NEA Documentation

# Analysis

Project Introduction

My project is a blockchain data structure and network system which solves many issues. When you use a currency like British Pounds or US Dollars, you are trusting a third party. The supply and value of these currencies are heavily affected by the respective government’s own control and monetary policy decisions. These currencies are regulated and issued by the respective governments. This can be an issue as this means a central authority has power to influence money supply, interest rates, and any other aspects of the currency’s operation. This is called centralisation, when a currency is managed by a central authority like a government and its bank associates. These central authorities can print more money or withdraw money from circulation, controlling the currency’s inflation rates.

My proposed program solves this problem by getting rid of the idea of a central authority controlling a currency. Blockchain makes currency decentralised, removing the element of trust in a third party. This allows for currency to operate on a distributed ledger, not governed by a central authority. The concept of a distributed ledger allows for this because anyone who wants to make transactions can make transactions without the need for a central intermediary. The blockchain is essentially a public record of transactions, and there are nodes, which anyone can operate, that carry a copy of the public record of transactions. Nodes can create transactions and add the transaction to their copy of the blockchain, then broadcast it onto the blockchain network for all the other peer nodes to pick up on, and add the transactions to their copies of the blockchain individually, so that all nodes are up to date and have the same copy of the blockchain. This whole process is done without the need of a central authority to regulate any transactions. There are many intricacies to this, as it sounds as though a lot of trust is still required, since malicious nodes could ruin the integrity and security of the blockchain, but there is a lot of mathematics behind blockchain that make it near impossible to make transactions in other user’s names, make transactions outside one’s balance, change transactions once they are added to the blockchain and broadcasted to all other nodes on the network, and also allows for privacy of users, which I will go into later in this document

Research

There are many existing system’s that demonstrate blockchain technology, the two I have researched are Bitcoin and Ether, the cryptocurrency belonging to Ethereum. Bitcoin was created as a digital currency, for the store of value and a medium of exchange. The focus of bitcoin is to provide a decentralised and secure way to transfer and store value, in the form of bitcoin cryptocurrency. Ethereum is also blockchain technology, but it has a much wider application, I will be focusing on its cryptocurrency, Ether, which has the same purposes as Bitcoin but through slightly different methods.

Transactions

The Bitcoin network operates similarly to how I mentioned above, it is a network, where individual nodes have their own copies of the blockchain, and can add transactions to the blockchain. Bitcoin solves many issues of security and integrity that arise from having a decentralised system like this for transactions through a lot of different ways. Starting from the core, transactions. A node can create transactions but how does Bitcoin make sure people are making transactions in their own name? Bitcoin utilises the RSA encryption algorithm, this algorithm generates a public and private key pair that are mathematically linked. Each user on the network has a public and private key. A user is represented on the network and referenced by their wallet address, which is derived from the public key. (in simpler systems, the public key can be used to represent a user on the blockchain, but in Bitcoin the wallet address is the public key put into a more standardised format, as a wallet can make transactions with different cryptocurrencies, not just Bitcoin, and different blockchain systems may generate the public key in different ways, lengths and such. Also RSA typically returns the public key in hexadecimal or binary string format, which is difficult to type out for users). When a transaction is made by a user, they must authenticate the transaction using a digital signature. A transaction includes the public key or wallet address, the amount being send, the public key or wallet address of the recipient receiving the transferred cryptocurrency, and a digital signature, which is made with the private key of the sender, and the cryptographically hashed contents of the transaction. (Bitcoin and blockchains in general use cryptographic hashing, specifically SHA-256, often for data as it is computationally deemed irreversible) The transaction is then broadcasted to the network and is added to the mempool, short for memory pool, a list of unconfirmed transactions (a transaction is confirmed when it is incorporated into the blockchain for long enough – more on this later). When a node picks up on the broadcast to add it to the transaction pool, it verifies the transaction. The public key and private key are mathematically linked in a way, due to RSA encryption to generate the key pairs, such that you can use the public key to verify a digital signature using a verification algorithm. It goes like this, the node takes the transaction, hashes it, and applies the public key on the digital signature using the verification algorithm (think of it like the public key is undoing the effect the private key has on the hashed transaction, because they are like inverses of each other). The node then compares the result to the hashed transaction, and if they are the same, then the user is in hold of the private key. This verifies and authenticates a transaction and is called asymmetric encryption, the node now just has to validate the transaction by making sure the user has sufficient funds by searching their transaction history in the blockchain, since the blockchain stores all transactions too. This process assumes the user keeps their private key private, as the holder of the private key can create transactions in the name of the public key linked to it. Although they are mathematically linked, the private key cannot be derived from the public key without infeasible computational power, due to the nature of the RSA encryption algorithm of which the mathematics are explained in more detail later.

Mining (proof-of-work)

Once a node verifies and validates a transaction, they add it to the transaction pool and also broadcast it to all other nodes, for all other nodes to validate the transaction individually, so that they can add it to their individual transaction pools, since the network needs to maintain equality. Decentralisation shows heavily here, as all nodes need to validate the transaction for themselves, they cannot just trust another node since anyone can run a node. Once a transaction in in the transaction pool it can be picked up by a specialised node called a miner. This is where Bitcoin begins to differ from Ether. All blockchains have their own set of rules that each node needs to maintain for the distributed ledger system to work. How the actual blockchain works is that the blockchain is a list of blocks, where each block is linked to the block before it. Each block contains transaction data, and some other identifying metadata, this is called the block header. The block header is hashed, this hash represents the block. The hash is then included in the next block. A block contains the hash of the previous block in its own hash calculation, linking the blocks together in a chronological fashion. A diagram of a blockchain

Description automatically generated

These contents of the block header are used to calculate the hash that represents the block

If you go back and change the contents of a block, for example its transaction data, you will have to recalculate the hash of every block after it, because if you change the transaction data of a block, the hash of the block changes, and this changes the hash of the next block since the hash of the next block uses the hash of the previous block to calculate its hash. This isn’t too computationally heavy, making it easy to rewrite history essentially, which is where Bitcoin’s Proof-of-work consensus comes into play. A miner node picks up transactions out of the transaction pool and creates a block with it. If a miner could create a block quickly, it would be easy to go back and rewrite the entire blockchain, but a miner cant create a block quickly because they have to solve a hash puzzle. There is a value in each block called the nonce (short for number only used once) and the blockchain network will have a difficulty target, a value which determines how difficult it is to solve the hash puzzle. The hash puzzle is essentially a certain amount of 0s the binary string of the block hash must begin with. There is no way to solve this other than guessing random values for the nonce which will keep changing the hash value, until the hash value of the block meets this requirement of 0s set by the difficulty target, this is the computationally expensive part of being a miner node. Once a miner meets the requirement, they are then allowed to create a block and add it onto the blockchain, and broadcast it to the network, allowing for other nodes to add the block onto their blockchain (after validating the block). In Bitcoin specifically, a miner node is incentivised to add blocks because creating blocks will reward you with a bitcoin reward, and transactions have their own transaction fees which are rewarded to the miner.

Immutability of Blockchain (attacks on Blockchain)

A malicious node will attempt to go back in the blockchain and change the contents of a block, but doing that means they will have to remine every single block in the chain. The history of the blockchain is already set, attempting to remine old blocks due to changing their contents will create a fork in the chain. Imagine the chain of blocks, and a fork coming off the first block to be changed with a list of blocks ahead of it that have been newly created (they are the remined blocks). Another part of the consensus of bitcoin is that if there is a fork, the one with the most computational work put in it is accepted as the blockchain. This means that the malicious node will have to remine the blocks at a rate faster than the creation of all blocks by all the other nodes together. This is called a 51% attack, because for a malicious node to successfully win the fork, they need the majority of the computational power of all nodes on the network, because if they don’t have that, then the rate at which nodes are adding blocks onto the main chain will be too fast and the shorter fork prong will not be deemed as the real blockchain. If a malicious node successfully remines blocks fast enough, it will have been able to change transaction history, and is now able to double spend currency that has already been transferred. As talked about earlier, a transaction is deemed confirmed in the blockchain once it has a transaction count of 6, the transaction count is the depth of the transaction in the blockchain, how many blocks deep the transaction is in. Bitcoin says that after 6 blocks it is confirmed as a transaction because it is unlikely that it is going to be able to get changed. A reminder that what is going on in the copy of the blockchain held by one node is not instantly regarded as the overall blockchain, since all other nodes have their own copy and validate transactions and blocks broadcasted from other nodes before adding it to their own copy, increasing the security and integrity of the decentralised distributed ledger of the blockchain network. Bitcoin is decentralised but it has lost its meaning slightly because there are a few nodes that are majorly responsible for the mining computational power, and when a lot of transactions from users are broadcasted to a select few nodes so that they broadcast the information to all other nodes, it contributes to centralisation.

Efficient Data Storage

Another component of blockchain that both Bitcoin and Ether have in common is the use of a data structure called Merkle Trees. When peer nodes in the network receives a block broadcasted from a miner node, they must validate the block and make sure it follows the rules of the blockchain network, they validate its Proof-of-Work in Bitcoin (difficulty target set by then network is reached), and Proof-of-Stake in Ether, a different consensus algorithm. After they verify the proof of work, they may add a block to the chain. Transactions are stored in the block, but a block header also includes something called a Merkle Root. All transactions in a block are organised in a Merkle Tree, a binary tree data structure where the leaf nodes are occupied by all the transactions, and each node above is the hash of the concatenation of the previous two nodes, called child nodes. The parent nodes are then paired up and used as the child nodes for the next level of the binary tree, the same process of concatenation and hashing. Each level is half the length of the previous level from the leaf node up to the point where there is only one node remaining in the tree, called the root. In this context of trees this is called the Merkle Root, which represents all the transactions without showing any transactions. Merkle Trees can be used for many purposes, in the context of Bitcoin it allows for a few things. One is efficient verification that a certain transaction is included in a block. Instead of searching the entire dataset of transactions, parties can use a Merkle Proof to verify that a specific transaction is in the tree. A transaction is stored in a leaf node of the tree, and there are a set of nodes related to this transaction, they are the nodes directly concatenating with the child nodes originating from the transaction. A Merkle proof is the set of sibling nodes that are directly concatenated with the transaction or parent node of the transaction for the hash of future nodes. In short, the Merkle proof is the path of nodes from the transaction in question to the Merkle root. A diagram of a structure

Description automatically generatedblue – set of nodes in Merkle proof, green – transaction in question

The transaction is then proved after concatenating and hashing to reach Merkle root, it is proved if the Merkle root generated from the set of nodes in the Merkle proof is the same as the Merkle root included in the block, because it means the transaction was a part of the original tree structure.

This process to confirm inclusion of data in the tree is much more efficient than searching the whole tree, with a time complexity of log(n) compared to the time complexity of n from just searching the tree. The nodes in Merkle trees are hashed for the immutability of blockchain, because of the nature of hashing algorithms, when some data is tampered with, the Merkle root included in the block header will change, requiring the Merkle root to be recalculated and the block would have to be remined. To be specific, transactions are represented by the transaction ID, the hash of the contents of a transaction object, this is how transactions are referenced in the blockchain, for the immutability of the transactions.

Ethereum Differences (proof-of-stake)

Ethereum does things slightly differently, it does not use a Proof-of-Work consensus algorithm, (mining being a hash puzzle, nonce values, etc) but it uses a Proof-of-Stake consensus algorithm. A consensus algorithm is used to reach agreement on how blocks should be added and validated to the blockchain, so that all nodes on the network can operate in a decentralised fashion but still follow the same rules to maintain the distributed ledger. In proof of stake, validators are responsible for confirming and adding transactions to the blockchain. Who validates the block is based off which node holds the most stake in the network, the more cryptocurrency the wallet of a node holds, the higher stake they have, the higher the chances of being selected as the validator of the block. This stake is temporarily locked in a smart contract as collateral, meaning it cannot be moved or spent or withdrawn in any way until they leave the blockchain network. This prevents malicious nodes as malicious nodes are penalised for any malicious actions and behaviour, in the form of slashing. This refers to portions of their stake being confiscated as a penalty of malicious actions. The incentive for a node to increase their stake in the blockchain and

What I am using from these

* Proof of work consensus algorithm
* Merkle tree data structure
* RSA encryption
* Asymmetric encryption
* SHA-256 hashing algorithm
* Bitcoin consensus algorithm (transaction count, fork resolution, etc)