**DECENTRALIZATION APPLICATION FOR SECURE MESSAGE**

Blockchain

-----------------

Blockchain Technology

- In simplest terms, blockchain is a mechanism for storing digital data.

- The data can literally be anything.

- The data can even be files, it doesn't matter.

- In the case of Bitcoin, it is the transactions (transfers of Bitcoin from one account to another).

- The data is stored in the form of blocks, which are chained together using hashes.

- Storing data in BLOCKs + using hashes to CHAIN them together = blockchain

Characteristics of Blockchain Networks

- All of the "magic" in blockchain comes from the way this data is added and stored in the blockchain.

- This yields some highly desirable and powerful characteristics:

- Immutability of history

- Un-hackability of the system

- Persistence of the data

- No single point of failure

Public blockchain

- A public blockchain network is completely decentralized and open to the public.

- No one entity has control over the network and they are secure in that data cannot be changed once validated on the blockchain.

- Anyone can join and participate.

- Examples: Bitcoin, Ethereum

Private blockchain

- A private blockchain network is primarily used by businesses who need greater privacy, security, and speed of transactions.

- Participants need an invitation to join.

- They operate quite similarly to public blockchains but have access controls that limit who can participate in the network.

- It operates like modern centralized database systems that restrict access to certain users.

- One or more entities control the network.

- Causes users to still have to rely on third-parties to transact.

- Example: Ripple, Hyperledger

Brief History of Bitcoin

- In 2008, a whitepaper was released by an individual or group under the identifier Satoshi Nakamoto.

- Titled "Bitcoin: A Peer-to-Peer Electronic Cash System."

- The paper combined cryptographic techniques and a peer-to-peer network without the need to trust a centralized authority to make payments from one person to another.

- It also introduced a distributed system of storing data (blockchain).

- We all now know this concept has far wider applicability than just payments, or cryptocurrencies.

- Blockchain technology has exploded across nearly every industry.

- It is now the underlying technology behind:

- Fully digital cryptocurrencies (i.e., Bitcoin)

- Distributed computing technologies (i.e., Ethereum)

- Open-source frameworks (i.e., Hyperledger Fabric)

Development

-----------

1. Store transactions into blocks

- I will be storing the data in JSON, a widely-used format.

- The generic term "data" is often used interchangeably with the term "transactions" on the Internet.

- The transactions in the application are packed into blocks.

- A block can contain one or many transactions.

- The blocks containing the transactions are generated frequently and added to the blockchain.

- Each block will have a unique ID, since there can be multiple blocks.

2. Make the blocks immutable

- I want to detect any kind of tampering in the data stored inside the block.

- In blockchain technology, this is accomplished using a hash function.

- It is a function that takes data of any size and produces data of a fixed sizes from it, which generally works to identify the input.

- The Python Standard Library has a hashlib library with a SHA-256 and SHA-512 hashing function.

- The characteristics of an ideal hash function are:

- It should be computationally easy to compute.

- Even a single bit change in data should make the hash change altogether.

- It should not be possible to guess the input from the output hash.

- I will store the hash of every block in a field inside a Block object to act like a digital fingerprint of data contained in it.

- Note: In most cryptocurrencies, the individual transactions in the block are also hashed, to form a hash tree, and the root of the tree might be used as the hash of the block.

- However, it is not a necessary requirement for the functioning of the blockchain.

3. Chain the blocks

- The blocks themselves are now set up.

- The blockchain is a collection of blocks, and I must implement it accordingly.

- I could store all of the blocks in a list (array) in Python, but it would not work.

- It is not sufficient.

- Someone could intentionally replace a block at a previous index in the collection/list.

- In the current (unfinished) implementation, creating a new block with altered transactions, computing the hash, and replacing it with any older block works and it should not.

- I must maintain the immutability and order of the blocks in some way.

- I need a way to ensure that any change in the past blocks invalidates the entire chain.

- One way to do this is to chain the blocks by the hash.

- By chaining, I mean to include the hash of the previous block in the current block.

- If the content of any of the previous blocks change, the hash of the block would change, which would lead to a mismatch with the previous\_hash field in the next block.

- If every block will be linked to the previous block by the previous\_hash field, I must manually generate the very first block ourselves.

- The very first block is called the genesis block, and it is generated manually or by some unique logic, in most cases.

4. Implementing a proof of work algorithm

- Selective endorsement vs. proof of work

- Consensus in a (private) blockchain for business is not achieved through mining, but through a process called selective endorsement.

- The network members control exactly who verifies transactions, much in the same way that business happens today.

- A problem arises: if I change the previous block, I can re-compute the hashes of all the following blocks quite easily and create a different valid blockchain.

- To prevent this, I must make the task of calculating the hash difficult and random.

- Instead of accepting any hash for the block, I will add some constraint to it.

- Let's add a constraint that the hash should start with a certain number of leading zeroes.

- I also know that unless I change the contents of the block, the hash will not change.

- I will introduce a new field in the Block, a nonce.

- A nonce is a number that will continue to change until there is a hash that satisfies the constraint.

- The number of leading zeroes, which will default to 2, decides the difficulty of the PoW algorithm.

- This PoW algorithm is difficult to compute but easy to verify once I figure out the nonce.

- Verifying will just involve running the hash function again.

5. Adding blocks to the chain, and mining

- In order to add blocks to the chain, I must first verify two components:

- The PoW that is provided is correct.

- The previous\_hash field of the block to be added points to the hash of the latest block in the chain.

- At this point, I must implement a mechanism for mining the blocks.

- The transactions are initially stored in a pool of unconfirmed transactions.

- The process of putting the unconfirmed transactions in a block and computing PoW is known as mining the blocks.

- Once the nonce satisfying the constraints is figured out, I can say that a block has been mined.

- At that point, the block is put into the blockchain.

- In most cryptocurrencies, miners may be awarded some cryptocurrency as a reward for spending their computing power to compute PoW.

6. Creating interfaces for the Flask web app

- I must create interfaces for the node to interact with other peers as well as with the application.

- I will be building it with the Flask web framework to create a REST-API to interact with the node.

- I need an endpoint for the app to submit a new transaction.

- It will be used by the app to add new data (posts) to the blockchain.

- I also need an edpoint to return the node's copy of the chain.

- It will be used to query all of the posts to display to the user.

- I also need an endpoint to request the node the mine the unconfirmed transactions (if any).

- It will be used to initiate a command to mine from the app itself.

- I also will add an endpoint to query the unconfirmed transactions.

- At this point, I have a functioning blockchain, where I can create new transactions (posts), and mine them to add them to the blockchain.

- However, the codebase at this point is meant to run on a single computer.

- I will need to add functionality to have multiple nodes to maintain the blockchain.

7. Establishing consensus and decentralization

- Even though I am linking blocks with hashes, still cannot trust a single entity.

- I will need multiple nodes to maintain the blockchain.

- I must create an endpoint to let a node know of other peers in the network.

- I must also create an endpoint to add new peers to the network.

- There is a problem with multiple nodes.

- Due to intentional manipulation or unintentional reasons, the copy of chains of a few nodes can differ.

- In that case, there must be an agreement upon some version of the chain.

- This is known as consensus, which must be achieved to maintain the integrity of the entire system

- A simple consensus algorithm could be to agree upon the longest valid chain when the chains of different participants in the network appear to diverge.

- The rationale behind this approach is that the longest chain is a good estimate of the most amount of work done.

- I also need to develop a way for any node to announce to the network that it has mined a block so that everyone can update their blockchain, and move on to mine other transactions.

- This involves creating another endpoint to add a block mined by a user to the node's chain.

- After every block is mined by the node, it should be announced, so that peers can then add it to their chains.

- Other nodes can simply verify the proof of work and add it to their respective chains.

8. Building the application

- At this point, the backend is all set up.

- I'll build an interface for the application, which is a view in the codebase.

- Using Flask, I'll use Jinja2 templates to render the web pages and some CSS for styling.

- The application needs to connect to a node in the blockchain network to fetch the data and submit new data.

- There can also be multiple nodes.

- The application has an HTML form to take user input, and then makes a POST request to a connected node to add the transaction into the unconfirmed transactions pool.

- The transaction is then mined by the network, and then finally will be fetched once the website is refreshed.