ANALYSIS

Blockchain data structure

The blockchain is just a data structure, like any other, it contains data. It was designed to contain transaction data for a new form of digital currency. This allows for a decentralised currency system as there is no one governing body managing transactions, anyone can collect transaction data and create a block to add to the blockchain. How does the data structure work?

This data structure is a linked list of blocks that contains various metadata like datetime (which is included to maintain a chronological order, and also provide some security as the data structure will be protected from “replay attacks”, when someone attempts to add an old block back into the blockchain, a form of fraudulent attack), the transaction data, its own hash, a fixed length value that is unreadable and essentially looks like nonsense, but it is generated algorithmically, meaning if you input the same bits of information into the hashing function, it will return the same value every time. In blockchain, usually the hashing algorithm SHA-256 is used, because the algorithm is irreversible, partially because it returns a 256-bit length string in hexadecimal, no matter the length of the input. It also helps decrease the chance of hash collisions due to the long output length (hash collisions are when different inputs result in the same output but these are rare occurrences). The block also contains the hash of the previous block and this is included in the hashing process for the signature hash of the block. Each block is linked to the previous block by containing its hash value, therefore the blockchain data structure is essentially a linked list of hash pointers.

When you are collecting transaction data to put inside a block, and you are about the compute the hash of the block, there is a rule you must follow and everyone must follow when creating a block to add to the blockchain. The hash of the block must follow a pattern, like it begins with eight zero’s, or ten zero’s, or any amount as long as it is the same rule for everybody. This means you must find an additional value, which is called the nonce value, to add to the block so that it changes the hash to follow this rule. There is no way of finding this apart from guessing, due to the nature of a hashing algorithm like SHA-256, so you find the nonce by searching and checking through every number from 0, until the hash returns a value that meets the criteria. This process is also called mining and can take a lot of computational power.

This data structure provides security in the sense that it is computationally immutable, meaning you would need a lot of computational power to be able to change data a block far down the blockchain. This is because of the nonce, as changing data in a block would change the hash, meaning you would have to solve the hash puzzle again, in other words, mine it. Then because the block now has a new hash, the next block’s hash changes because it was initially generated from a value including the old hash of the previous block, so you will have to mine the next block as well since its hash will change due to the previous hash included in the hashing process has changed. This creates a chain of events where if you wanted to mutate any part of a block, you will have to remine that block and every single block after it. If you cannot match the rate of new blocks being added to the blockchain then you would never finish remining all the blocks, and this means you would first have to catch up with the new blocks being added then you would more than 50% of the computational power of all the miners (people creating new blocks) to create blocks at a faster rate. More on why in section: Blockchain Networks

User authentication

When a user creates a wallet, they generate a private key and a public key. The public key represents them on the blockchain and everyone can see this, the private key is supposed to be kept private though. When a transaction is made, to combat the fact that someone could make a transaction in someone else’s name fraudulently, it is signed with a digital signature. This digital signature is made using the hashed version of the transaction data, encrypting this with the private key, authenticating the sender, then the receiver decrypts the digital signature with the public key, and it should return the hash of the transaction data, verifying the digital signature, meaning the sender is really the owner of the public key they claim they are sending from. This is a form of asymmetric encryption and the encryption algorithm used to generate these public and private keys is usually RSA algorithm. This protects the system from fraudulent transactions as you cannot send a transaction in someone else’s name without their private key.

Node responsibilities

Because the blockchain allows for a decentralised system, all nodes – computers connected to the blockchain network that have a copy of the blockchain history and the capability to mine (solve the hash puzzle to create a block) -must follow the same rules and regulations to avoid inconsistency and therefore insecurity. Each node must participate in maintaining the network’s ledger, this is done using a consensus mechanism. This is essential for decentralised networks as there is not central authority that can validate transactions. The consensus mechanism most blockchain networks use is Proof of Work, meaning that a node must solve complex mathematical equations such as the hash puzzle to validate a block, and in return they are rewarded with the currency the decentralised network uses, the cryptocurrency.

Another issue is the sender paying too much for how much they own. One way to combat this is using the UTXO model, where every time a transaction is made and it is verified the owner of the public key has their wallet recorded onto a ledger. Every time they make a transaction, the balance deductions are recorded onto the ledger, every time their public key is used in a transaction as them as the receiver it is also recorded, this way the balance is kept track of for everyone. This ledger can be implemented using relational databases through SQL and various database management systems – IN PROGRESS

RSA and SHA-256 – IN PROGRESS

The maths behind the RSA Encryption algorithm is complex so first I will go through a simplified case, and then extend its definition to the full algorithm. First the algorithm chooses two large prime numbers, we will call p and q. They are to be kept secret and are chosen randomly. Now we find the product of p and q, and call this new value N. So pq = N. We now find the product of (p-1) and (q-1). This process is called finding the Phi (φ) of N, denoted by the function φ(N). This is Euler’s totient function of N. This means that if we had p and q it’s very easy to find φ(N), its algorithmic, whereas if we had φ(N) it would be very difficult to find p and q, depending on how large the prime numbers are. We now need another number, lets call it e, that is relatively prime to φ(N). What this means is finding a value that has no common factors with φ(N), for example if φ(N) was 10, a number relatively prime to 10 would be 3 as you cannot divide 3 or 10 by any number to gain an integer (other than 1). We are going to use this number e as the exponent we raise the public key to. Now we need to find another constant d, this value is going to represent the modular multiplicative inverse of e modulo φ(N). This is a lot to take in so we have to break this down. d is a number which allows this statement to be true – e\*d = 1 mod φ(N).

DESIGN

The base of the program is the data structure of the blockchain which will hold the token, in this case being transaction data with the base unit being 1 – think of some unit later. The data structure can be broken down into two classes, individual from each other. One is the block class, which will have attributes like data, the hash of the previous block which can be found by accessing the hash of the last block in the blockchain, since it is just a linked list of hash pointers it can be done using the index of the last position in the chain (-1), the nonce value which is assigned a value of 0 to begin with and is later put through a while loop to go check through every value (incrementing in 1) for the hash puzzle to be solved, the timestamp using the datetime library, the index of the block for order sake and to ease the process of dealing with forks in the blockchain, and the hash which is generated from all these attributes used as inputs. The block class will be able to use a function for hash generation, a function that returns the string representation of the block which is used in the hashing algorithm process for consistency among all the blocks as to what is determining the hash, a mine function which will run a while loop, incrementing the nonce value by 1 each iteration, under the condition of the first x digits of the hash output being 0s (hash puzzle – x variable is assigned in the blockchain class definition as the hash puzzle is individual to the blockchain), and a genesis block generation function, which will be the first block in a blockchain as it wont have a previous hash so it needs to be instantiated differently.

The blockchain class will have few components, it will have a chain attribute which will be a list that has an instance of the block class pre-appended with the genesis method applied to it, this will make sure the chain assignment includes the genesis block within the list already so that any blocks that are going to be appended to the list do not run into the issue of not having a previous hash attribute, due to it being the first block. The blockchain class will also have a proof-of-work attribute, this is the rule that every node has to follow when mining, so that they all mine under the same hash puzzle, for example this attribute might be ‘0’\*8 so that when the mine method is called on an instance of the block class, it knows under what condition the while loops runs, which will be – while the slice [x:] of the hash is not equal to the proof-of-work variable is True. You can check the history of the blockchain with a method and a pow (proof-of-work) and length variable return function.

The data the instances of the block class holds is transaction data, which will be held in a server and collected. To test the concept, instead of using sockets and implementing functioning servers, I will use a mock server class which will instantiate circular queues replicating the server and the order in which transactions will be collected due to a queue’s first-in-last-out nature, as nodes will collect the oldest transactions first. Apart from the queue data structure itself, the mock server class has the attributes of … . The methods applicable to this class will consist of the standard actions you can perform on a queue, including enqueue to append complete transactions to the mock server, dequeue to pop them out of the first index, but not peek as it does not really have a place in this context.

IN PROGRESS

* What attributes and methods does the user class entail
* How are public and private keys generated
* How does the transaction process take place on the network

The transactions have many components, the public key of a user who is sending the units in a transaction, a digital signature generated by the sender to authenticate the sender and allow the receiver to verify the sender’s public key, and the receiver’s public key. These public keys represent users on the blockchain network. Each user also has a private key which is mathematically linked to the public key. In a way such that if you encrypt some data with the private key, it can be decrypted with the public key. These public-private key pairs can be generated using the RSA encryption algorithm, which uses modular arithmetic, large prime numbers and relatively prime numbers generated from the Euler totient function, to mathematically generate a public and private key for a user. How these are used is, the sender will encrypt the transaction data with their private key, and send it to the receiver. The receiver then uses the sender’s public key to decrypt it, this confirms that the user who is claiming to be the owner of the public key, is really the owner as they have the private key that is linked to it (as long as the private key has successfully been kept safe from others). These actions of sending and receiving are done over the blockchain network using sockets, the network will collect these transaction requests and the node that is creating a block using this