Study and Implementation of a Decentralized Application That Can Provide Permissionless Financial Services Using an EVM-Based Blockchain

Mario M. Orozco

Bachelor Colloquium

July 15, 2022

Study and Implementation of a **Decentralized Application** That Can Provide **Permissionless Financial Services** Using an EVM-Based **Blockchain**

Mario M. Orozco

Bachelor Colloquium

July 15, 2022

Contents

Goals

Blockchain

Ethereum

Decentralized Finance (DeFi)

Implementation

Demo

Conclusion

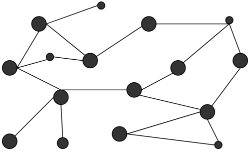
Questions

Goals

The **goals** of the Thesis were to provide a better understanding of:

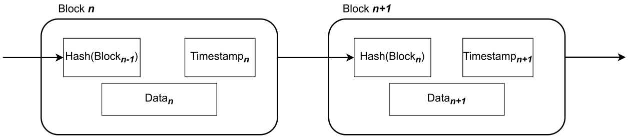
- The way that decentralized applications or **dapps** are **built**.
- The used **technology**.
- **Developing a Dapp** that uses well-known DeFi protocols, to let users access permissionless financial services such as lending and borrowing of crypto assets.

Blockchain



The term blockchain usually refers to a **public ledger** of transaction records, **shared and synchronized** by a **peer-to-peer network**, and a **consensus protocol**.

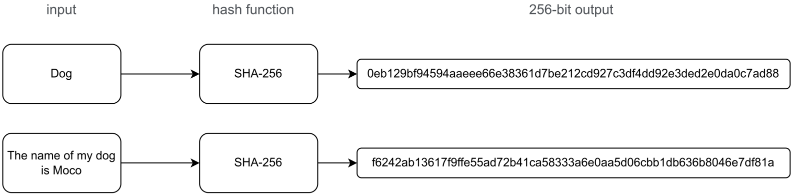
But what is a blockchain?



- Is a type of **database** (**append-only, read only** data structure), that stores data in 'blocks'.
- Blocks are chronologically ordered by discrete timestamps and linked to each other using **Cryptographic Hash Functions**.
- **CHF** and **Merkle Trees** ensure the integrity of the blockchain.

Cryptographic Hash Functions (CHF)

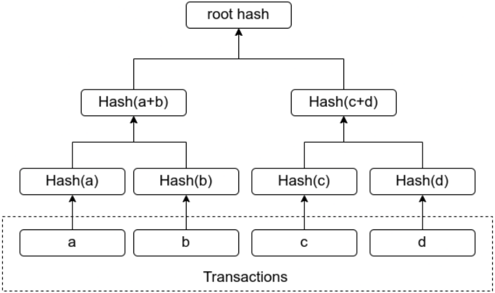
A cryptographic hash function maps a given data of **variable size to a fixed length** n-bit string *(hash value)*. $H : \{0,1\}^* \rightarrow \{0,1\}^n$.


input
hash function
256-bit output
Dog
SHA-256
0eb129bf94594aaeee66e38361d7be212cd927c3df4dd92e3ded2e0da0c7ad88
The name of my dog is Moco
SHA-256
f6242ab13617f9ffe55ad72b41ca58333a6e0aa5d06cbb1db636b8046e7df81a

CHF Properties

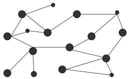
- **Collision resistance:** Computationally infeasible $O(2^{\frac{n}{2}})$ that different inputs, i.e., blocks of data, will generate the same hash value.
- **Preimage resistance:** Computationally infeasible $O(2^n)$ to retrieve the input data from its hash value (one-way function).
- **Second preimage resistance:** For a given hash value, it is computationally infeasible $O(2^n)$ to find another input, i.e., a block data input, that generates the same hash value.

Merkle Tree



- Binary Tree: Efficient Data Structure: $2log(N)$.
- Constructed bottom-up.

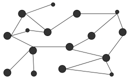
Consensus Protocol



Who can append a new data block?

- **Any participant** or node of the network, depending on the blockchain type, **can append a new block**.
- **Participants** on the blockchain **must verify** any transaction according to the set of rules or consensus protocol of the blockchain network.

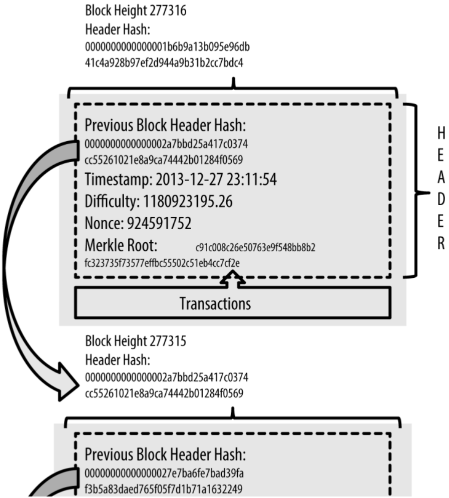
Consensus Protocol



The most common consensus mechanisms are:

- **Proof of work (PoW)**: Nodes have to solve a cryptographic task in order to validate a block; the first node to find a solution can submit the transaction.
- **Proof of state (PoS)**: The node that can validate a transaction is randomly selected, depending on the ”stake” that a node has on the network.

Bitcoin Example



Block Height 277316
Header Hash:
0000000000000001b6b9a13b095e96db
41c4a928b97ef2d944a9b31b2cc7bdc4

Previous Block Header Hash:
0000000000000002a7bbd25a417c0374
cc55261021e8a9ca74442b01284f0569
Timestamp: 2013-12-27 23:11:54
Difficulty: 1180923195.26
Nonce: 924591752
Merkle Root: c91c008c26e50763e9f548bb8b2
fc323735f73577effbc55502c51eb4cc7cf2e

Transactions

Block Height 277315
Header Hash:
0000000000000002a7bbd25a417c0374
cc55261021e8a9ca74442b01284f0569

Previous Block Header Hash:
0000000000000027e7ba6fe7bad39fa
f3b5a83daed765f05f7d1b71a1632249

SHA-256,
POW

Bitcoin

The distributed computation system that bitcoin introduced: a **proof-of-work** algorithm to produce a consensus in a distributed system without a central trusted authority, in combination with the blockchain storage system, solved the **double-spend problem** and the **Byzantine generals problem**.

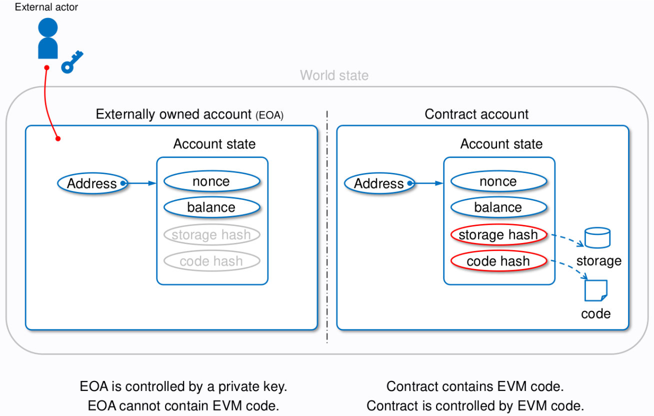
Ethereum

Ethereum is a decentralized computing platform or a **decentralized pseudo-turning complete virtual machine**, which runs **smart contracts** and stores its state changes in its blockchain.

Ethereum has an account-based model

- Every **account** represents a **state**.
- An account is **mapped with** a 160 bits address (**public key**).
- **Two types** of accounts: Externally owned account (**EOA**), controlled by a private key and (smart) **contract account**, controlled by EVM code.

Ethereum has an account-based model



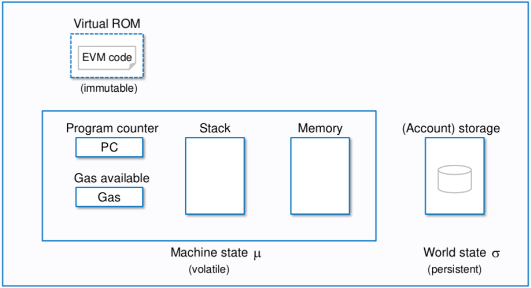
EOA is controlled by a private key.
EOA cannot contain EVM code.

Contract contains EVM code.
Contract is controlled by EVM code.

Transactions and Messages

- Two types of transactions: **Message calls**, **Contract creations**.
- **Messages** can be seen as **internal transactions between contracts** triggered by a message call.

Ethereum Virtual Machine (EVM)



- A quasi-Turing-complete machine.
- The quasi means that its computation is limited by the available **gas** that the executed contract has.

Ether and Gas

- Ether (ETH) is the native cryptocurrency of Ethereum.
- ETH is used to pay for transaction fees or EVM computing power in the form of gas units.
- The gas unit is represented in *gwei*, and is set by the participants of the consensus protocol (miners), based on the supply and demand of the network computational power.
- For instance, with an average gas price of 15 gwei, and $G_{transaction} = 21000$, sending a transaction would cost: $21000 * 15 = 315000$ *Gwei* or 0.000315 *Ether*.

Unit Name	**Wei Value**
Ether	1^{18}
Gwei	1^{9}
Wei	1

Smart Contracts

- A **computer program** executed by the EVM.
- Smart Contracts **have a balance and can send transactions** in the form of messages.
- Characteristics: **Immutability**, **Determinism**, **Limitations**, **Lifecycle**, **Composability**, **Permissionless**.

Solidity

- Solidity is an object-oriented or contract-oriented high-level programming Language, designed to **target the EVM**.
- The **Solidity compiler** converts Solidity contracts into **EVM bytecode** and generates the contract **Application Binary Interface (ABI)**, which enables the interaction with contracts.

Tokens

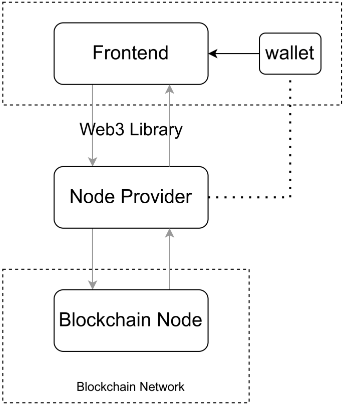
- In the context of Ethereum, a token is a **digital representation** or abstraction of something that lives on the Blockchain.
- For instance, tokens can represent **currencies**, **shares in a company**, or even a **virtual pet**.
- Unlike Ether, which is managed by the Ethereum protocol, tokens are **created and handled by smart contracts**.
- Everyone can create a token. However, some **standards** need to be followed **for the creation of tokens**.

Tokens

- **ERC20** Token Standard, for the representation of **fungible tokens**. For instance, a token that represents a currency or a share in a company.

- The **ERC721** Token Standard, for **non-fungible tokens** or representation of unique goods like an ID or collectibles.

Decentralized Applications (Dapps)



- A dapp has its backend running on a decentralized peer-to-peer network like Ethereum.
- The backend is smart contracts executed by the EVM→same properties as smart contracts.

Decentralized Finance (DeFi)

- Decentralized finance or DeFi refers to a set of **dapps** that are **focused on financial services**.
- The term finance involves the creation, and management of money, in **traditional finance systems** this is done by financial institutions(Banks), which emit, buy and sell financial instruments(cash, bonds, loans...) on financial markets(exchanges), all **regulated by Laws**.
- In **DeFi**, these practices and processes are determined by protocols that **rely on smart contracts**, which means Defi is an open, permissionless, and composable stack of protocols built on top of a blockchain such as Ethereum.

Decentralized Finance (DeFi)

- DeFi has gained a lot of traction in the past years and according to Defipulse, the Total Locked Value or TVL (held by smart contracts) in all the DeFi applications and protocols today is:
  TOTAL VALUE LOCKED (USD)

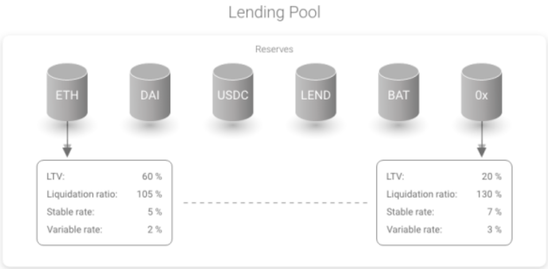
Maker Protocol

- One **essential piece of DeFi is stablecoins**.
- **Stablecoins** are cryptocurrencies **pegged** to a **predetermined value**, usually to the USD value.
- Stablecoins enable DeFi users to access more traditional assets, **avoid**ing the natural **volatility of crypto assets**.
- The **Maker Protocol** is a project built on Ethereum that **allows users to generate the stablecoin DAI**.
- DAI is **pegged to the USD** value and collateral backed by different crypto assets authorized by the MakerDao, for instance, ETH.
- Example: The Maker ETH-C Vault has a minimum collateral ratio of 170%, meaning that for every $170 of ETH deposit in the Vault a maximum of 100 DAI can be generated.

Aave Protocol

- Aave is a DeFi liquidity protocol that **enables users to lend and borrow** crypto assets.
- Users that supply liquidity to the protocol (**lenders**) **earn interest** on their deposited assets, and **borrowers** are able to **borrow crypto assets after depositing collateral** to the pool contract.
- The **interest rate** for users (borrowers and lenders) is decided **algorithmically based on the reserves** available in the pool.

Aave Protocol



- The **reserves** are the multiple currencies deposited on the pool expressed in ETH value.

Aave Protocol: Rates and Risk Parameters

- The interest rate, for lenders and borrowers, is determined by the status of the specific reserve.
- For **lenders**, the interest rate depends on the *current liquidity rate*: $R_t = R_o U$, where R_o is the overall **borrow rate** of the reserve.

$$R_o = \begin{cases} 0 \quad if \quad B_t = 0 \\ \frac{B_v R_v + B_s R_{sa}}{B_t} \quad if \quad B_t > 0 \end{cases} \tag{1}$$

Aave Protocol: Rates and Risk Parameters

With the total amount of borrowed liquidity B_t (*total borrows*), expressed as the sum of the total stable borrows plus the total variable borrows :

$$B_t = B_s + B_v$$

And U is the *utilization rate*, which is the representation of the utilization of the deposited assets, defined as follows:

$$U = \begin{cases} 0 \quad if \quad L_t = 0 \\ \frac{B_t}{L_t} \quad if \quad L_t > 0 \end{cases} \tag{2}$$

Where L_t is the *total liquidity* available in the reserve.

Aave Protocol: Rates and Risk Parameters

Loan-To-Value (LTV)

- A user will be able to borrow a maximum amount depending on the Loan-To-Value of the desired asset reserve.
- For instance, if the LTV value of a given asset is 60%, for every 1 ETH value of collateral, a user can borrow a maximum of 0.6 ETH value of the desired asset reserve.

Aave Protocol: Rates and Risk Parameters

Liquidation Threshold

- The percentage at which a borrow position can be liquidated.
- For instance, if the liquidation threshold of a given asset is 80% and the borrow position value surpasses 80% of the collateral value, the position can be liquidated.

Aave Protocol: Rates and Risk Parameters

Health Factor

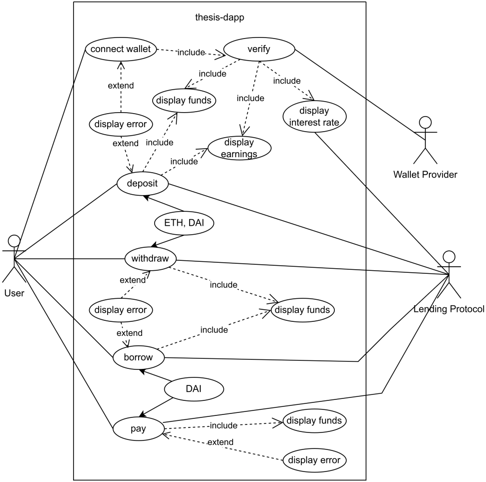
- Indicates if the borrow position of the user can be liquidated. If $H_f < 1$, the borrow position can be liquidated.

$$H_f = \frac{\sum Collateral_i \ in \ ETH \ \times LiquidationThreshold_i}{Total \ Borrows \ in \ ETH}$$

Aave Protocol: Tokenization, aTokens

- Aave Tokens or **aTokens** are ERC20 tokens that lenders, and borrowers, receive once the deposit or borrow transaction is processed.
- aTokens **maps 1:1**, the deposited or borrowed asset, also known as the underlying asset.
- For instance, if a user deposits 100 DAI, 100 aDAI is sent to its account.
- For lenders, the balance of a specific aTokens grows depending on the interest rate of the underlying asset.

Implementation



Live Demo

:)

Conclusion

conclusions

Questions

Thanks!