Blockchain Basics
Coursera

Liza Dahiya

April 2020

Contents

1 Bitcoin & Blockchain

1.1 Bitcoin

Bitcoin is the most popular cryptocurrency. Its two major contributions include a continuously working digital currency system, and a model for *autonomous decentralized application technology* called the blockchain. This digital currency was introduced by a mysterious person, **Satoshi Nakamoto** (a pseudonym), around 2008-2009.

1.2 Blockchain

Blockchain enables peer-to-peer transfer of digital assests without the use of intermediaries. It was created to support famous cryptocurrency **Bitcoin**.

1.2.1 Benefits of Blockchain

- Blockchain doesn't cost any fee for any transactions. It is in a way equivalent to 'cutting out middle men fee' & eliminating platform for a 'match-making platform'.
- The blockchain database isn't stored in any single location, i.e. data is **decentralized** meaning the records it keeps are truly public and easily verifiable. No centralized version of this information exists for a hacker to corrupt.

- The blockchain is transparent so one can track the data if they want to

The blockchain is a **linked list** that contains data and a hash pointer that points to its previous block, hence creating the chain.
Hash Pointer also contain hash of the data inside the previous block along with the data of the address of its previous node.

1.2.2 Peer-to-Peer Network

The blockchain uses a special kind of network called **"peer-to-peer network"** which partitions its entire workload between participants, who are all equally privileged, called "peers".
One of the main uses of the peer-to-peer network is file sharing, also called **torrenting**.

1.2.3 The Gossip Protocol

The network follows the gossip protocol. Think of how gossip spreads. Suppose Alice sent 3 ETH (Ether) to Bob. The nodes nearest to her will get to know of this, and then they will tell the nodes closest to them, and then they will tell their neighbors, and this will keep on spreading out until everyone knows. Nodes are basically your nosy, annoying relatives.

1.2.4 Trust among a decentralized system

Functions of intermediaries are shifted to the periphery of peer participants in a blockchain infrastructure. Peers are not necessarily know to each other.

1. Validate
2. Verify
3. Confirm transactions

2 Blockchain Structure

Transaction form a basic element of a blockchain, many transactions together make a block, which combine together to form chain; next block is chosen by special peer nodes called **miners**.

2.1 Unspent Transaction Output

Bitcoin and many other protocols uses Unspent Transaction Output (UTXO's) to store data about transactions and user balances. These are a list of "unspent" Bitcoin amounts that have been sent to a user, but have not yet been sent from him/her. The sum of these outputs is the user's total balance. A UTXO contains:

- Unique Identifier of transactiion that created the UTXO

- Index of the UTXO in the transaction's output
- Value
- (Optional) Condition uder which the output can be spent

Example:

Alice gives Bob 1 BTC , and the system now recognizes that there is 1 BTC signed to Bob that he has not yet given to anyone else. If Bob already had 1 BTC, then his balance on the blockchain would be 1 BTC + 1 BTC. Bob's Bitcoin balance is the sum of all Bitcoin signed to him, similar to how all the fiat cash in Bob's leather wallet is the sum of all fiat cash given to him. If he wants to combine his two separate BTC, he must do so in another transaction, much like he needs to do if combining two $5 bills into a $10 bill.

2.2 Transaction

A transactions brings about a transfer in the value of blockchain. A transaction contains:

- Reference number of the current transaction
- Reference(s) to one or more input UTXO's
- Reference(s) to one or more output UTXO's, newly generated by the current transaction
- Total input amount and output amount

3 Basic Operations

There are two basic roles of participants in a blockchain: participants who initiate transfer of value by creating a transaction and participants called **miners**who pick on added work or computation to verify transactions, broadcast transaction, compete to claim the right to create a block, work on reaching consensus by validating the block, broadcasting the newly created block, and confirming transactions. For doing this they get a reward of some bitcoins.

' All valid transactions are added to a pool of pransaction, which are then miners select a pool of transactions to create a block. **Miners compete by solving a puzzle,** to add a block to the transaction. Solved block then gets broadcast. And after verification, a new block is added to the chain.

In case of Bitcoin, this puzzle is a computation of puzzle and the central processing unit (or CPU). This algorithm is called **Proof-of-work-protocol**. Hence, Transaction zero, index zero of the confirmed block is created by the miner of the block. It has a special UTXO and does not have any input UTXO. It is called the **coinbase transaction** that generates a minor's fees for the block creation.

3.1 Wallet

Wallet is a program that allows you to store and exchange your bitcoins. Since only you should be able to spend your bitcoins, each wallet is protected by a special cryptographic method that uses a unique pair of distinct but connected **keys**: a private and a public key.

The digital signature which is generated when a transaction request is encrypted with a private key; is a string of text resulting from your transaction request and your private key; therefore it cannot be used for other transactions.

3.1.1 Tracking Wallet Balance

The blockchain system doesn't keep track of account balances at all; it only records each and every transaction that is verified and approved.

To determine your wallet balance, you need to analyze and verify all the transactions that ever took place on the whole network connected to your wallet.

3.2 Loop Hole for Security

An attacker could send a transaction, wait for the counterpart to ship a product, and then send a reverse transaction back to his own account. In this case, some nodes could receive the second transaction before the first and therefore consider the initial payment transaction invalid, as the transaction inputs would be marked as already spent.

It's not secure to order the transactions by timestamp because it could easily be counterfeit. Therefore, there is no way to tell if a transaction happened before another, and this opens up the potential for fraud.

Note: A disagreement about which block represents the end of the chain tail opens up the potential for fraud again.

3.3 Mining Bitcoins

The activity of running the bitcoin blockchain software in order to obtain these bitcoin rewards is called "**mining**" — and it's very much like mining gold.

4 Beyond Bitcoin

There are majorly three types of blockchain:

- Coins & cryptocurrency; (bitcoin)
- Cryptocurrency & Business logic; (Ethereum/ smart contract)
- Software support for business logic

It can also be classified as private, public, permissioned or consortium blockchain.

5 Smart Contracts: Ethereum

Ethereum supports smart contract and virtual machines on which smart contracts execute. A smart contract is a piece of code deployed in the blockchain node. Execution of a smart contract is initiated by a message embedded in the transaction.

Structurally, a smart contract resembles a class definition in an object oriented design. It has data, functions or methods with modifiers public or private, along with getter and set of functions. **Solidity** is one such language.

```
    pragma solidity^0.4.0;

    contract SimpleStorage{
        uint storedData;
        function set(uint x){
            storedData=x;
        }
        function get() constant returns (uint x){
            return storedData;
        }
    }
```

Code Explaination:

The first line tells about the version of Solidity working in. The line uint storedData; declares a state variable called storedData of type uint (unsigned integer of 256 bits). Every node in Ethereum network should be able to execute the code irrespective of that underlying type of hardware or operating system. A smart contract written a high level programming language is translated into **EVM byte code**, and then, deployed on the **Ethereum Virtual Machine**, EVM where it is executed. Every node will host the same smart contract codes on the EVM.

5.1 Definition

Smart contracts help you exchange money, property, shares, or anything of value in a transparent, conflict-free way while avoiding the services of a middleman. The system works on the If-Then premise and is witnessed by hundreds of people, so you can expect a faultless delivery.

‘If I receive cash on delivery at this location in a developing, emerging market, then this other [product], many, many links up the supply chain, will trigger a supplier creating a new item since the existing item was just delivered in that developing market.’

5.2 Subcurrency

```
pragma solidity >=0.5.0 <0.7.0;

contract Coin {
    // The keyword "public" makes variables
    // accessible from other contracts
    address public minter;
    mapping (address => uint) public balances;

    // Events allow clients to react to specific
    // contract changes you declare
    event Sent(address from, address to, uint amount);

    // Constructor code is only run when the contract
    // is created
    constructor() public {
        minter = msg.sender;
    }

    // Sends an amount of newly created coins to an address
    // Can only be called by the contract creator
    function mint(address receiver, uint amount) public {
        require(msg.sender == minter);
        require(amount < 1e60);
        balances[receiver] += amount;
    }

    // Sends an amount of existing coins
    // from any caller to an address
    function send(address receiver, uint amount) public {
        require(amount <= balances[msg.sender], "Insufficient
            balance.");
        balances[msg.sender] -= amount;
        balances[receiver] += amount;
        emit Sent(msg.sender, receiver, amount);
    }
}
```

Code Explaination:

The line address public minter; declares a state variable of type address. The address type is a 160-bit value that does not allow any arithmetic operations.

The keyword public automatically generates a function that allows you to access the current value of the state variable from outside of the contract.

The next line, mapping (address => uint) public balances; also creates a public state variable, but it is a more complex datatype. The mapping type maps addresses to unsigned integers.

Mappings can be seen as hash tables which are virtually initialised such that every possible key exists from the start and is mapped to a value whose byte-representation is all zeros.

The line event Sent(address from, address to, uint amount); declares an "event", which is emitted in the last line of the function send.

The mint function sends an amount of newly created coins to another address. The require function call defines conditions that reverts all changes if not met. In this example, require(msg.sender == minter); ensures that only the creator of the contract can call mint, and require(amount $< 1e60$); ensures a maximum amount of tokens. This ensures that there are no overflow errors in the future.

5.2.1 Coin Viewer

```
Coin.Sent().watch({}, '', function(error, result) {
    if (!error) {
        console.log("Coin transfer: " + result.args.amount +
            " coins were sent from " + result.args.from +
            " to " + result.args.to + ".");
        console.log("Balances now:\n" +
            "Sender: " + Coin.balances.call(result.args.from) +
            "Receiver: " + Coin.balances.call(result.args.to));
    }
})
```

6 Ethereum Structure

Ethereum introducted the concept of an **account**, which are essentially *state objects* as a part of the protocol. . A transaction directly updates the account balances as opposed to maintaining the state such as in the bitcoin UTXOs. There are two types of accounts:

- Externally Owned Accounts: controlled by private keys.
- Contract Accounts: controlled by the code and can be activated only by an EOA.

Every account has a coin balance. The participant node can send transaction for Ether transfer or it can send transaction to invoke a smart contract code or both. An account must have sufficient balance to meet the fees needed for the transactions activated. Fees are paid in **Wei**. Wei is a lower denomination of **Ether**. One Ether equals 10^{18} Weis.

A transaction in Ethereum includes the recipient of the message, digital signature of the sender authorizing the transfer, amount of Wei to transfer, an optional data field or payload that contains a message to a contract, **STARTGAS** which is a value representing the maximum number of computational steps the transaction is allowed.

Gas price a value representing the fee sender is willing to pay for the computations.

6.1 Creating an account

```
//to create an account
$ geth account new
  Your new account is locked with a password. Please give a
      password. Do not forget this password.
  Passphrase:
  Repeat Passphrase:
  Address: {168bc315a2ee09042d83d7c5811b533620531f67}

 //To view accounts
 $ geth account list
    account #0: {a94f5374fce5edbc8e2a8697c15331677e6ebf0b}
    account #1: {c385233b188811c9f355d4caec14df86d6248235}
    account #2: {7f444580bfef4b9bc7e14eb7fb2a029336b07c9d}

//importing your personal wallet
geth wallet import /path/to/my/presale-wallet.json

//updating your account
geth account update b0047c606f3af7392e073ed13253f8f4710b08b6
```

7 Ethereum Operations

For a simple Ether transfer, the amount to transfer and the target address are specified along with the fees or gas points. Ethereum node is a computational system representing a business entity or an individual participant. Ethereum Full Node hosts the software needed for transaction initiation, validation, mining, block creation, and smart contract execution.

Smart contract is designed, developed, compiled and deployed on the **Ethereum Virtual Machine**, EVM that can be more than one smart contract in an EVM. Changes after execution of the code is stored in the blockchain which keeps the record of current and previous hash.

8 Incentive Model

Every action in Ethereum requires crypto fuel, or gas. **Gas points** are used to specify the fees inside of Ether, for ease of computation using standard values. Ethereum has specified gas points for each type of operation. Every operation that can be performed by a transaction or contract on the Ethereum platform costs a certain number of gas, with operations that require more computational resources costing more gas than operations that require few computational resources. Mining process computes gas points required for execution of a transaction.

- **Gas limit** is the amount of gas points available for a block to spend.
- **Gas spent** is the actual amount of gas spent at the completion of the block creation.

8.1 Mining Incentive Model

Ethereum has specified gas points for each type of operation. Mining process computes gas points required for execution of a transaction. The proof of work puzzle winner, miner that creates a new block, is incentivized with the base fees of three Ethers, and the transaction fees in Ethereum blockchain. The winning miner also gets the fees, gas points for execution of a smart contract transactions.

The blocks created by them are called **Ommer Blocks**. These are added as Ommer Blocks, or side blocks, to the main chain. Ommer miners also get a small percentage of the total gas points as a consolation and for network security.

8.2 Proof of Work vs Proof of Stake

8.2.1 Proof of Work

Proof of work is a protocol that has the main goal of deterring cyber-attacks such as a distributed denial-of-service attack (DDoS) which has the purpose of exhausting the resources of a computer system by sending multiple fake requests.

Going deeper, proof of work is a requirement to define an expensive computer calculation, also called **mining**, that needs to be performed in order to create a new group of trustless transactions (the so-called block) on a distributed ledger called blockchain.

From a technical point of view, the mining process is an **operation of inverse hashing**: it determines a number (nonce), so the cryptographic hash algorithm of block data results in less than a given threshold or **difficulty**; is what determines the competitive nature of mining: more computing power is added to the network, the higher this parameter increases, increasing also the average number of calculations needed to create a new block.

8.2.2 Proof of Stake

Proof of stake will make the consensus mechanism completely virtual. While the overall process remains the same as proof of work (POW), the method of reaching the end goal is entirely different.

In POS, instead of miners, there are **validators**. The validators lock up some of their **Ether as a stake** in the ecosystem. Following that, the validators **bet** on the blocks that they feel will be added next to the chain. When the block gets added, the validators get a block reward in proportion to their stake.

8.2.3 Which is better?

Proof of stake is better to use because:

- Bitcoin transactions use up too much energy.
- A safer network as attacks become more expensive: if a hacker would like to buy 51% of the total number of coins, the market reacts by the fast price appreciation.

9 Public-Key Cryptography

Let's consider simple symmetric key encryption, in which the same key is used for encryption and decryption, so it is called **symmetric key**. Example, **Ceasar encryption** is the simplest one with alphabets of a message are shifted by a fixed number, and this number is called the **Key**. But there are two issues with this type of encryption:

- It is easy to derive the secret key from the encrypted data.
- Passing of the key to the participant transacting.

Public-key cryptography addresses these issues by employing instead of a single secret key, two different keys, a public and a private key. The public-key private key pair has the unique quality that even though a data is encrypted with the private key, it can be decrypted with the corresponding public-key and vice versa.

When someone wants to send an encrypted message, they can pull the intended recipient's public key from a public directory and use it to encrypt the message before sending it. The recipient of the message can then decrypt the message using their related private key. On the other hand, if the sender encrypts the message using their private key, then the message can be decrypted only using that sender's public key, thus authenticating the sender.

A popular implementation of public key, private key is the **Rivest Shamir Adleman (RSA)** algorithm. Common application of RSA is the passwordless user authentication, for example for accessing a virtual machine on Amazon cloud.

RSA derives its security from the computational difficulty of factoring large integers that are the product of two large prime numbers. Multiplying two large primes is easy, but the difficulty of determining the original numbers from the product – factoring – forms the basis of public key cryptography security.

Elliptic Curve Cryptography, ECC family of algorithms is used in the bitcoin as well as an Ethereum block chain for generating the key pair. ECC is stronger than RSA for a given number of bits. 256 bit ECC key pair is equal in strength to about 3072 bits of RSA key pair. And hence both bitcoin and Ethereum use ECC based algorithms for their encryption needs.

9.1 Uses of Asymmetric Cryptography

Asymmetric cryptography is typically used to authenticate data using digital signatures. A digital signature is a mathematical technique used to validate the authenticity and integrity of a message, software or digital document. It is the digital equivalent of a handwritten signature or stamped seal.

Benefits of asymmetric cryptography

- the key distribution problem is eliminated because there's no need for exchanging keys.
- security is increased as the private keys don't ever have to be transmitted or revealed to anyone.
- the use of digital signatures is enabled so that a recipient can verify that a message comes from a particular sender.

- it allows for non-repudiation so the sender can't deny sending a message.

10 Hashing

Hashing plays a critical role in the blockchain process, and also in the integrity of the transaction and confidentiality of data. **A hash function or hashing transforms and maps an arbitrary length of input data value to a unique fixed length value.** Input data can be a document, tree data, or a block data. Even a slight difference in the input data would produce a totally different hash output value. The following are two basic requirements of a hash function. The algorithm chosen for the hash function should be:

- **a one-way function**. It is to make certain that no one can derive the original items hashed from the hash value. What pre-image resistance states are that given H(A) it is infeasible to determine A, where A is the input and H(A) is the output hash.
- **it should be collision free, or exhibit extremely low probability of collision**. Is to make sure that the hash value uniquely represents the original items hashed. There should be extremely low probability that two different data sets map onto the same hash value.
- **Quick Computation:** The hash function should be capable of returning the hash of input quickly. If the process isn't fast enough then the system simply won't be efficient.
- **Small Changes In The Input Changes the Hash**

These requirements are achieved by choosing a strong algorithm such as **secure hash**, and by using appropriately large number of bits in the hash value. Most common hash size now is 256 bits and the common functions are SHA-3, SHA-256 and Keccak.

10.1 The Birthday Paradox

If you meet any random stranger out on the streets the chances are very low for both of you to have the same birthday. In fact, assuming that all days of the year have the same likelihood of having a birthday, the chances of another person sharing your birthday is 1/365 which is 0.27%. In other words, it is really low.

However, having said that, if you gather up 20-30 people in one room, the odds of two people sharing the exact same birthday rises up astronomically. In fact, there is a 50-50 chance for 2 people sharing the same birthday in this scenario!

Why does that happen? It is because of a simple rule in probability which goes as follows. Suppose you have N different possibilities of an event happening, then you need square root of N random items for them to have a 50% chance of a collision.

So applying this theory for birthdays, you have 365 different possibilities of birthdays, so you just need Sqrt(365), which is 23 , randomly chosen people for 50% chance of two people sharing birthdays.

10.1.1 What is the application of this in hashing?

Suppose you have a 128-bit hash which has 2^{128} different possibilities. By using the birthday paradox, you have a 50% chance to break the collision resistance at the sqrt(2^{128}) = 2^{64}th instance.

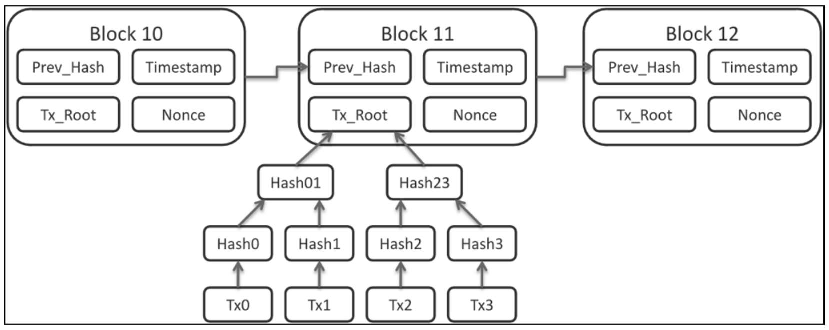
As you can see, it is much easier to break collision resistance than it is to break preimage resistance. No hash function is collision free, but it usually takes so long to find a collision. So, if you are using a function like SHA-256, it is safe to assume that if H(A) = H(B) then A = B.

10.2 Data Structures

There are two data structure properties that are critical if you want to understand how a blockchain works. They are:

- Pointers
- Linked List

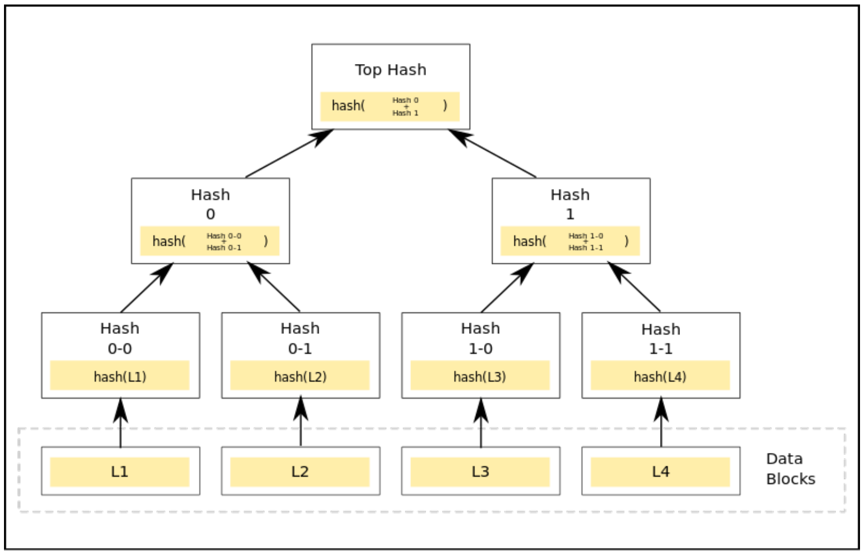
A hash pointer is similar to a pointer, but instead of just containing the address of the previous block it also contains the hash of the data inside the previous block. The picture shows a block header:



10.3 Types of Hashing

We use two types of hashing: simple hashing and Merkle tree hashing with ADD as a hash function. In the **simple hash approach**, all the data items are linearly arranged and hashed.

In a **Merkle tree-structured approach**, the data is at the leaf nodes of the tree, leaves are pairwise hash to arrive at the same hash value as a simple hash. A typical Merkle Tree looks like:



When we have a fixed number of items to be hashed, such as the items in a block header, and we are verifying the composite block integrity and not the individual item integrity, we use simple hash. When the number of items differ from block to block, for example, number of transactions, number of states, number of receipts, we use the tree structure for computing the hash. Tree structure helps the efficiency of repeated operations, such as transaction modification and the state changes from one block to the next. **Log N versus N**.

11 Transaction Integrity

To manage the integrity of a transaction we need:

- A secure unique account address. We need a standard approach to uniquely identify the participants in the decentralized network.
- Authorization of the transaction by the sender through digital signing.
- Verification that the content of that transaction is not modified.

Addresses of accounts are generated using public key, private key pair.

1. **Step 1**, at 256-bit random number is generated, and designated as the private key. Kept secure and locked using a passphrase.
2. **Step 2**, an ECC algorithm is applied to the private key, to get a unique public key. This is the **private public key pair**.
3. **Step 3**, then a hashing function is applied to the public key to obtain account address. The address is shorter in size, only 20 bytes or 160 bits.

A transaction for transferring assets will have to be authorized, it has to be non-repudiable, and unmodifiable.

1. Find the hash of the data fields of the transaction.
2. Encrypt that hash using the private key of the participant originating the transaction. Thus, digitally signing the transaction to authorize and making the transaction non-repudiable.
3. This hash just added to the transaction. It can be verified by others decryiptng it using the public key of the sender of the transaction, and recomputing the hash of the transaction.

Then, compare the computed hash, and the hash received at the digital signature. If that is a match, accept the transaction. Otherwise, reject it. **Note** that for the complete transaction verification, the timestamp, nons, account balances, and sufficiency of fees are also verified.

12 Securing Blockchain

Some of the main components of the Ethereum block are the header, the transactions, including the transaction hash or the transaction root, and the state radio booth, the state hash, or the state root.

In Ethereum, the **block hash** is the block of all the elements in the block header, including the transaction root and state root hashes. It is computed by applying a variant of **SHA-3 algorithm** called **Keccak** and all the items of the block header. The block header hash is calculated by running the block header through the SHA256 algorithm **twice**.

Hashes of transaction in a block are processed in a tree structure called Mekle tree hash. Merkle tree hash is also used for computing the state root hash, since only the hash of the chained states from block to block have to be re-computed. It is also used for receipt hash root.

Smart contract execution in Ethereum results in **state transitions**. Every state change requires state root hash re-computation. Instead of computing hash for the entire set of states, only the affected path in the Merkle tree needs to be re-computed. Block hash in Ethereum is computed by first computing the state root hash, **transaction root hash** and then receipt root hash, shown at the bottom of the block header. These roots and all the other items in the header are hash together with the variable nodes to solve the proof of work puzzle. **Purpose of block hash:**

1. verification of the integrity of the block and the transactions,
2. formation of the chain link by embedding the previous block hash in the current block header.

If any participant node tampers with the block, it's hash value changes resulting in the mismatch of the hash values and rendering the local chain of the node in an invalid state.

13 The Trust trail

The trust in a decentralized blockchain is also about securing, validating, verifying, and making sure resources needed for transaction execution are available.

The Trust Trail is defined by these operations: validate transaction, verify gas and resources, gather transactions, execute transaction to get a new state, form the block, work towards consensus, finalize the block by the bidder, and everyone add the block to their chain and confirm the transactions.

- **Execute transactions:**

  Merkle tree hash of the validated transactions is computed in Ethereum. This is the transaction root of the block header. All miners execute the transaction for either transfer, as well as for execution of smart contracts. The state resulting from transaction execution are used in computing the Merkle tree hash of the states, the state root of the block header. The receipt root of the block header is also computed.

- **Consensus Protocol:**

  A secure chain is a single main chain with a consistent state. Every valid block added to this chain, adds to the trust level of the chain. A protocol to choose the next block and add it to the blockchain is called as Proof of Work.

  - **Proof of Work** uses hashing. First, compute the hash of the block header elements that is a fixed value, and a nonce that is a variable. If hash value is less than 2^{128} for bitcoin, and less than function of difficulty for ethereum, the puzzle has been solved. If it has not been solved, repeat the process after changing the nonce value. If the puzzle has been solved, broadcast the winning block that will be verified by other miners. Non-winning miner nodes add the new block to the local copy of the chain, and move on to working on the next block. Common criticisms include that it requires enormous amounts of computational energy, that it does not scale well (transaction confirmation takes about 10-60 minutes) and that the majority of mining is centralized in areas of the world where electricity is cheap.
  - The most common alternative to proof of work is **proof of stake.** n this type of consensus algorithm, instead of investing in expensive computer equipment in a race to mine blocks, a 'validator' invests in the coins of the system. All the coins exist from day one, and validators (also called stakeholders, because they hold a stake in the system) are paid strictly in transaction fees. In proof of stake, your chance of being picked to create the next block depends on the fraction of coins in the system you own (or set aside for staking). A validator with 300 coins will be three times as likely to be chosen as someone with 100 coins.

- **Robustness** is the ability to satisfactorily manage exceptional situations. We start with the securechain indicated by three blocks and we want to add to this chain. Here,
  - **if two miners have solved the consensus puzzle very close to each other**. Bitcoin protocol allows this chain split or two chains for the next cycle. One led by each of the competing blocks. The probability that the next block will happen at the same time in both these chains is extremely low. So the winner of the next cycle for block creation consolidates one of the chains and that chain becomes the accepted chain. In this case, the newest block is added to the main chain. The transaction in the other blocks are returned to the unconfirmed pool.

    Etherium handles more than one person we know by allowing Omar or Runner-Up blocks and allocating a small incentive for these Runner-Up blocks. This incentive model helps in keeping the chains secure. New blocks are added only to the mainchain and not to the Runner-Up chains. That are Runner-up blocks are maintained for six more blocks after they were added.

  - **There's a possibility that digital currency and other consumables are single used digital assets, can be intentionally or inadvertently reused in transactions.** A policy for handling transaction and double spending in Bitcoin is to allow the first transaction that reference the digital asset and reject the rest of the transaction that reference the same digital asset. In Ethereum, a combination of account number and a global nonce is used to address the doublet spending issue. Every time a transaction is initiated by an account, a global nonce is included in the transaction. After that, the nonce is incremented. Time stamp on the nonce in the transaction should be unique and verified to prevent any double use of digital assets.

- **Forks:** Forks are mechanisms that add to the robustness of the Blockchain framework. **A Soft Fork** is a fork where updated versions of the protocol are backwards compatible with previous versions. **A Hard Fork** is a change of the protocol that is not backwards compatible with older versions of the client. Participants would absolutely need to upgrade their software in order to recognize new blocks.
  Soft Fork and Hard Fork in the blockchain word are like the release of software patches, and new versions of operating systems respectively. Forks are mechanisms that add to the robustness of the blockchain framework. Well-managed forks help build credibility in the blockchain by providing approaches to manage unexpected faults and planned improvements.