Ethereum-I

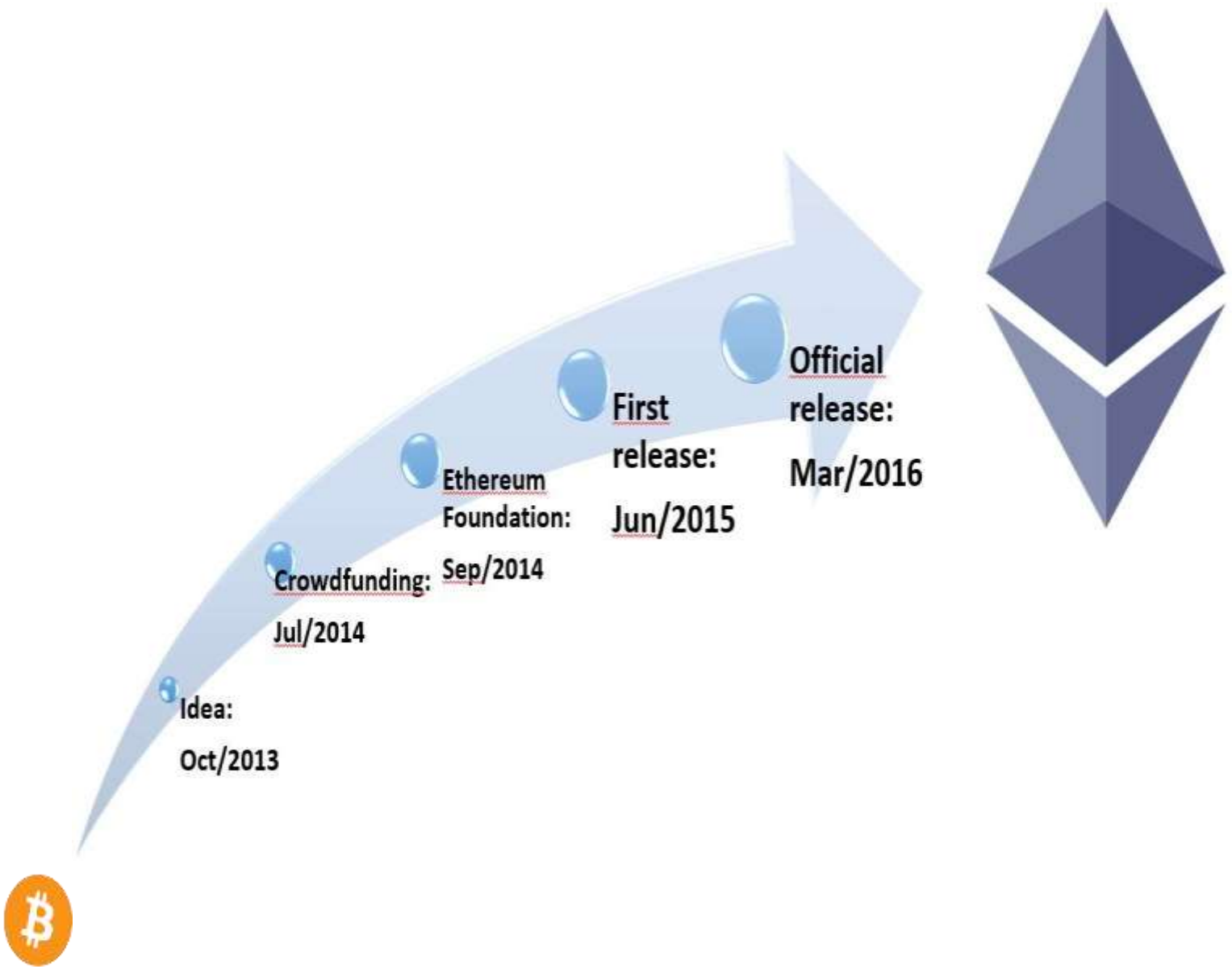
Overview

- Ethereum Basics
- Under the hood
- App deployment and connections

History of Ethereum

- Russian-Canadian programmer
- Co-founded Ethereum when he was 19 years old
- **Vitalik Buterin**
- White paper in Nov 2013

History of Ethereum - Timeline



Important Concepts

- Cryptography (similar to Bitcoin)
- Blockchain
  - Accounts (Two types) and Wallets
    - Externally Owned Accounts (EOA)
    - Contract Accounts
  - Transactions
- Smart Contracts
  - Solidity
    - Language Used for Smart Contract Development

Cryptography

- Hash functions
- Symmetric key Cryptography
- Asymmetric key Cryptography
- Signatures

Hash Functions

- BTC uses SHA-256
- Ethereum uses Keccak-256
  - Similar to SHA-3 (variant)
  - Won contest for security in 2007
  - Used for all hashing in Ethereum
  - Derived differently than standard block-cipher based hashes or previous SHA functions

Digital Signatures (Digital Proof)

- Same use-case/cryptographic method (ECDSA) as BTC:
  - Elliptic Curve Digital Signature Algorithm

- Signer uses private key to generate a signed message
- Signed message can be verified using the signer's public key
- Hashes are signed in Ethereum, not the data itself

Blockchain

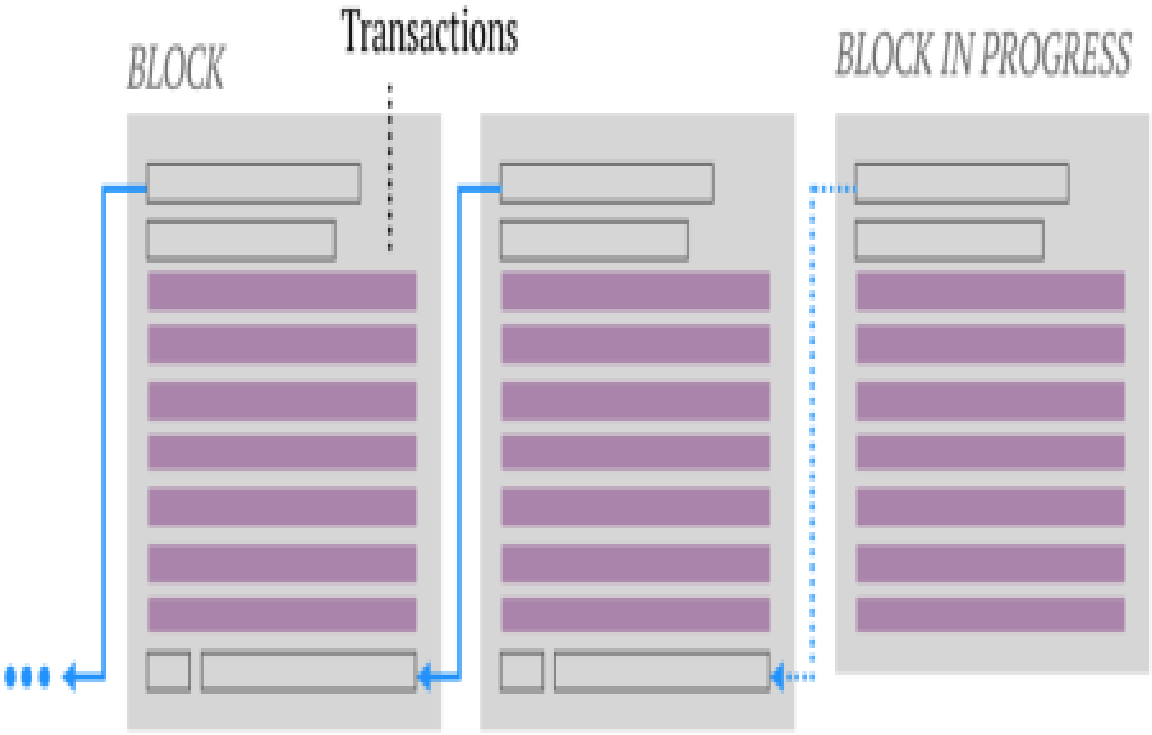
Fully Distributed Database like BTC

Advantages:

- Highly Secure
- Transparent
- Immutable

Disadvantages:

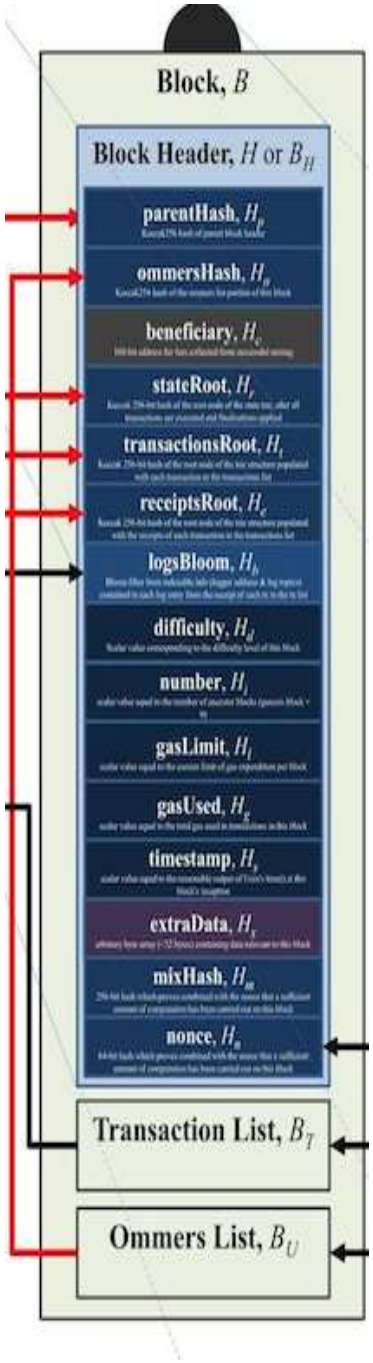
- Scaling
- Performance



Ethereum Blockchain

Blocks consist of 3 elements

- Transaction List
  - List of all transactions included in a block
- Block Header
  - Group of 15 elements
- Ommer List
  - List of all Uncle blocks included (described later)
  - It's possible for two blocks to be created simultaneously by a network. When this happens, one block will be left out. This leftover block is called an ommer block.



Ethereum Blockchain

Uncles/Ommers

- Sometimes valid block solutions don't make main chain
  - Any broadcast block (up to 6 previous blocks back) with valid PoW and difficulty can be included as an uncle
  - Maximum of two can be included per block
- Uncle block transactions are not included – just header
- Ethereum **miners or validators are rewarded** for creating ommer blocks in the Ethereum system through transaction fees to pay for their work.
- No reward for orphan/stale block in **bitcoin**

Ethereum Blockchain

Uncles/Ommers Rewards:

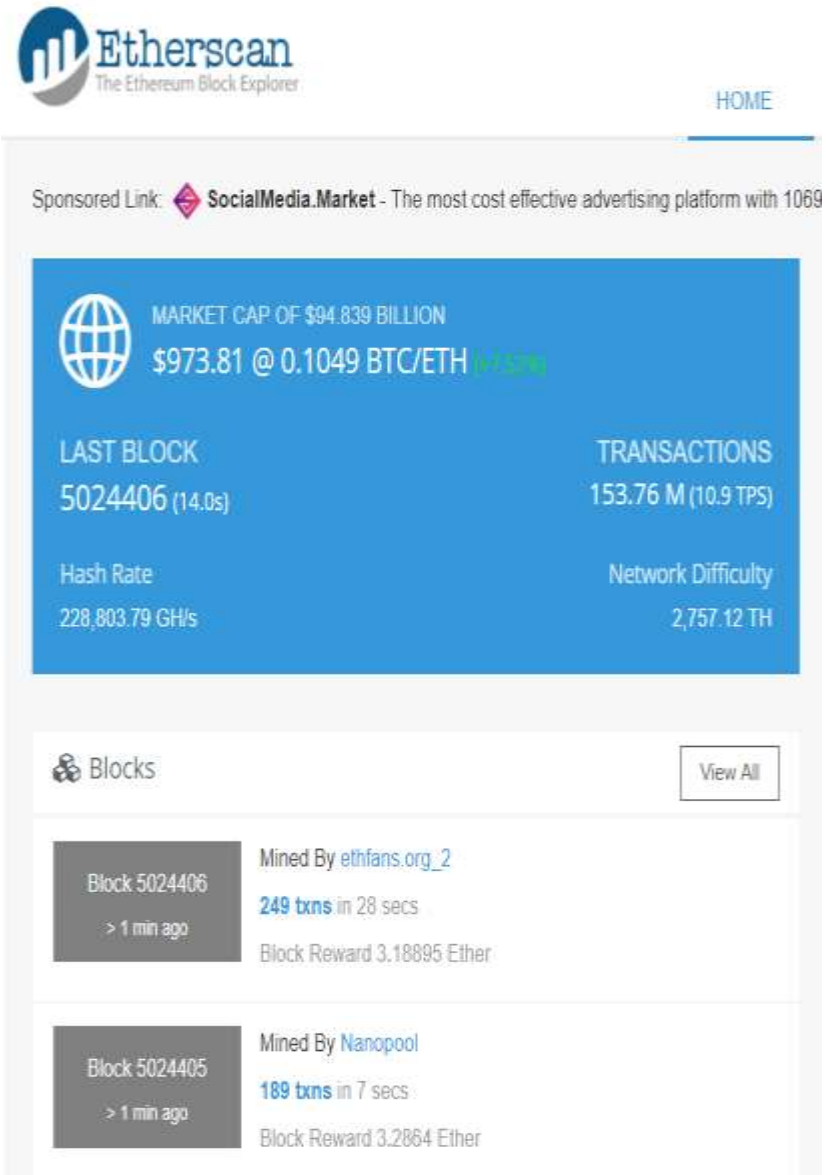
- Uncle headers can be included in main block for 1/32 of the main block miner's reward given to said miner
- <Current_award> * 7/8 ETH
- Miners of uncle blocks receive percent of main reward according to:
  - (Un + (8 - Bn)) * 5<current_reward> / 8,

  where Un and Bn are uncle and block numbers respectively.
  - Example (1333 + 8 - 1335) * ⅝ = 3.75 ETH

https://bitsonblocks.net/2016/10/02/gentle-introduction-ethereum/

Ethereum Blockchain

- All blocks visible like BTC
- However, blocks have a different structure than BTC



https://etherscan.io/

Ethereum Blockchain

Blocks faster than BTC and reward is different

- Every 12 seconds
- 5 ETH main reward
- Miners can make a bit more by including uncle blocks (1/32 of an ETH each) up to maximum of two

Ethereum Blockchain

Blocks faster than BTC and reward is different

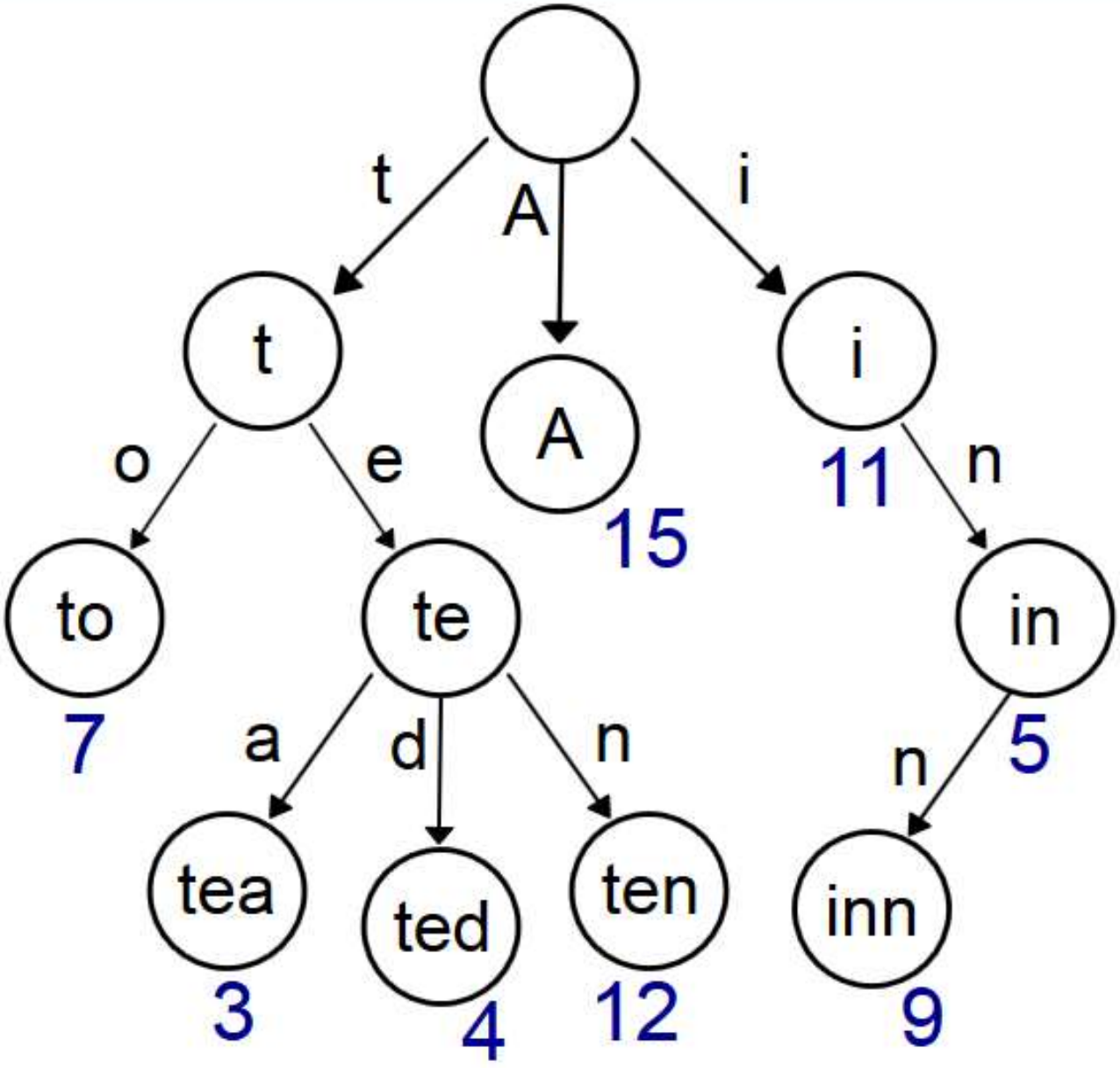
- Uses EthHash mining algorithm (different than Bitcoin)
  - Helps mitigate ASIC (Application-Specific Integrated Circuit) and GPU (graphics processing units) advantages
  - Involves smart contract execution
- Difficulty is adjusted every block
  – this is an important identifier for the Uncle blocks

Ethereum Blockchain

Key differences

- Blocks keep track of balances – not “unspent transaction outputs” like BTC
- Merkle-Patricia tries used (they have three branches compared to the Merkle tree’s two)
- Will transition from Proof of Work to Proof of Stake with Casper protocol

- Patricia: *Practical Algorithm*



Ethereum Modified Merkle-Paricia-Trie System

An interpretation of the Ethereum Project Yellow Paper

G. Wood, "Ethereum: A secure decentralised generalised transaction ledger", 2014.

Lee Thomas

Block Header, H or B_H

stateRoot, H_r

Keccak 256-bit hash of the root node of the state trie, after all transactions are executed and finalisations applied

Hash function:

KECCAK256()

Simplified World State, σ

Keys							Values
a	7	1	1	3	5	5	45.0 ETH
a	7	7	d	3	3	7	1.00 WEI
a	7	f	9	3	6	5	1.1 ETH
a	7	7	d	3	9	7	0.12 ETH

World State Trie

ROOT: Extension Node

prefix	*shared nibble(s)*	*next node*
0	a7	

Branch Node

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*a*	*b*	*c*	*d*	*e*	*f*	*value*

Leaf Node

prefix	*key-end*	*value*
2	1355	45.0ETH

Extension Node

prefix	*shared nibble(s)*	*next node*
0	d3	

Leaf Node

prefix	*key-end*	*value*
2	9365	1.1ETH

Prefixes

0 - Extension Node, even number of nibbles
1□ - Extension Node, odd number of nibbles,
2 - Leaf Node, even number of nibbles
3□ - Leaf Node, odd number of nibbles
□ = 1st nibble
1 nibble = 4 bits

Branch Node

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*a*	*b*	*c*	*d*	*e*	*f*	*value*

Leaf Node

prefix	*key-end*	*value*
3□	7	1.00WEI

Leaf Node

prefix	*key-end*	*value*
3□	7	0.12ETH

Ethereum Nodes

- Validate all transactions and new blocks
- Operate in a P2P fashion
- Each contains a copy of the entire Blockchain
- **Light clients - store only block headers**
  - Provide easy verification through tree data structure
  - Don’t execute transactions, used primarily for balance validation
- Implemented in a variety of languages (Go, Rust, etc.)

Accounts and Wallets

Accounts:

- Two Kinds:
  - External Owned Accounts - (EOA, most common account)
  - Contract Accounts
- Consist of a public/private keypair
- Allow for interaction with the blockchain

Wallets:

- A set of one or more external accounts
- Used to store/transfer ether

Accounts and Wallets

External Account (EOA, Valid Ethereum Address)

- Has an associated nonce (amount of transactions sent from the account) and a balance

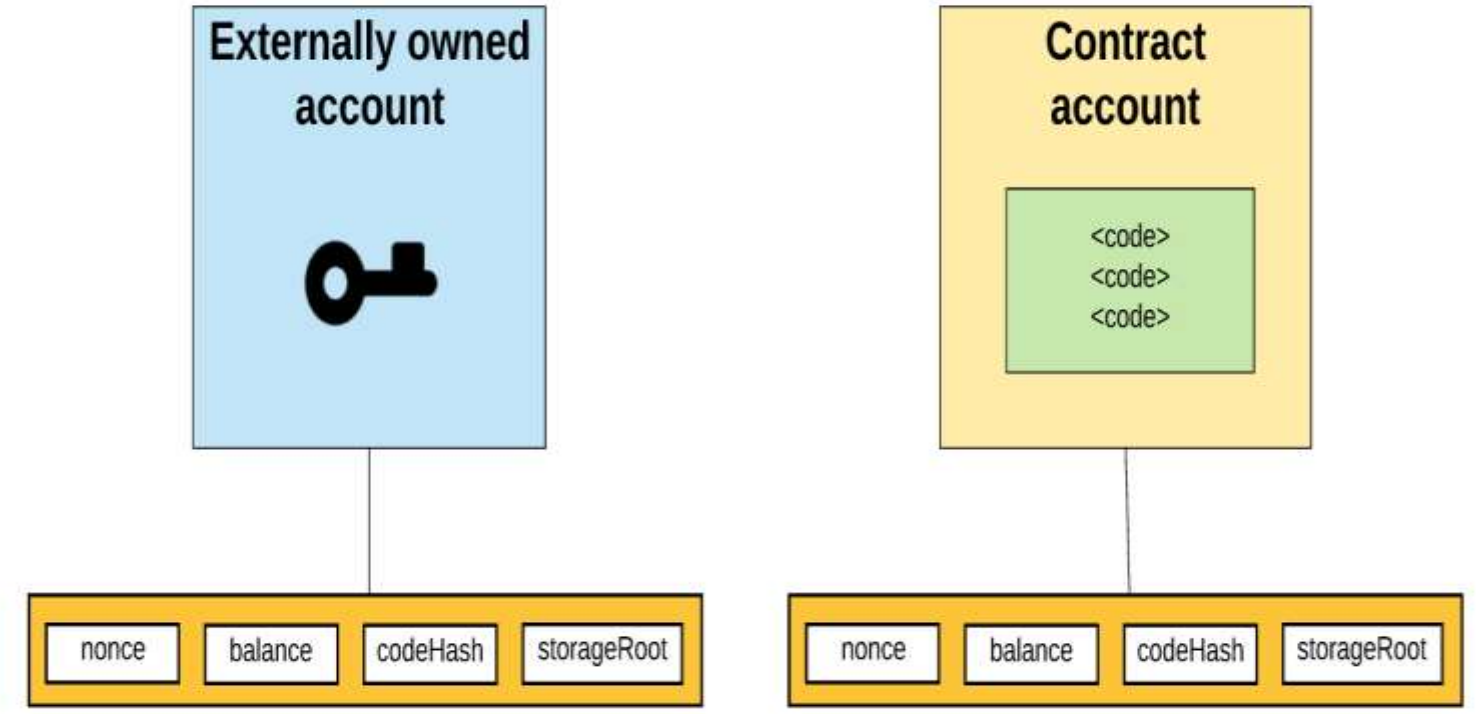
- codeHash - Hash of associated account code, i.e. a computer program for a smart contract (hash of an empty string for external accounts, EOAs)

- Storage Root is root hash of Merkle-Patricia trie of associated account data

Accounts and Wallets

Contract Account

- Ethereum accounts can store and execute code
  - Has an associated nonce and balance
  - codeHash - hash of associated account code storageRoot contains Merkle tree of associated storage data



Example Account

Private Key:

0x2dcef1bfb03d6a950f91c573616cdd778d9581690db1cc43141f7cca06fd08ee

- Ethereum Private keys are **66 character** strings (with 0x appended). Case is irrelevant. Same derivation through ECDSA as BTC.

Address:

0xA6fA5e50da698F6E4128994a4c1ED345E98Df50

- Ethereum Private keys map to addresses directly. Simply the **last 40 characters** of the **Keccak-256 hash** of the public key. Address is **42 characters total** (append 0x to front).

Transactions

- A request to modify the state of the blockchain
  - Can run code (contracts) which change global state
    - Contrasts only balance updates in BTC
- Signed by originating account
- Types:
  - Send value from one account to another account
  - Create smart contract
  - Execute smart contract code

Ether Denominations

- Wei - lowest denomination
  - Named after Wei Dai - author of b-money paper (1998), many core concepts used in BTC implementation
  - 1/1,000,000,000,000,000,000 (quintillion)

- Szabo - next denomination
  - Named after Nick Szabo - author of Bit-Gold

- Finney – 2nd highest denomination
  - Named after Hal Finney - received first Tx from Nakamoto

Multiplier	Name
10^0	Wei
10^{12}	Szabo
10^{15}	Finney
10^{18}	Ether

http://www.weidai.com/bmoney.txt

Smart Contracts

- Executable code
- Turing Complete
- Function like an external account
  - Hold funds
  - Can interact with other accounts and smart contracts
  - Contain code
- Can be called through transactions

Code Execution

- Every node contains a virtual machine (similar to Java)
  - Called the Ethereum Virtual Machine (EVM)
  - **Compiles** code from high-level language to bytecode
  - Executes smart contract code and broadcasts state
- ***Every full-node on the blockchain processes every transaction and stores the entire state***

Gas

- Halting problem (infinite loop) – reason for Gas
  - Problem: Cannot tell whether or not a program will run infinitely from compiled code
  - Solution: charge fee per computational step to limit infinite loops and stop flawed code from executing
- Every transaction needs to specify an estimate of the amount of gas it will spend
- Essentially a measure of how much one is willing to spend on a transaction, even if buggy

Gas Cost

- Gas Price: current market price of a unit of Gas (in Wei)
  - Check gas price here: https://ethgasstation.info/
  - Is always set before a transaction by user
- Gas Limit: maximum amount of Gas user is willing to spend
- Helps to regulate load on network
- Gas Cost (used when sending transactions) is calculated by gasLimit*gasPrice.
  - All blocks have a Gas Limit (maximum Gas each block can use)

PoW vs. PoS

Ethereum in the process of moving to Proof of Stake

- This approach does not require large expenditures on computing and energy
- Miners are now “validators” and post a deposit in an escrow account
- The more escrow you post, the higher the probability you will be chosen to nominate the next block
- If you nominate a block with invalid transactions, you lose your escrow

PoW vs. PoS

Ethereum in the process of moving to Proof of Stake

- One issue with this approach is that those that have the most ethereum will be able to get even more
- This leads to centralization eventually
- On the other hand, it reduces the chance of a 51% attack and allows for near instant transaction approvals
- The protocol is called Casper and this will be a hard fork

https://blockonomi.com/ethereum-casper/

Other approaches to conensus

There are many other types of consensus

- (PoW) Proof of Work (Bitcoin, Ethereum, …)
- (PoS) Proof of Stake (Ethereum in future)
- (PoI) Proof of Importance (used in NEM)
- (PBFT) Practical Byzantine Fault Tolerance (Hyperledger Fabric)
- (FBFT) Federated Byzantine Fault Tolerance (Ripple, Stellar)
- (DPoS) Delegated Proof of Stake
- (PoET) Proof of Elapsed Time (Hyperledger Sawtooth)

https://medium.com/@chrshmmmr/consensus-in-blockchain-systems-in-short-691fc7d1fefe

Reading

- Highly recommended intro
- https://medium.com/@preethikasireddy/how-does-ethereum-work-anyway-22d1df506369