial verticals:

- ❖ Finance and Commerce first
 - ❖ It's what Blockchain's purpose is
 - ❖ It's where the money is
 - ❖ Greatest opportunity to reduce business cost
- ❖ Non-finance uses
 - ❖ Specialist Blockchains dedicated to one task
 - ❖ Typically tied to a cryptocurrency
 - ❖ Generalist Blockchains to be used as a 'platform'

Blockchain Technology is moving very fast – still a lot to learn and new opportunities on many levels, industries, services, and trade!!



Where Are We Now?

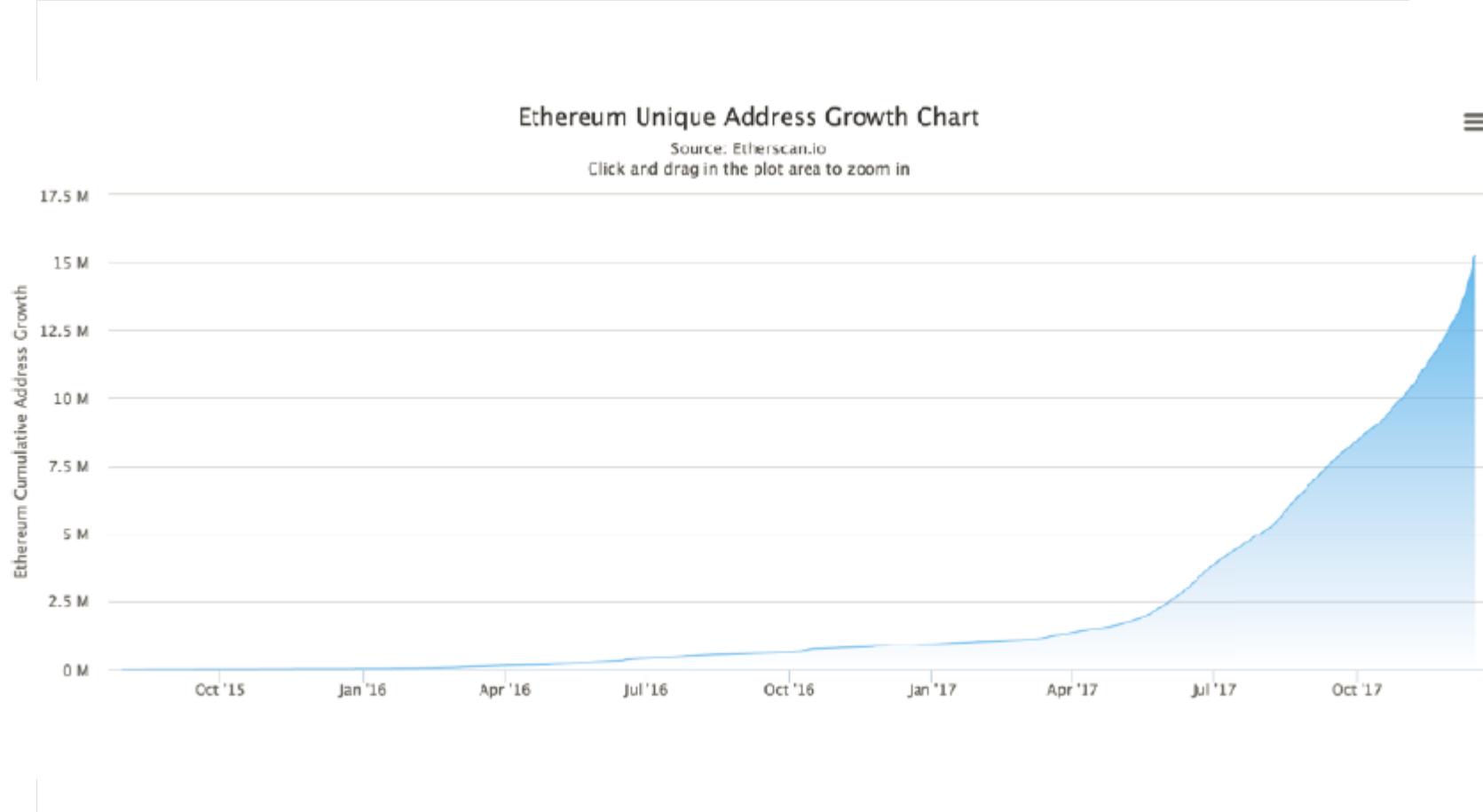




Investment Into the Sector

- ❖ Reid Hoffman (LinkedIn) Invested US\$20M in Blockstream
- ❖ Sir Richard Branson backed BitPay (Exchange) in a US\$30 Million
- ❖ Circle (Exchange) raised US\$50 Million - led by Goldman Sachs
- ❖ NYSE led a US\$75 Million Investment in Coinbase (Exchange)
- ❖ Large BaaS (Blockchain as a Service) investments by IBM, SAP, Cisco, AWS (2017 -2018)
- ❖ Mercedes Benz invests \$110 million to explore blockchain use cases
- ❖ Blockchain investments are extending to the underdeveloped countries and underbanked areas of the world quickly, providing a way to do personal business, as well as commerce

Growth in CryptoAddresses



Wall Street Blockchain Investment Growing



Three blockchain startups selected for Barclays Accelerator, with one aiming to provide blockchain solutions for the insurance industry



Citi wants “to [accelerate] emerging technologies that have the potential to transform financial services experiences for Citi’s customers”



UBS is set to open a London-based research lab to explore the application of blockchain technology in the financial services industry

Sources: CoinDesk, Bank Innovation



Web 3.0



Internet Technology Progression

Internet 1.0
Static Information
WW Explosion of Information

Yahoo - A Guide to WWW

[What's New] [What's Cool] [What's Popular] [Stats] [A Random Link]

[Top] [Up] [Search] [Mail] [Add] [Help]

- * [Art \(469\)](#)
- * [Business \(6426\)](#)
- * [Computers \(2609\)](#)
- * [Economy \(43\)](#)
- * [Education \(1487\)](#)
- * [Entertainment \(6199\)](#)
- * [Environment and Nature \(193\)](#)
- * [Events \(33\)](#)
- * [Government \(1031\)](#)
- * [Health \(367\)](#)
- * [Humanities \(163\)](#)
- * [Law \(163\)](#)
- * [News \(185\)](#)
- * [Politics \(143\)](#)
- * [Reference \(474\)](#)
- * [Regional Information \(2606\)](#)
- * [Science \(2634\)](#)
- * [Social Science \(93\)](#)
- * [Society and Culture \(643\)](#)

... Removed by Censor v2.0 by Estral Barberights --> 23836 entries in Yahoo!

Internet 2.0 Social Engagement
Twitter, Facebook, New Voice of People Worldwide

reface.me { because people read Status Updates, not books }
<http://reface.me/news/become-a-facebook-beta-tester/>

Get a Facebook Sneak Peek

Apply to be a beta tester and get the first look at upcoming Facebook products.

Become a Facebook Beta Tester

reface.me

Facebook is looking for beta testers of their upcoming Questions feature. Apply now!

30 minutes ago · Comment · Like · Share · Promote

Jello Feesh I have a question: Is the "like" a comment made on a comment new? or did I just notice that? lol
22 minutes ago · Like · 1 person · Delete · Flag

Jello Feeshthis-----AAA
21 minutes ago · Like · Delete · Flag

Jessie Black i think that is new! neat.
8 minutes ago · Like · Delete · Flag

reface.me { because people read Status Updates, not books } It most certainly is! Good catch, Jello!
2 seconds ago · Like · Delete

Write a comment...

Internet 3.0 The Internet of Value. Communication, commerce, and participation at all levels. i.e. The Internet of Things (IoT)



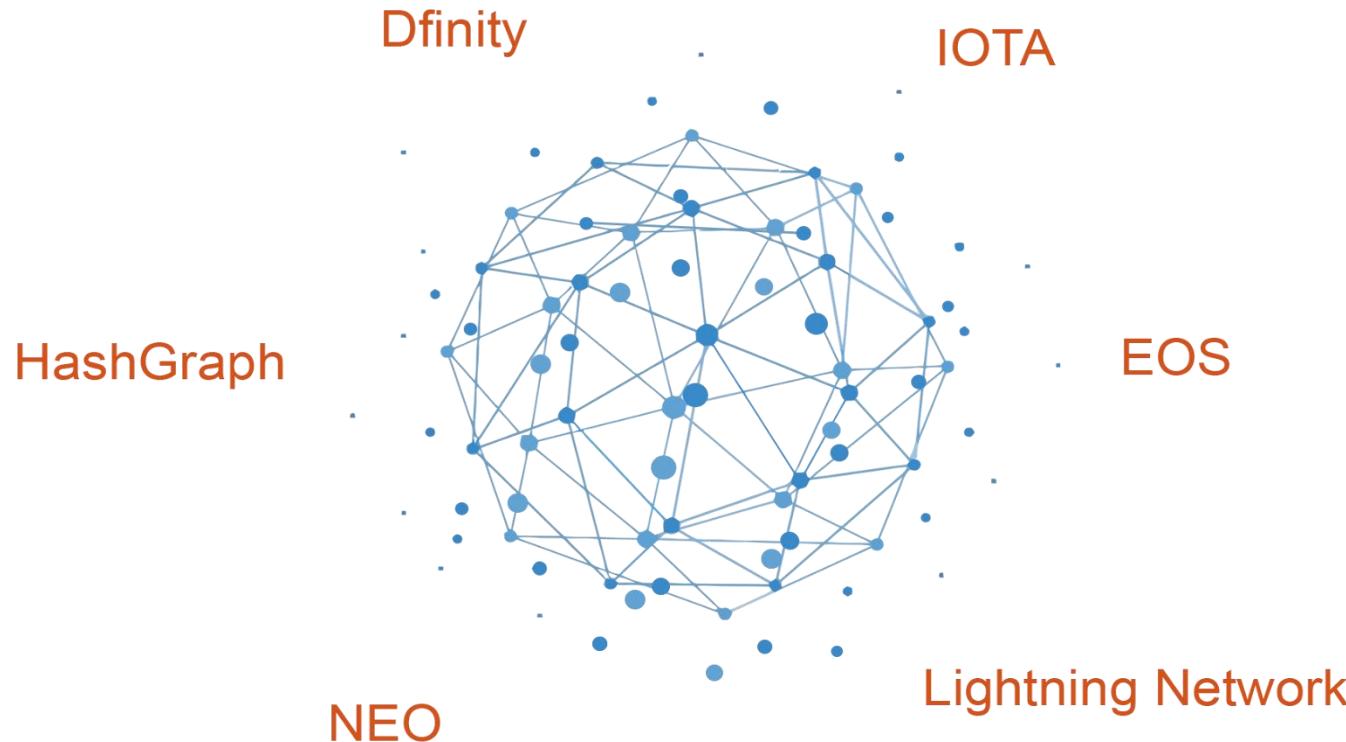
A network of decentralized markets and communities. Create, operate, and govern. Powered by Ethereum, Aragon, and IPFS.



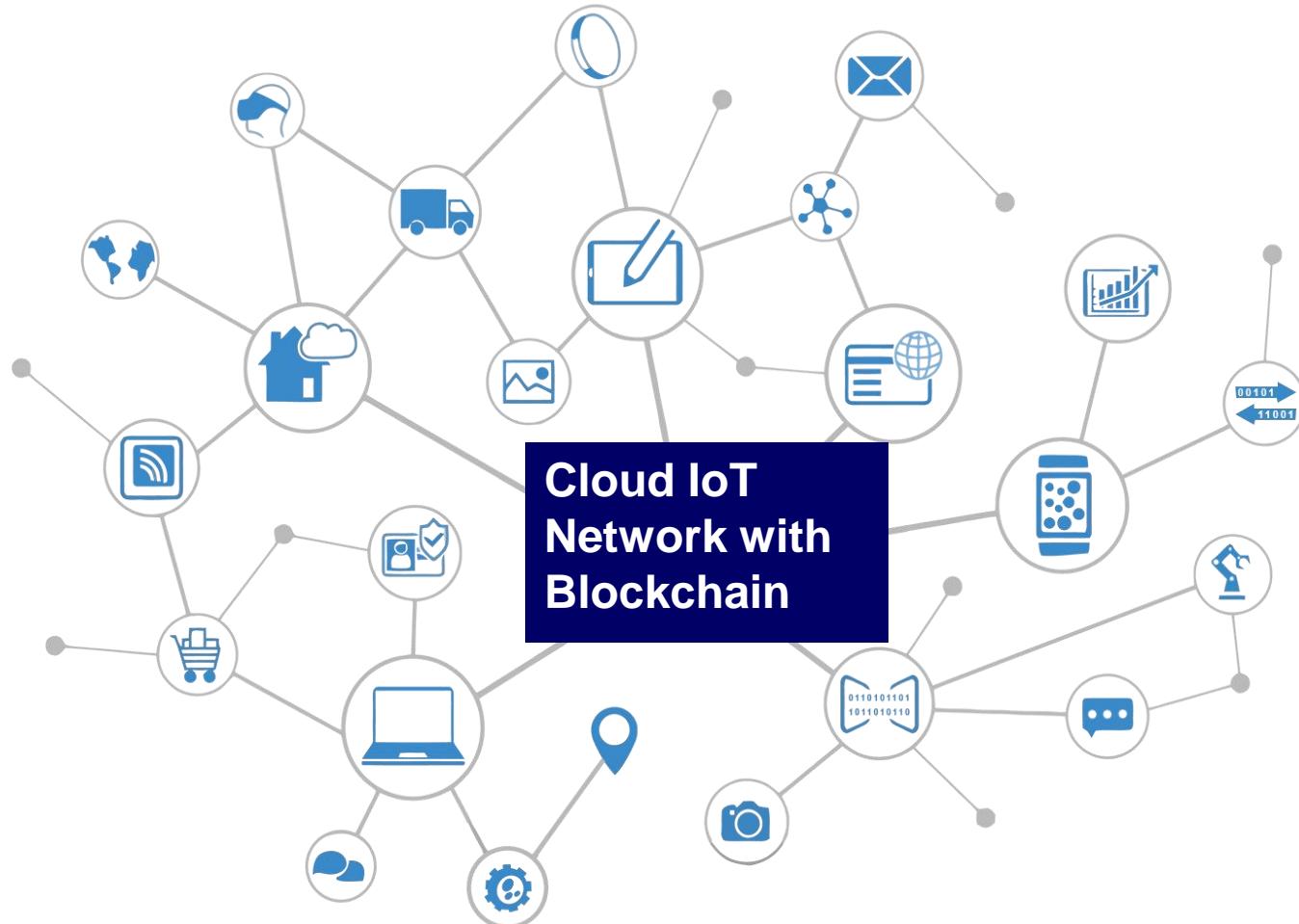


Blockchain – New 3.0 Solutions

Many new Blockchain networks are evolving to provide scalability and solve many of the current problems. New protocols such as Lightning's Off-Chain solution, and IOTA's parallel transactions are just two of the schemes being developed and tested.



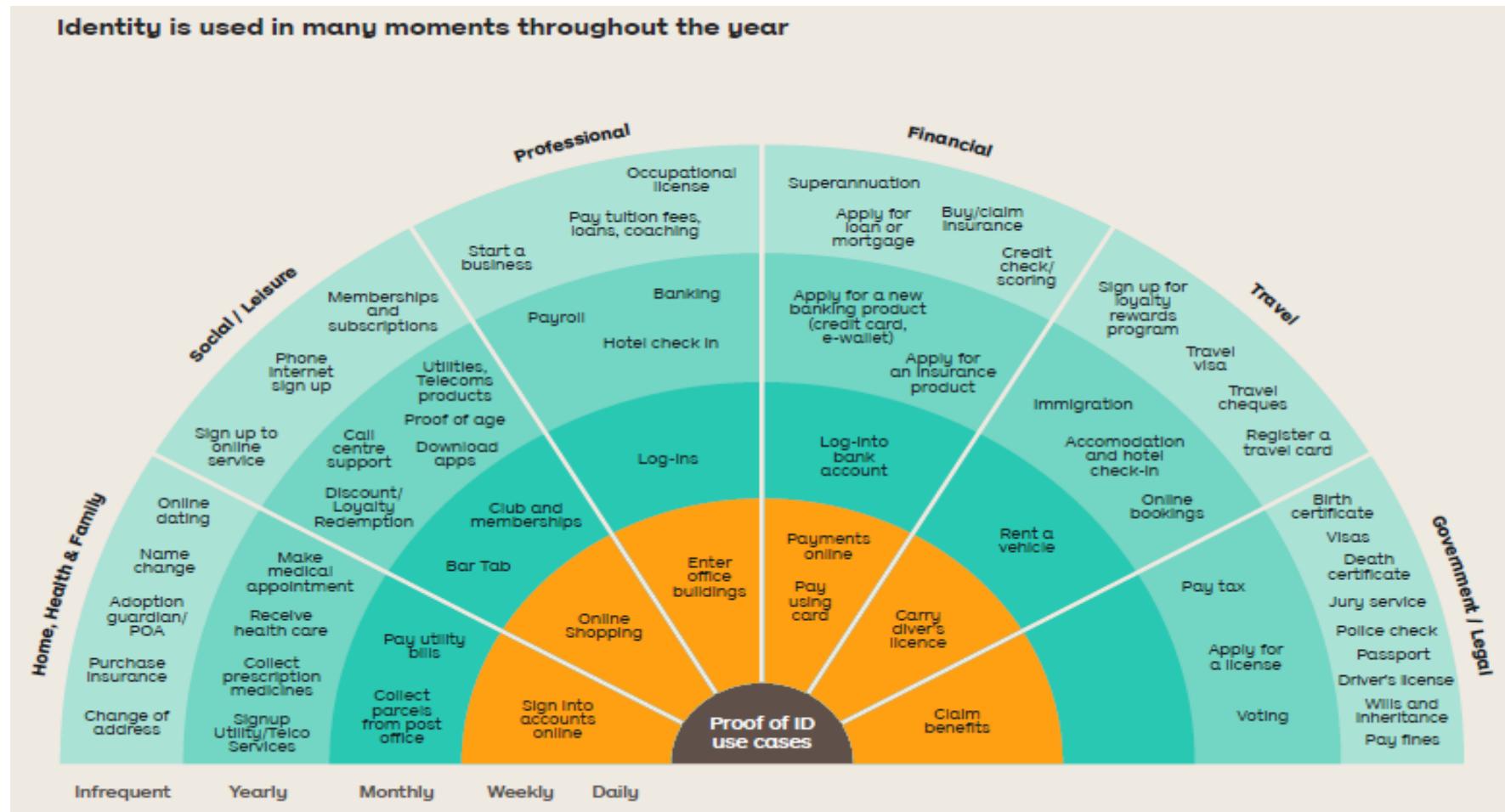
Blockchain – Future Internet 3.0



Digital Identity Touch Points using Blockchain



Identity is used in many moments throughout the year





Blockchain Implementation



Top 5 Blockchain Platform Features

Summary of Features of top 5 Blockchain Platforms for Enterprises

	Ethereum	Hyperledger Fabric	R3 Corda	Ripple	Quorum
Industry-focus	Cross-industry	Cross-industry	Financial Services	Financial Services	Cross-industry
Governance	Ethereum developers	Linux Foundation	R3 Consortium	Ripple Labs	Ethereum developers & JP Morgan Chase
Ledger type	Permissionless	Permissioned	Permissioned	Permissioned	Permissioned
Cryptocurrency	Ether (ETH)	None	None	Ripple (XRP)	None
% providers with experience ¹	93%	93%	60%	33%	27%
% share of engagements ²	52%	12%	13%	4%	10%
Coin Market Cap ³	\$91.5 B (18%)	Not applicable	Not Applicable	\$43.9 B (9%)	Not Applicable
Consensus algorithm	Proof of Work (PoW)	Pluggable framework	Pluggable framework	Probabilistic voting	Majority voting
Smart contract functionality	Yes	Yes	Yes	No	Yes

1. Based on responses from 15 leading blockchain service providers

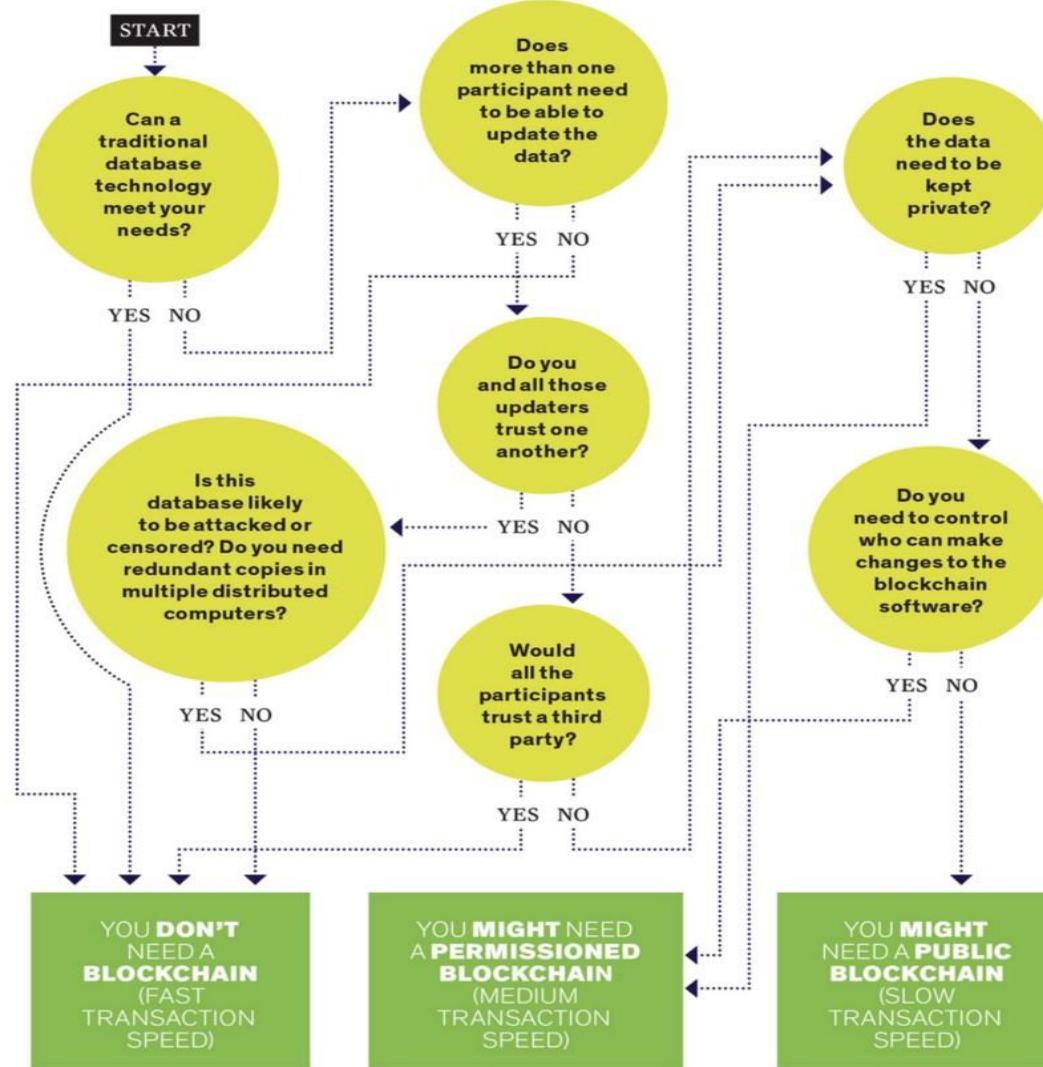
2. Based on a random sample of set of 50 enterprise blockchain engagements across multiple industries

3. Coinmarketcap.com as of Feb 20, 2018, 6:20 PM UTC

Source: HfS Research, 2018



When to use Blockchain





When to use Blockchain

- ❖ Database?
 - ❖ Centralized
 - ❖ Decentralized
- ❖ Secure network transfer?
 - ❖ # parties, frequency
 - ❖ Information
 - ❖ Money (value)
- ❖ Business process automation?
 - ❖ QA/Compliance/Audit
 - ❖ Always-available information
 - ❖ Data sovereignty

Use Case Example: Factom: **Health insurance claims billing**

- Automated claims billing, validation, payment, and settlement
- Multi-party value chain: patient, service provider, billing agent, insurance company, payor, government, collections



When to use Blockchain

- Large Business Network
- Need for multiple ledgers
- Public Verification
- Coin / Token
- Audit / Review
- Can't be solved with a database
- Non-technical (human) problems / barriers
- Systems integration



Requirements Definition

- ❖ Where would having a single shared set of trusted information help in value chain ecosystem?
 - ❖ Blockchain is a single shared database of information and transactions between parties in a value chain.
- ❖ What are obvious ways to deliver customer value?
 - ❖ Financial: What is the cost of transactions/information transfer now? What is the business case for moving to a blockchain solution?
 - ❖ QA Regulation/Compliance: audit-log demonstrates compliance; assures chain of custody.



Next Steps

- ❖ Identify 2-3 Blockchain use cases that would address your business requirements
- ❖ Design and Implement Pilot Project
- ❖ Deployment Strategy
- ❖ Competitive Edge: lead blockchain single shared database and processes in your industry ecosystem
- ❖ Resources
 - ❖ Blockchain consultants, system integrators/vendors