Blockchain – Smart Contracts – 3

(Aug 26, 2019)

Dr. Deven Shah

TCET Mumbai

COMPONENTS OF ETHEREUM

- Accounts
- Keyfiles
- Validators or Miners
- Ethereum Virtual Machine
- Ethereum Software Releases
- Ethereum Clients
- Wallets and Full Nodes
- Mist Browser
- Geth
- Web3.js
- Ganache
- Smart Contracts

ACCOUNTS

- Every account is defined by a pair of keys, a private key and public key.
- Accounts are indexed by their address which is derived from the public key.
- Currently, there are two types of accounts on the Ethereum blockchain:
  - Externally owned accounts (EOAs)
  - Contract accounts (CA)
- All action on the Ethereum block chain is set in motion by transactions fired from externally owned accounts.
- Every time a contract account receives a transaction, its code is executed as instructed by the input parameters sent as part of the transaction.
- The contract code is executed by the Ethereum Virtual Machine on each node participating in the network as part of their verification of new blocks.

ACCOUNTS

- **Externally Owned Accounts (EOAs)**
  - has an ether balance,
  - can send transactions (to transfer ether or to trigger contract code),
  - is controlled by private keys,
  - has no associated code.
  - Example:https://etherscan.io/address/0x2d7c76202834a11a99576acf2ca95a7e66928ba0

ACCOUNTS

- **Contract accounts**
  - has an ether balance,
  - has associated code,
  - code execution is triggered by transactions or messages (calls) received from other contracts.
  - when executed
    - perform operations of arbitrary complexity (Turing completeness)
    - manipulate its own persistent storage, i.e., can have its own permanent state
    - can call other contracts
  - Example: https://etherscan.io/address/0xcbe1060ee68bc0fed3c00f13d6f110b7eb6434f6#code

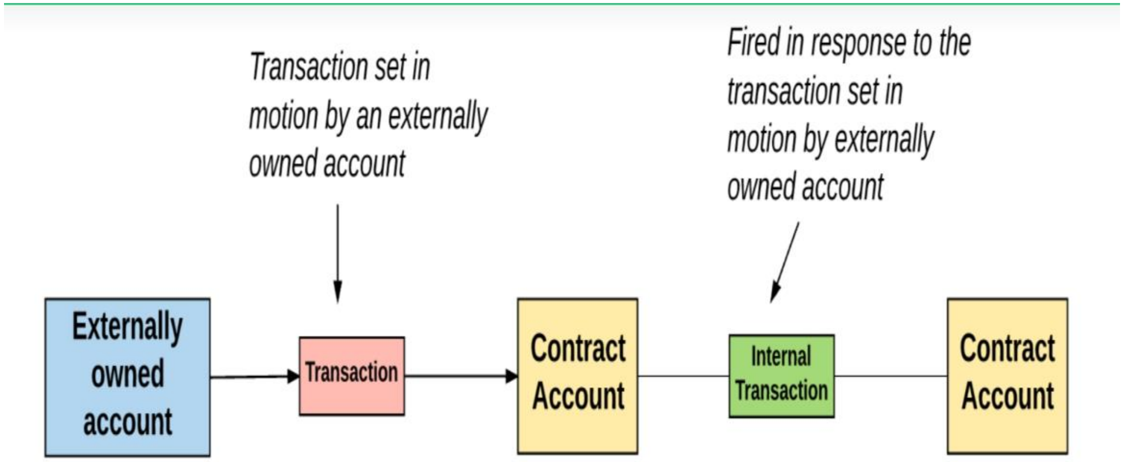
- Each account has state associated with it and 20 byte address

- EOA use private key to sign transaction for another EOA or CA

- EOA to CA – activate contract account’s code

- CA can’t initiate any transactions on its own.

- CA only fire transaction



Transaction set in motion by an externally owned account

Fired in response to the transaction set in motion by externally owned account

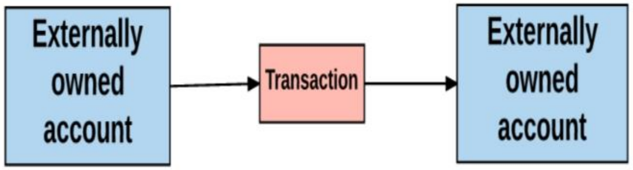
Externally owned account

Transaction

Contract Account

Internal Transaction

Contract Account



Externally owned account

Transaction

Externally owned account

Account state

- Regardless of type of account, Account state consists of four components
- **Nonce:** for EOA this no. represent the number of transactions sent from the account’s address. For CA, this no. is the number of contract created by the account
- **Balance:** The number of Wei owned by this address
- **Storage Root:** A hash of root node of Merkle Patricia tree i.e this tree encodes the hash of the storage contents of this account and it is empty by default
- **Code Hash:** The hash of EVM code of this account . For CA code got hash and for EOA hash of empty string

Global stage of Ethereum is mapping between account address and account state

KEYFILES

- Every private key/address pair is encoded in a keyfile.
- Keyfiles are JSON text files which you can open and view in any text editor.
- The critical component of the keyfile, your account’s private key, is always encrypted, and it is encrypted with the password you enter when you create the account.

VALIDATORS OR MINERS

- Anyone (with the appropriate computing hardware) can take on the role of a miner and validate and generate blocks.
- **Block Reward**
  - In order to incentivize miners for supporting the Ethereum network, a block reward is granted to the lucky miner who generates the “correct” block.
  - Currently, the block reward is set at 2 Ether.
- **Mining Pools**
  - Small-scale miners may join a mining pool for a consistent payout, instead of waiting for their mining rigs to successfully mine a “correct” block.
  - The mining pool bands together a large number of miners and distributes each block reward (minus a small “pool fee”) that they obtain according to the contribution from each miners.

ETHEREUM VIRTUAL MACHINE

- The Ethereum network functions as one large computer which executes programs in lockstep; it is a **machine** which is **“virtualized” by a network of other machines.**
- Being composed of many private computers, the **Ethereum Virtual Machine (EVM)** itself can be said to be a **shared computer which is ownerless.**
- **Changes to the EVM** are achieved through ***hard forking***: persuading the entire community of node operators to upgrade to a new version of the Ethereum software.
- Changes to the network can’t simply be pushed by the core development team.
- This **ownerless configuration** is meant to maximize uptime and security, while minimizing the incentive for malicious activities.
- EVM, thus, performs **computation without relying on the central server.**

ETHEREUM SOFTWARE RELEASES

- **Olympic** (testnet): Launched May 2015 – a testing release where coins are not compatible with ‘real’ ETH. A testnet still runs in parallel to the main live network so that developers can test their code.
- The complete launch process of Ethereum was divided into 4 stages.
- This was done to make sure that various phases got their own developmental time and that every stage was developed as efficiently and optimally as possible.
- The 4 stages are as follows:
- **Frontier**: Launched 30 July 2015 – an initial live release with a way for people to mine ETH and build and run contracts.
- **Homestead**: Launched 14 March 2016 – some protocol changes, more stability.
- **Metropolis**: Future launch – moving from command-line to graphical interfaces.
- **Serenity**: Future launch – moving from Proof of Work to Proof of Stake (Casper).

ETHEREUM CLIENTS

- The official Ethereum clients are all **open source** – that is you can see the code behind them, and tweak them to make your own versions. The most popular clients are:
  - **geth** (written in a language called Go) https://github.com/ethereum/go-ethereum
  - **eth** (written in C++) https://github.com/ethereum/cpp-ethereum
  - **pyethapp** (written in Python) https://github.com/ethereum/pyethapp
- These are all command-line based programs and so additional software can be used for a nicer graphical interface.
- Currently the official and most popular graphical one is Mist (https://github.com/ethereum/mist), which runs on top of geth or eth.

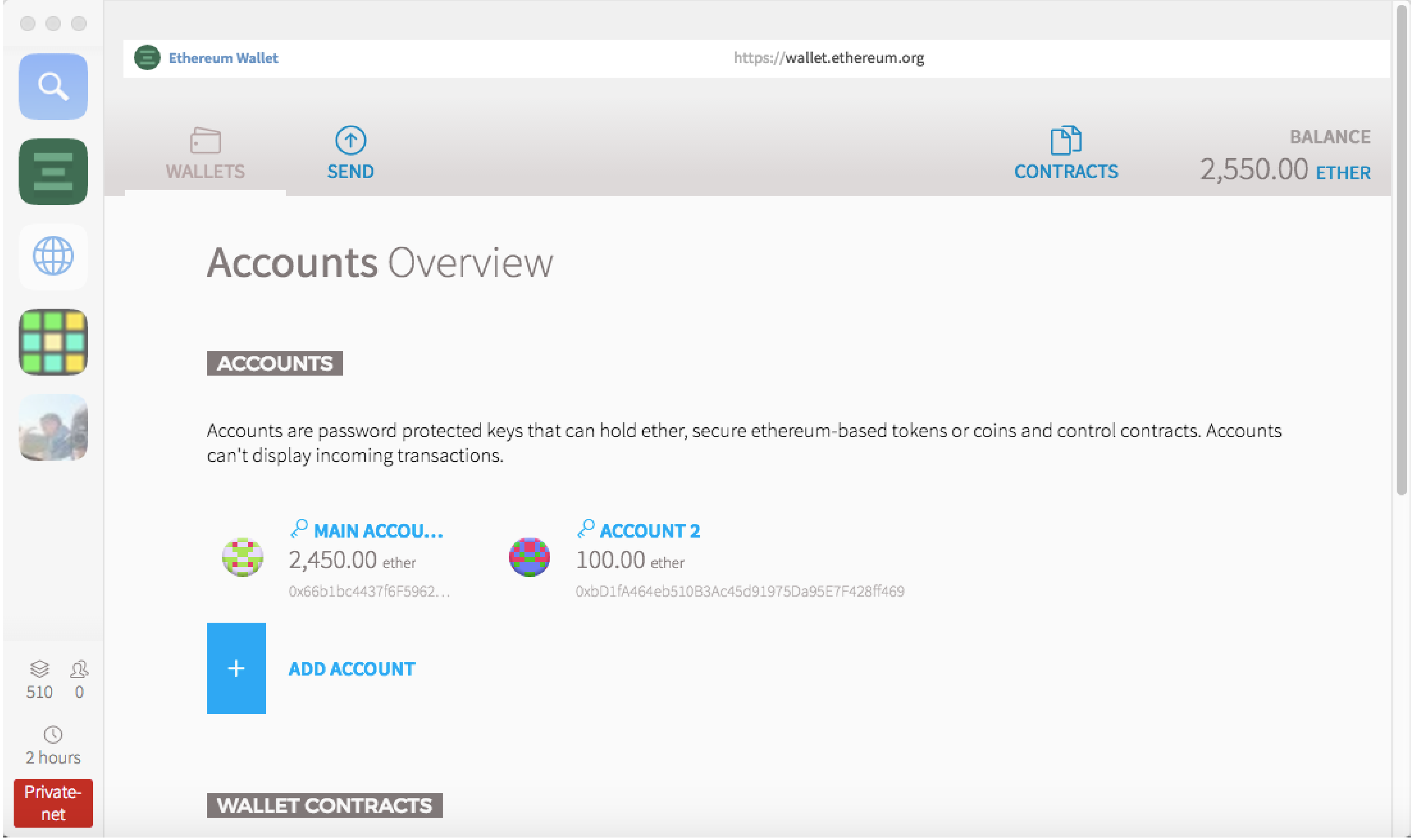
WALLETS AND FULL NODES

- **Wallet** usually denotes a lightweight node that connects to a blockchain to perform basic functions, such as sending and receiving cryptocurrency.
- *Wallets* are software applications for desktop or mobile devices that hold your *keys* to the EVM.
- These keys correspond to an *account*, which is referred to by a long account address.
- In Ethereum, accounts do not store your name or any other personal information. They are pseudonymous.
- **Full nodes** are command-line interfaces that can perform the full gamut of operations allowed by the network.

MIST BROWSER

- **Ethereum** has several **client application**s, the most useful is the **Mist browser**.
- It is a **user-friendly wallet** that can perform some of the duties of a full node—namely, executing smart contracts.
- **Eventually**, **entire web-app-like programs will be accessible through Mist**, with their back ends built on Ethereum; that’s why it’s called a ***browser***.
- **Today**, it’s useful for sending and receiving the ether cryptocurrency.
- But **tomorrow, it may also be a distribution point for consumer and enterprise software applications**, almost like an App Store.
- The Mist browser is **compatible** with Linux, macOS, and Windows computers with both 32- and 64-bit architectures.

MIST BROWSER



Accounts Overview

ACCOUNTS

Accounts are password protected keys that can hold ether, secure ethereum-based tokens or coins and control contracts. Accounts can't display incoming transactions.

MAIN ACCOU...
2,450.00 ether
0x66b1bc4437f6F5962...

ACCOUNT 2
100.00 ether
0xbD1fA464eb510B3Ac45d91975Da95E7F428ff469

WALLET CONTRACTS

GETH

- Geth is a multipurpose **command line tool** that runs a full Ethereum node implemented in Go.
- It is the **main deliverable** of the Frontier Release.
- It offers **three interfaces**:
  - **the command line subcommands and options**
  - **a Json-rpc server**: geth can be launched with a json-rpc server that exposes the JSON-RPC API.
  - **an interactive console**: geth can be launched with an interactive console, that provides a javascript runtime environment exposing a javascript API to interact with your node.
- JavaScript Console API includes the **web3 javascript Dapp API** as well as an additional admin API.
- By **installing and running geth**, you can take part in the ethereum live network and
  - Mine real ether
  - Transfer funds between addresses
  - Create contracts and send transactions
  - Explore block history
  - And much much more.

WEB3.JS

- When you want to make one of the peer-to-peer (P2P) networks accessible through a web browser, you need to use special software libraries such as **Web3.js to connect an application's front end** (the GUI you see in a browser), via JavaScript APIs, to its back end (the blockchain).
- web3.js is a **collection of libraries** which allow you **to interact with a local or remote ethereum node, using a HTTP or IPC connection.**
- **Web3-eth** is for the ethereum blockchain and smart contracts
- **Web3-shh i**s for the whisper protocol to communicate p2p and broadcast
- **Web3-bzz i**s for the swarm protocol, the decentralized file storage
- **Web3-utils** contains useful helper functions for Dapp developers.

GANACHE

- Ganache is a personal blockchain for Ethereum development you can use to deploy contracts, develop your applications, and run tests.
- **I**t is available as both a desktop application as well as a command-line tool (formerly known as the TestRPC).
- Ganache is available for Windows, Mac, and Linux.
- When you launch Ganache, the screen will show
  - some details about the server, and also
  - list out 10 accounts and
  - their private keys.
- **Each account is given 100 ether**.
- Having ether automatically in all accounts allows you to focus on developing your application.

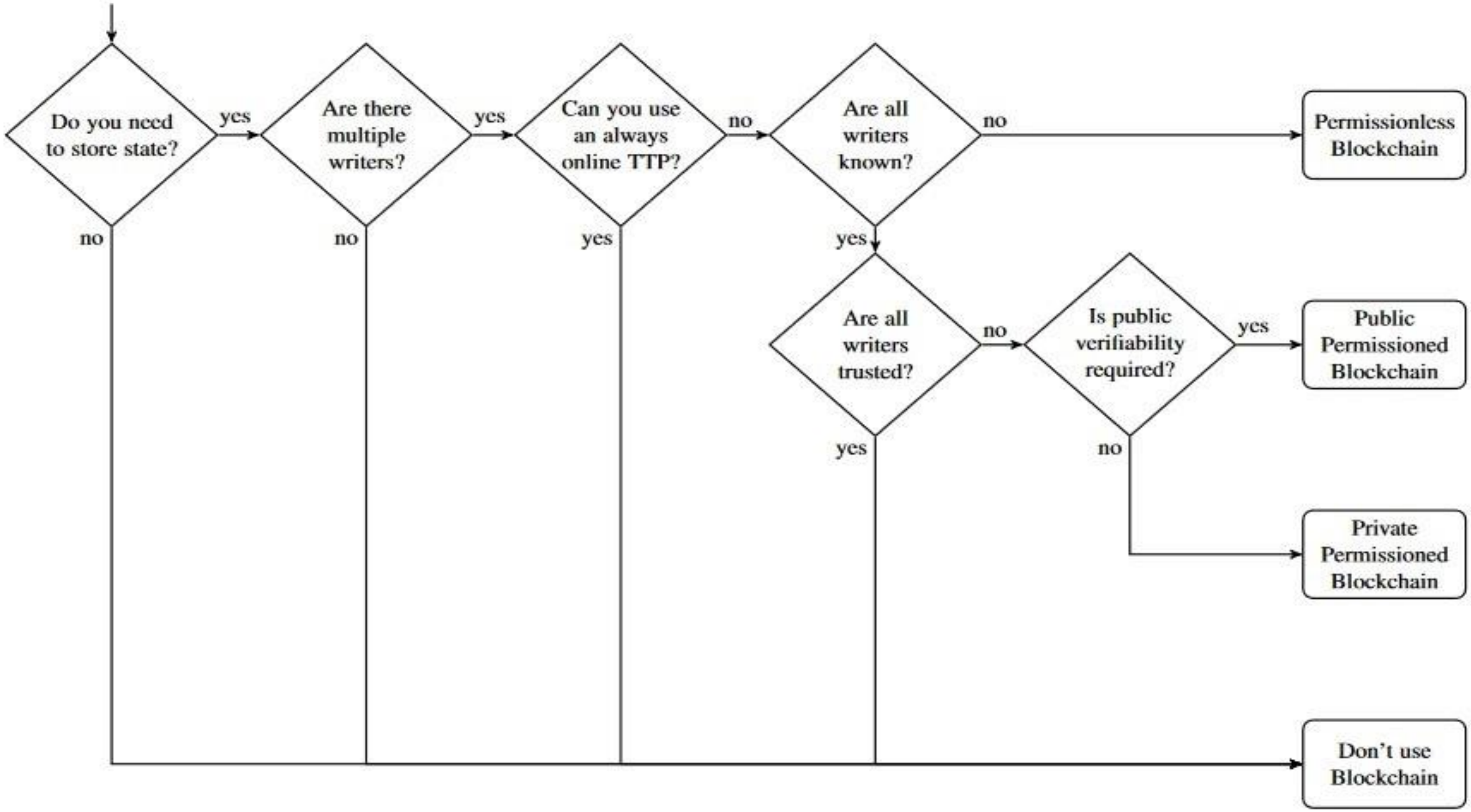
WHERE IS MY ETHER?

- Ether is not contained in any particular machine or application.
- Your ether balance can be queried, and ether sent or received, by any computer running an Ethereum node or wallet.
- Even if the computer where your Mist wallet lives gets destroyed, never fear: all you need is your private key, and voila, you can access your ether from another node.
- However, if you hand over your private keys to someone else, that person can access the EVM and pull your money out without you ever knowing.
- As far as the network is concerned, *anyone with your private key is you*.

PRIVATE AND PUBLIC ETHEREUM

- Ethereum is an **open-source public blockchain.**
- An Ethereum network is **a private network** if the nodes are not connected to the main network nodes.
- Private ethereum requires an **invitation or permission to join.**
- For entrepreneurs, it is preferable to build their network on top of the public Ethereum.
- **Permissioned blockchains** are the ones where corporate stakeholders are given certain rights and privileges to read and write to the company chain.
- For permissioned blockchains, **wallet addresses** are typically issued by a trusted third-party who verifies your permission to enter the system, just the way an office building’s security pass allows you to transact inside the building.

DO YOU NEED A BLOCKCHAIN?


Do you need to store state?
yes
no
Are there multiple writers?
yes
no
Can you use an always online TTP?
no
yes
Are all writers known?
no
yes
Permissionless Blockchain
Are all writers trusted?
no
yes
Is public verifiability required?
yes
no
Public Permissioned Blockchain
Private Permissioned Blockchain
Don't use Blockchain

DO YOU NEED A BLOCKCHAIN?

- Blockchain has been extensively used for various applications. But Blockchain may not be applicable for all scenarios. So, how should one determine whether blockchain is appropriate solution for a particular problem?
- In any case, if blockchain is an appropriate solution then should one opt for public blockchain or for private blockchain, permissioned or permissionless blockchain.
- Let us discuss step by step approach to determine whether blockchain fits in as best solution for given scenario.
- **DATA STORAGE**:
  - Data storage is the very first parameter while deciding for Blockchain solution.
  - Systems where data storage is not required, don’t need to opt for blockchain.
  - Storing data might be an essential requirement of many systems.

DO YOU NEED A BLOCKCHAIN?

- **PARTICIPANTS**
  - If data storage is essential, check who all are the writers / participants of the system.
  - Systems having multiple readers and single writer should prefer database.
  - Whenever there are multiple writers in a system, determine whether all the writers are well-known?
  - For systems having recognized and trusted writers, again database is better solution.
  - Or if a trusted third party is available online 24*7, go for database.
- Blockchain is efficient in dealing with the uncertainty of identity and trust.
- Hence, when writers are not well-known, go for Permissionless Blockchain.
- Whereas when writers are known and trusted, determine whether public verifiability is crucial.
- If yes, choose Public Permissioned Blockchain.
- Systems where public verifiability is not important, Private Permissioned Blockchain are best solutions.

DO YOU NEED A BLOCKCHAIN?

- **NETWORK REQUIREMENTS**
  - Network requirement is another dimension for determining whether Blockchain is the best solution.
  - Systems requiring high throughput, low latency and having only trusted writers should always go for database.
  - Throughput is comparatively very low in blockchain and latency is high in blockchain.
  - Therefore, systems that cannot tolerate delay should not opt for blockchain.
  - Private Blockchain can support high number of readers and untrusted writers whereas Permissioned blockchain is preferred for limited writers, trusted or untrusted