Ethereum

Campbell R. Harvey*
Duke University and NBER

Ashwin Ramachandran
Duke University

Brent Xu
ConsenSys

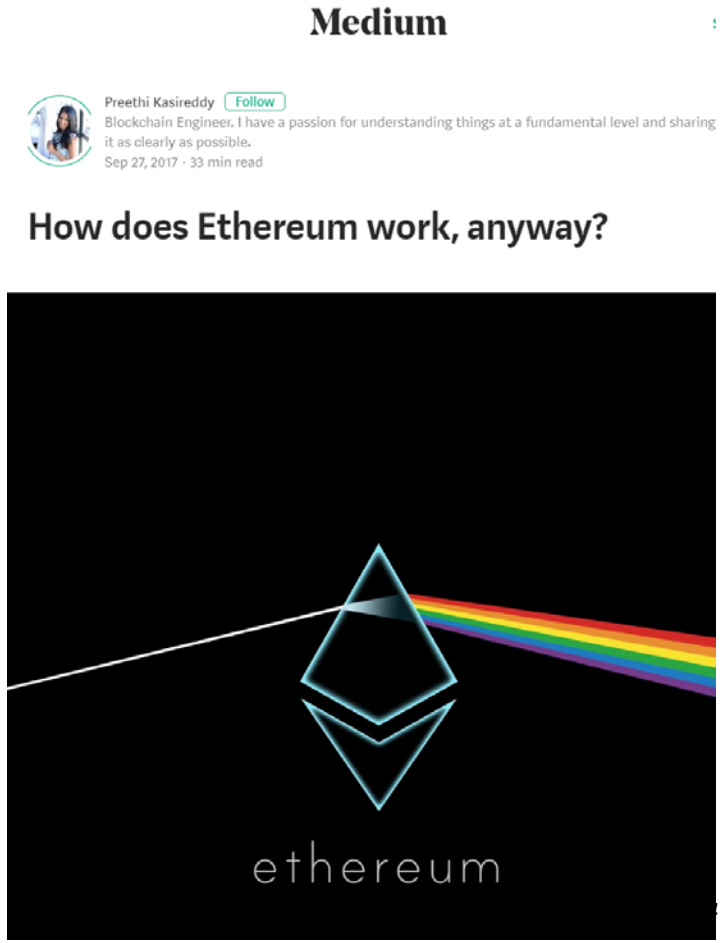
February 12, 2018

Overview

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

Overview

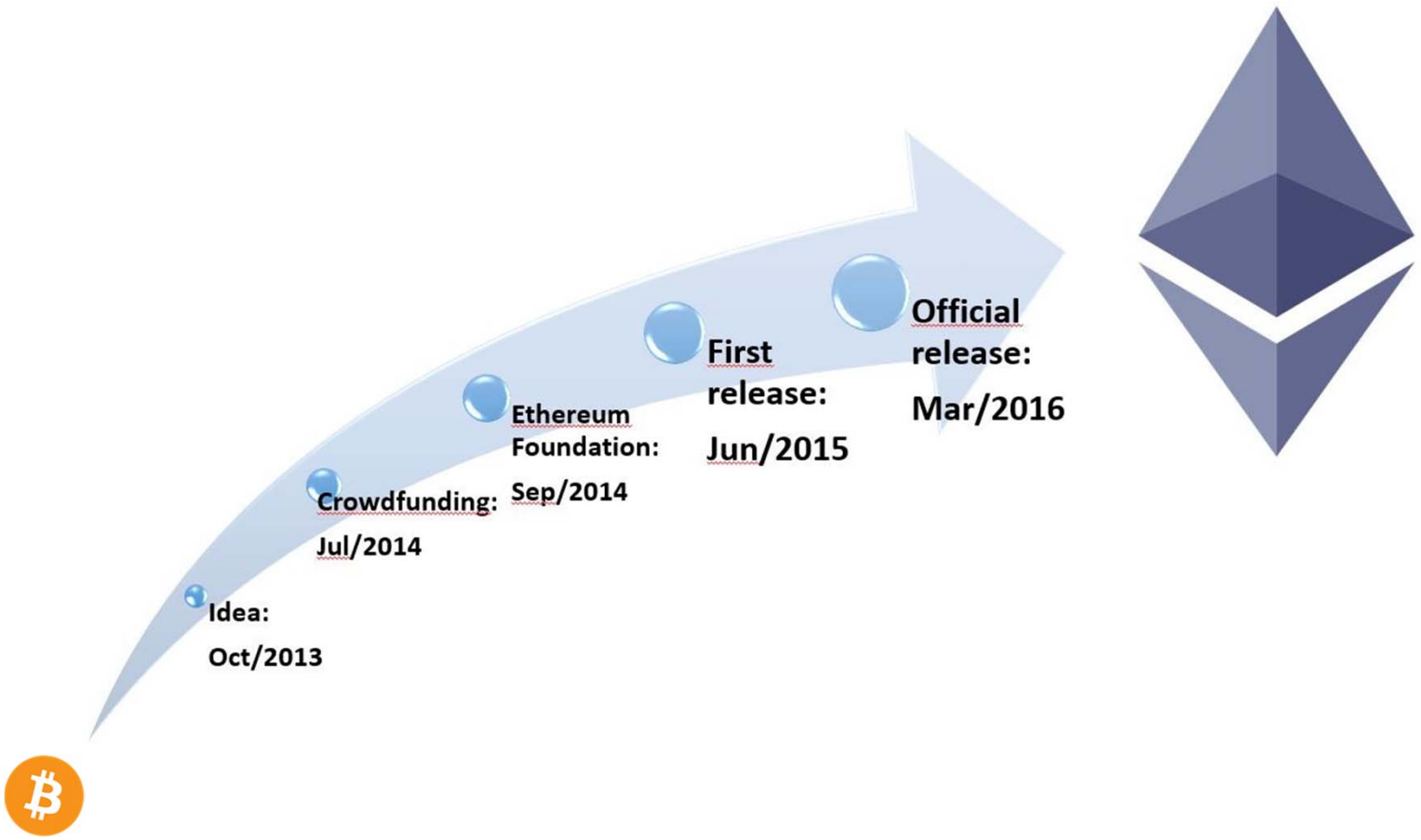
- Highly recommended intro
- https://medium.com/@preethikasireddy/how-does-ethereum-work-anyway-22d1df506369
- I draw some graphics from the above article in the presentation



History of Ethereum

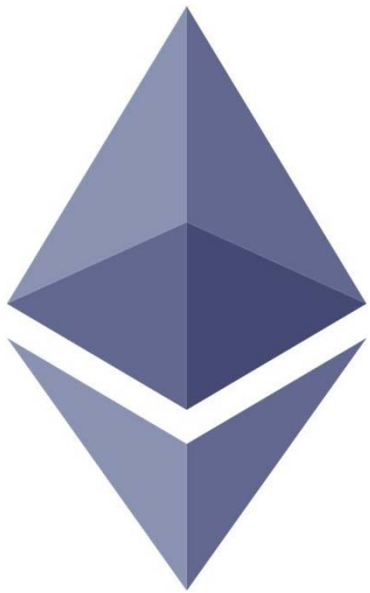
- Russian-Canadian programmer
- Co-founded Ethereum when he was 19 years old

History of Ethereum - Timeline



Important Concepts

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



Cryptography

- Hash functions
- Symmetric Cryptography
- Asymmetric 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
- 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

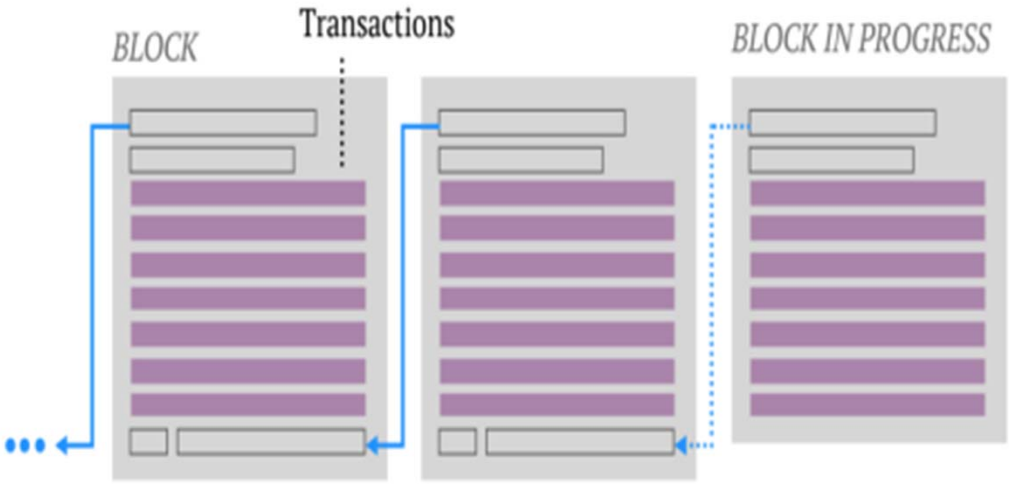
Fully Distributed Database like BTC

<u>Advantages</u>:

- Highly Secure
- Transparent
- Immutable

<u>Disadvantages</u>:

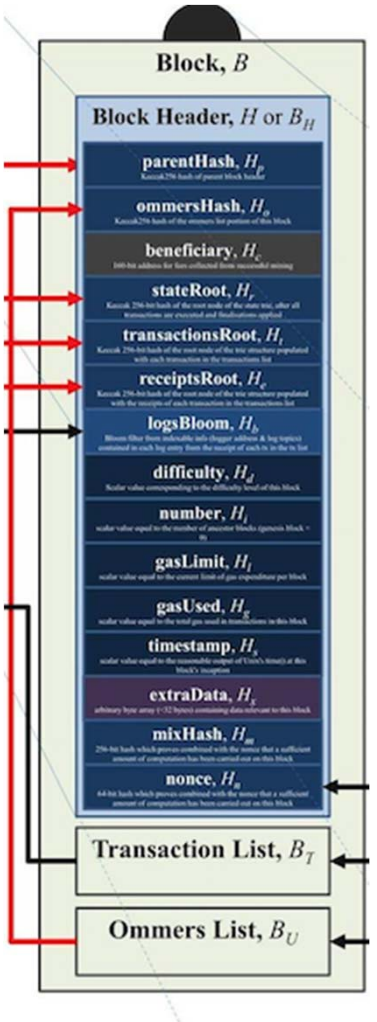
- Scaling
- Performance



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)



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
- Aimed to decrease centralization and reward work

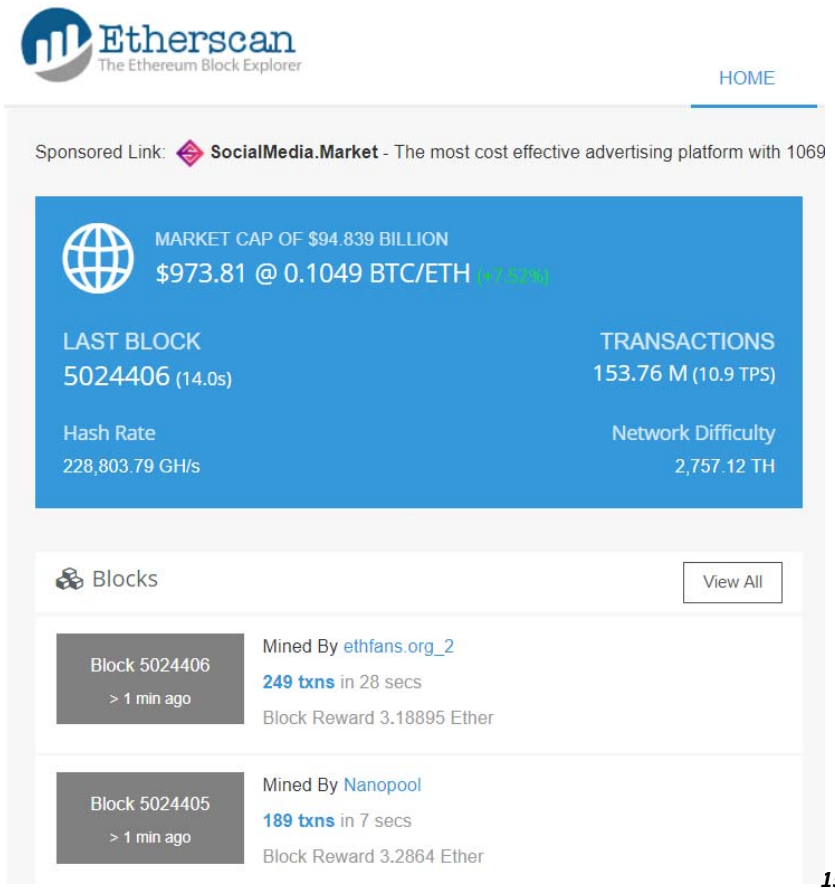
Ethereum Blockchain

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
- Miners of uncle blocks receive percent of main reward according to:
  - $(U_n + (8 - B_n)) * 5 / 8$, where U_n and B_n are uncle and block numbers respectively.
  - Example (1333 + 8 - 1335) * ⅝ = 3.75 ETH

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 and GPU advantages
  - Involves smart contract execution
- Difficulty is adjusted every block (not every two weeks) – 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
- See appendix for more details

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

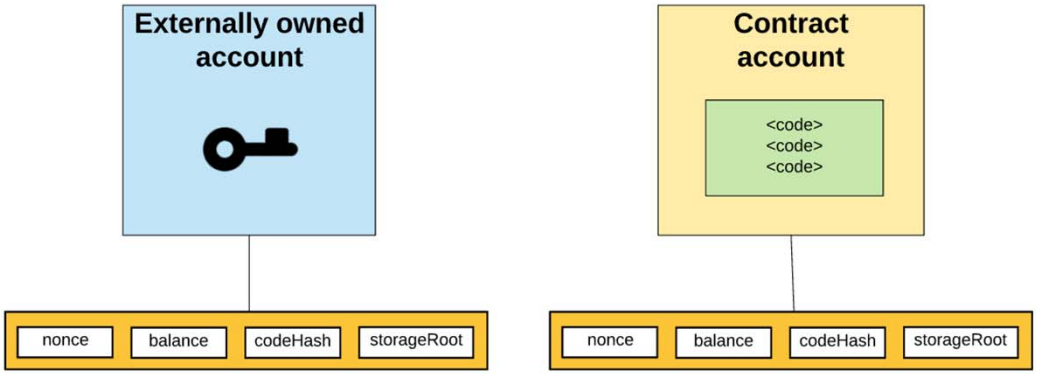
External Account (EOA, Valid Ethereum Address)

- 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

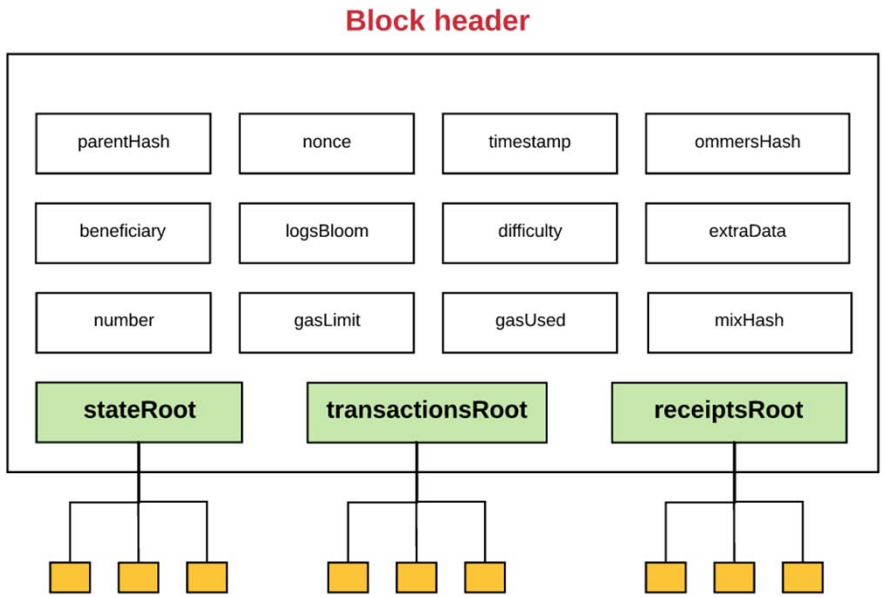
- (PoW) Proof of Work (Bitcoin, Ethereum, …)
- (PoS) Proof of Stake (Ethereum in future)
- (PoI) Proof of Importance (used in NEM)
- (PBFT) Practical Byzantine Fault Tolerance (Hyperledger Fabric)
- (FBFT) Federated Byzantine Fault Tolerance (Ripple, Stellar)
- (DPoS) Delegated Proof of Stake
- (PoET)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

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

Appendix materials

A. Ethereum Blockchain Header

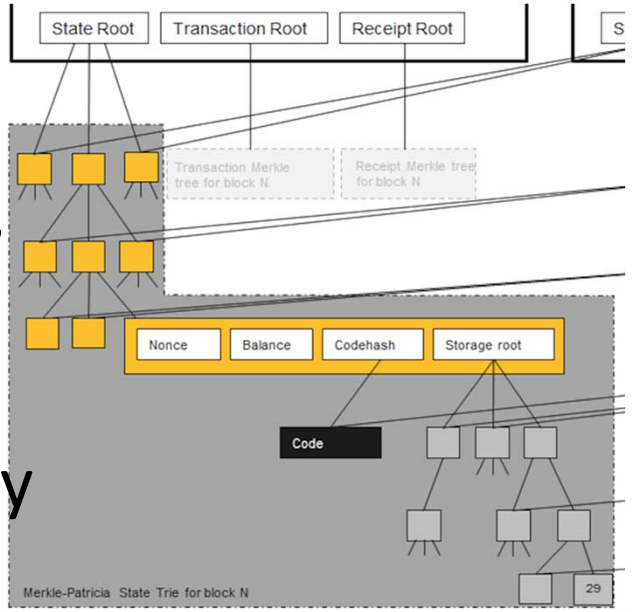
- Hash of included ommer's stored in block header
- State root is the hash of a merkle trie that holds all account information
- Similar storage structure for transactions and receipts



A. Ethereum Blockchain State

StateRoot, TransactionRoot, and ReciptsRoot

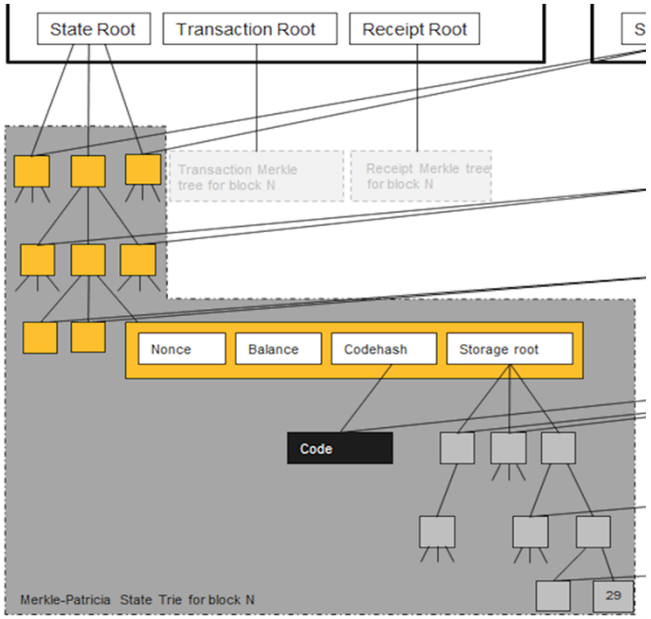
- Stored in data structure known as a Merkle-Patricia trie
- Similar to the Merkle trie used in BTC, but with three leaves per node
- Trie is cryptographically secure as any alteration of a leaf or intermediary node results in a different root hash



A. Ethereum Blockchain State

StateRoot

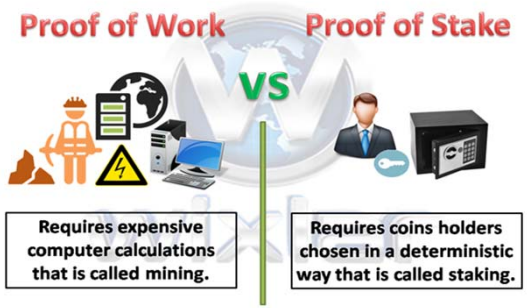
- Each node in the stateRoot trie represents an Ethereum address
- Each address has 4 components
  - Nonce - list of number of Tx's from address
  - CodeHash - hash of associated code
  - StorageRoot - Merkle-Patricia tree root of account storage contents
  - Balance - balance of account

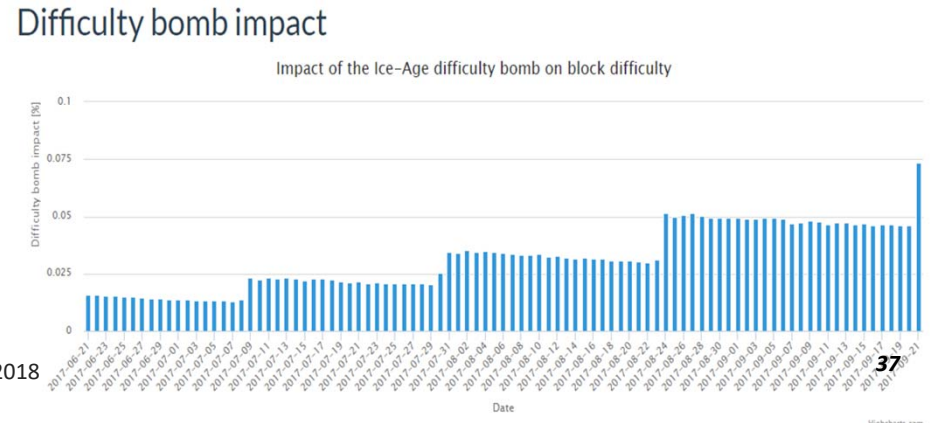


A. Ethereum Blockchain

Ethereum “difficulty bomb”

- Spike (increase) in mining difficulty
- Introduced to attempt to reduce number of miners
  - Aimed to pre-date shift of algorithm from PoW to Proof-of-Stake (PoS)





Campbell R. Harvey 2018

B. Smart Contract Programming

- Solidity (javascript based), most popular
  - Not yet as functional as other, more mature, programming languages
- Serpent (python based)
- LLL (lisp based)

B. Smart Contract Programming

Solidity

Solidity is a language similar to JavaScript which allows you to develop contracts and compile to EVM bytecode. It is currently the flagship language of Ethereum and the most popular.

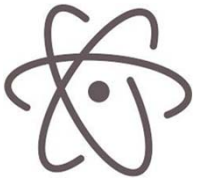
- Solidity Documentation - Solidity is the flagship Ethereum high level language that is used to write contracts.
- Solidity online realtime compiler

Serpent

Serpent is a language similar to Python which can be used to develop contracts and compile to EVM bytecode. It is intended to be maximally clean and simple, combining many of the efficiency benefits of a low-level language with ease-of-use in programming style, and at the same time adding special domain-specific features for contract programming. Serpent is compiled using LLL.

- Serpent on the ethereum wiki
- Serpent EVM compiler

B. Smart Contract Programming



Atom Ethereum interface - Plugin for the Atom editor that features syntax highlighting, compilation and a runtime environment (requires backend node).

Atom Solidity Linter - Plugin for the Atom editor that provides Solidity linting.

Vim Solidity - Plugin for the Vim editor providing syntax highlighting.

Vim Syntastic - Plugin for the Vim editor providing compile checking.

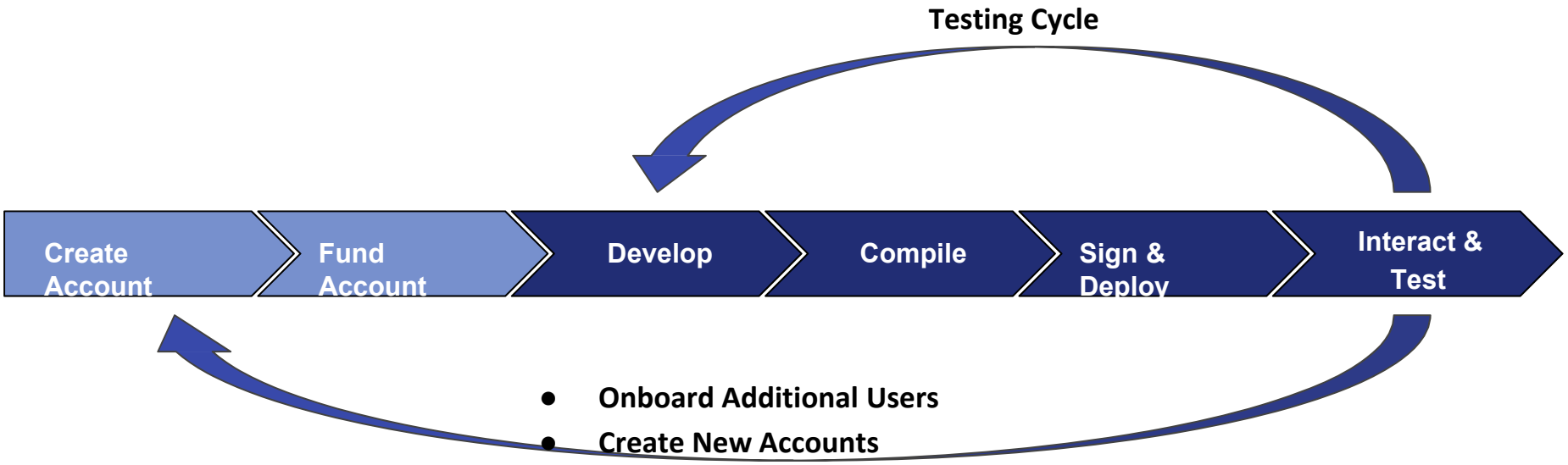
B. Smart Contract Programming: Solidity

```
contract Example {

   uint value;

   function setValue(uint pValue) {
      value = pValue;
   }

   function getValue() returns (uint) {
      return value;
   }

}
```

B. Smart Contract Programming: Solidity

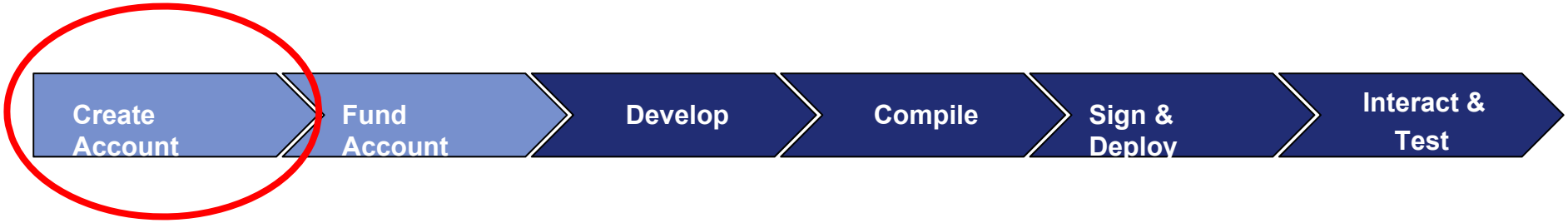
```
var logIncrement =
   OtherExample.LogIncrement({sender: userAddress,
uint value});

logIncrement.watch(function(err, result) {
  // do something with result
})
```

C. Development Workflow



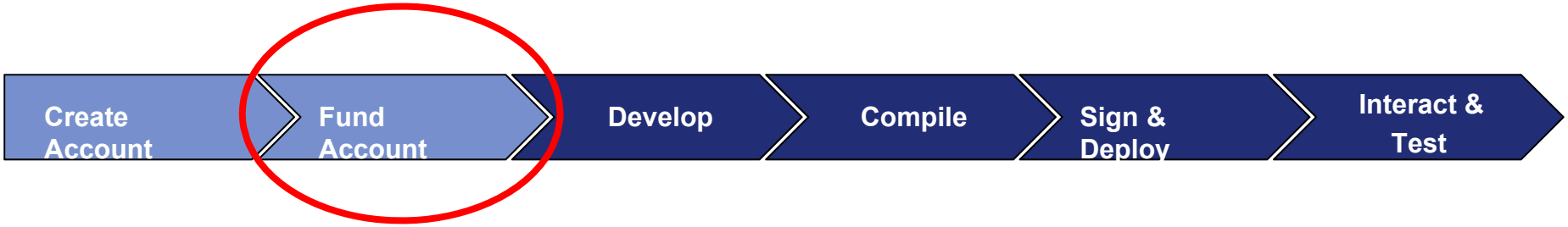
- **Onboard Additional Users**
- **Create New Accounts**
- **Develop New Applications**

C. Development Workflow: Create Account



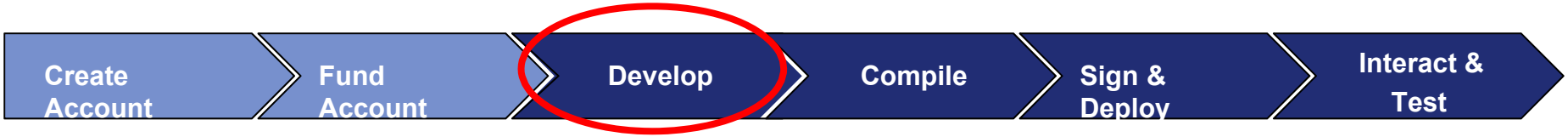
- Programmatically: Go, Python, C++, JavaScript, Haskell
- Tools
  - MyEtherWallet.com
  - MetaMask
  - TestRPC
  - Many other websites

C. Development Workflow: Fund Account



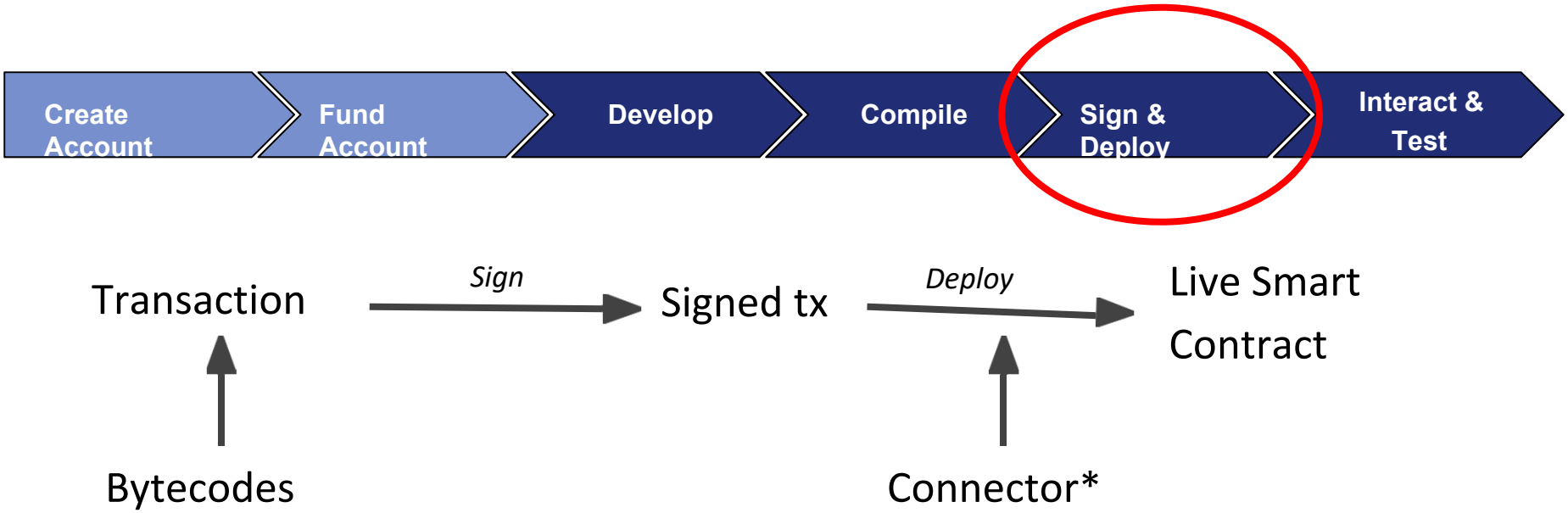
- From friends
- Faucet
- Exchanges (for public blockchain)

C. Development Workflow: Develop



- **Ethereum Application Components:**
  - **Base application**: can be developed in **any** language
  - **Smart contract**: developed in Solidity or one of the other contract compatible languages
  - **Connector library**: facilitates communication between base application and smart contracts (Metamask)

C. Development Workflow: Sign and Deploy



*Library that facilitates communication and connection with Blockchain; Connects your code to a running node.

C. Development Workflow: TestRPC

TestRPC/TestChain

- Local development or Test Blockchain
- https://github.com/ethereumjs/testrpc

C. Development Workflow: TestRPC

- EthereumJS TestRPC: https://github.com/ethereumjs/testrpc is suited for development and testing
- It's a complete blockchain-in-memory that runs only on your development machine
- It processes transactions instantly instead of waiting for the default block time – so you can test that your code works quickly – and it tells you immediately when your smart contracts run into errors
- It also makes a great client for automated testing
- Truffle knows how to use its special features to speed up test runtime by almost 90%.