“The SandBox”(The gaming asset ownership-BLOCK-CHAIN)

Here's a breakdown of The Sandbox's blockchain implementation for in-game asset ownership:

The Sandbox and Blockchain: A Powerful Combination

The Sandbox is a virtual world built on the Ethereum blockchain, allowing players to experience a new level of ownership and control over their in-game assets. Here's how it works:

- **Tokens Rule the Game:** The Sandbox uses two main tokens:
    - **SAND:** This is the platform's fungible token (ERC-20) used for things like buying LAND (virtual plots) and participating in governance.
    - **ASSETS:** These are ERC-721 non-fungible tokens (NFTs) that represent unique in-game items like avatars, wearables, and game creations. Each NFT is one-of-a-kind and verifiable on the blockchain.
- **LAND Ownership:** Imagine buying a virtual plot of land. In The Sandbox, you can do this using SAND tokens. Each LAND is an NFT, granting you ownership of that digital space. You can build experiences, games, or even rent it out to others.

- **Creating and Owning Assets:** The Sandbox empowers players to create their own game elements. Using VoxEdit, a built-in tool, players can design voxel objects (3D block-based creations) and convert them into ASSETS (NFTs). These ASSETS can be anything from clothing for your avatar to entire game environments. Owning an ASSET means you have complete control over it. You can use it in your LAND, sell it on the Marketplace to other players for SAND, or even collect royalties if others use your creations in their games.

- **The Marketplace:** This is where the magic of buying, selling, and trading ASSETS happens. Players can use SAND to

purchase ASSETS created by other players, adding unique items to their collection or using them in their LAND.

Benefits of Blockchain in The Sandbox:

- **True Ownership:** Blockchain ensures verifiable ownership of LAND and ASSETS. Unlike traditional games where in-game items are owned by the company, in The Sandbox, you truly own your creations and virtual land.

- **Monetization:** Players can create and sell their ASSETS, generating revenue within the game. This fosters a creator economy where skilled players can build a business within The Sandbox.

- **Secure Trading:** The Ethereum blockchain provides a secure platform for trading ASSETS, eliminating the risk of fraud or item duplication.

- **Open & Decentralized:** The Sandbox aspires to be a decentralized platform where players have a say in its future. SAND token holders can participate in governance decisions through Decentralized Autonomous Organization (DAO) voting.

Putting it All Together: How Blocks, Ethereum, and Tokens Work in Harmony

1. **Create with VoxEdit:** You design your masterpiece voxel creation using VoxEdit.
2. **Become an Owner:** Once happy with your creation, you can convert it into an ASSET using SAND tokens. This "mints" your creation onto the Ethereum blockchain, making you the verified owner.
3. **Own Your LAND:** Use SAND to purchase LAND, a virtual plot where you can build experiences, games, or even rent it out to others. Each LAND is also an NFT, securing your ownership on the blockchain.

4. **Trade on the Marketplace:** The Sandbox has a marketplace where you can use SAND to buy ASSETS created by other players, adding unique items to your collection or using them in your LAND.
5. **Monetize Your Skills:** Since you own your ASSETS, you can sell them on the marketplace for SAND, generating revenue within the game.

Benefits of Blockchain in The Sandbox:

- **True Ownership:** Unlike traditional games, you truly own your creations and LAND thanks to the power of blockchain.
- **Secure Trading:** The Ethereum blockchain ensures secure trading of ASSETS, eliminating the risk of fraud or item duplication.
- **Earning Potential:** The ability to sell your creations opens doors for players to create and earn within The Sandbox.

In conclusion, The Sandbox utilizes blockchain technology to create a secure and empowering environment for players. By leveraging blocks on the Ethereum blockchain and tokens like SAND and ASSETS, The Sandbox fosters a world of creativity, ownership, and exciting possibilities.

Overall, The Sandbox's blockchain implementation creates a unique gaming experience where players have more control, ownership, and opportunity to create, play, and earn.

* blockchain is a distributed system at its core.

It is a distributed ledger which can be centralized or decentralized. A blockchain is originally intended to be and is usually used as a decentralized platform

Distributed systems are a computing paradigm two or more nodes work with each other in coordinated way to achieve a common outcome. It is modeled in such a way that end users see it as a single logical platform

Eg: google search engine

NOde:

*A **node** can be defined as an individual player in a distributed system.

*All nodes are capable of sending and receiving messages to and from each other.

*Nodes can be honest, faulty, or malicious, and they have memory and a processor.

*A node that exhibits irrational behavior is also known as a **Byzantine node** after the Byzantine Generals Problem.

The Byzantine Generals problem

*In 1982, a thought experiment was proposed by Lamport and others in their research paper, The Byzantine Generals Problem which is available at: https://www.microsoft.com/en-us/research/publication/byzantine-generals-problem/

* whereby a group of army generals who lead different parts of the Byzantine army are planning to attack or retreat from a city.

* The only way of communicating among them is via a messenger.

* They need to agree to strike at the same time in order to win.

* The issue is that one or more generals might be traitors who could send a misleading message.

* Therefore, there is a need for a viable mechanism that allows for agreement among the generals, even in the presence of the treacherous ones,

* so that the attack can still take place at the same time. As an analogy to distributed systems,

* the generals can be considered nodes, the traitors as Byzantine (malicious) nodes, and the messenger can be thought of as a channel of communication among the generals.

This problem was solved in 1999 by Castro and Liskov who presented the **Practical Byzantine Fault Tolerance** (**PBFT**) algorithm, where consensus is reached after a certain number of messages are received containing the same signed content.

This type of inconsistent behavior of Byzantine nodes can be intentionally malicious, which is detrimental to the operation of the network. Any unexpected behavior by a node on the network, whether malicious or not, can be categorized as Byzantine.

(ppp sss tt) – BC **TYPES Are proved so talented**

Types of Blockchain:

1) Public Blockchains: open to all, anyone can be a node in decision making

public – not validators – So no rewards (Cryptocurrency) during verifications / minning the blocks

not owned for by any one.

Consensus mechanism – to get one truth value

permission less blockchains.

2) private Blockchains:

group of individuals

only open to specified persons have permissions

- permission blockchains.
-

3) Hybrid Blockchains / Semi private Blockchains:

→ combinded above B.C

- Some ~~time~~ Part it is private and other ~~times~~ Part is public
- private part is controlled by specific persons and public part is available to the all.

4) Side chains:

- pegged Sidechains
- coins can be moved from one blockchain to another and more back.
- 2 types of side chain - (1) one-way pegged sidechain
  (2) two-way pegged sidechain

5) Permissioned ledger:

participants of the network are known and already trusted.

- don't use distributed consensus
- They use agreement protocol.
- It can be public BC but with record of the access control.

6) Distributed ledger:

This is distributed among its participants and spread across multiple sites or organizations.

7) Shared ledger:

Used to describe or application or database that is shared by the public or a consortium.

8) Fully private and proprietary blockchains:

Ex: Sharing data b/w various government departments.

no mainstream application.

9) Tokenized Blockchains:

Standard blockchains that generate cryptocurrency as a result of a consensuses or minning or intial distribution.

10) Tokenless blockchains:

→ Not ~~take~~ transfer of money/transaction takes place But only data is transfered b/w nodes.

Features of Blockchain

1) Distribution Consensus
2) Transaction verification
3) platform for smart Contract
4) Transfering values b/w peers
5) Generating cryptocurrency
6) providing Security
7) Smart property
8) Uniqueness
9) immutability
10) Smart Contracts.

Consensus means agreement and achieved through 2 ways:
(i) Election
(ii) Distribution
using computational resources
→ computational ability - proof of work

Distributed consensus → safety & liveness together. Can't be provided

paxos - 1st Blockchain

puzzle is easy to solve verify
easy to difficult to solve

Application of Blockchain
- Healthcare System
- Supply chain management
- Agriculture
- Design of website

2 types

1) leader-based
2) Byzantine Fault tolerance-based.

Proof of work: Type of consensus mechanism.

relies on proof that enough computational resource have been spent before proposing a value for acceptance by the network.

→ used in Bitcoin. → It is energy instensive.

Proof of stake:

node or user has enough stake in the system.

→ introduced by peercoin. used in Ethereum Blockchain

Stake is value/money we bet on a certain outcome.

coin-age-based selection:

The older the node becomes, the higher the chances of it becoming the new validator.

Random Block Selection:

validator chosen with combination of lowest hash value and higher stake.

Validators - minters
- forgers

51% attacks:

advantages
1) Security
2) Decentralization.
3 Energy effeciency.

Weakness:
1) New technology
2)

Blockchain using POS
→ Ethereum
→ peercoin
→ Nxt

Variants
→ Regular
→ Delegated
→ leased.

Delegated Blockchain:
→ used in bitshares blockchain.
→ voting by nodes is done here.

Proof of Elapsed Time:

uses Introduced by the intel it uses Trusted Execution Environment. Context of Intel sawtooth lake blockchain Project.

Deposit based Consensus:

Nodes that ~~tshi~~ wish to participate on the network have to put in a security deposit before they can propose a block.

Proof of importance:

It monitors the usage and movement of tokens by the user to establish a level of trust and importance.

Federated consensus or Federated Byzantine Consensus

nodes in the protocol keep a group of publicly trusted peers and propagated only those transactions that have been validated by the majority of trusted nodes.

Reputation-based mechanisms.

→ voting from other members.

→ basis of the reputation it has built on network overtime.

Benefits and limitations of Blockchain:

1) Decentralization
2) Transparency & Trust
3) Immutability
4) High availability
5) Highly Secure.
6) Cost saving
7) Simplification of pardigms
8) Faster dealings.

1) Scalability
2) Adaptability
3) Regulation
4) privacy
5) Relatively immature technology.

***Technical definition**: Blockchain is a peer-to-peer, distributed ledger that is cryptographically-secure, append-only, immutable (extremely hard to change), and updateable only via consensus or agreement among peers.*

Peer-to-peer

The first keyword in the technical definition is *peer-to-peer*. This means that there is no central controller in the network, and all participants talk to each other directly. This property allows for cash transactions to be exchanged directly among the peers without a third-party involvement, such as by a bank.

Distributed ledger

Dissecting the technical definition further reveals that blockchain is a *distributed ledger*, which simply means that a ledger is spread across the network among all peers in the network, and each peer holds a copy of the complete ledger.

Cryptographically-secure

Next, we see that this ledger is *cryptographically-secure*, which means that cryptography has been used to provide security services which make this ledger secure against tampering and misuse. These services include non-repudiation, data integrity, and data origin authentication. You will see how this is achieved later in Chapter 3, *Symmetric Cryptography* which introduces the fascinating world of cryptography.

Append-only

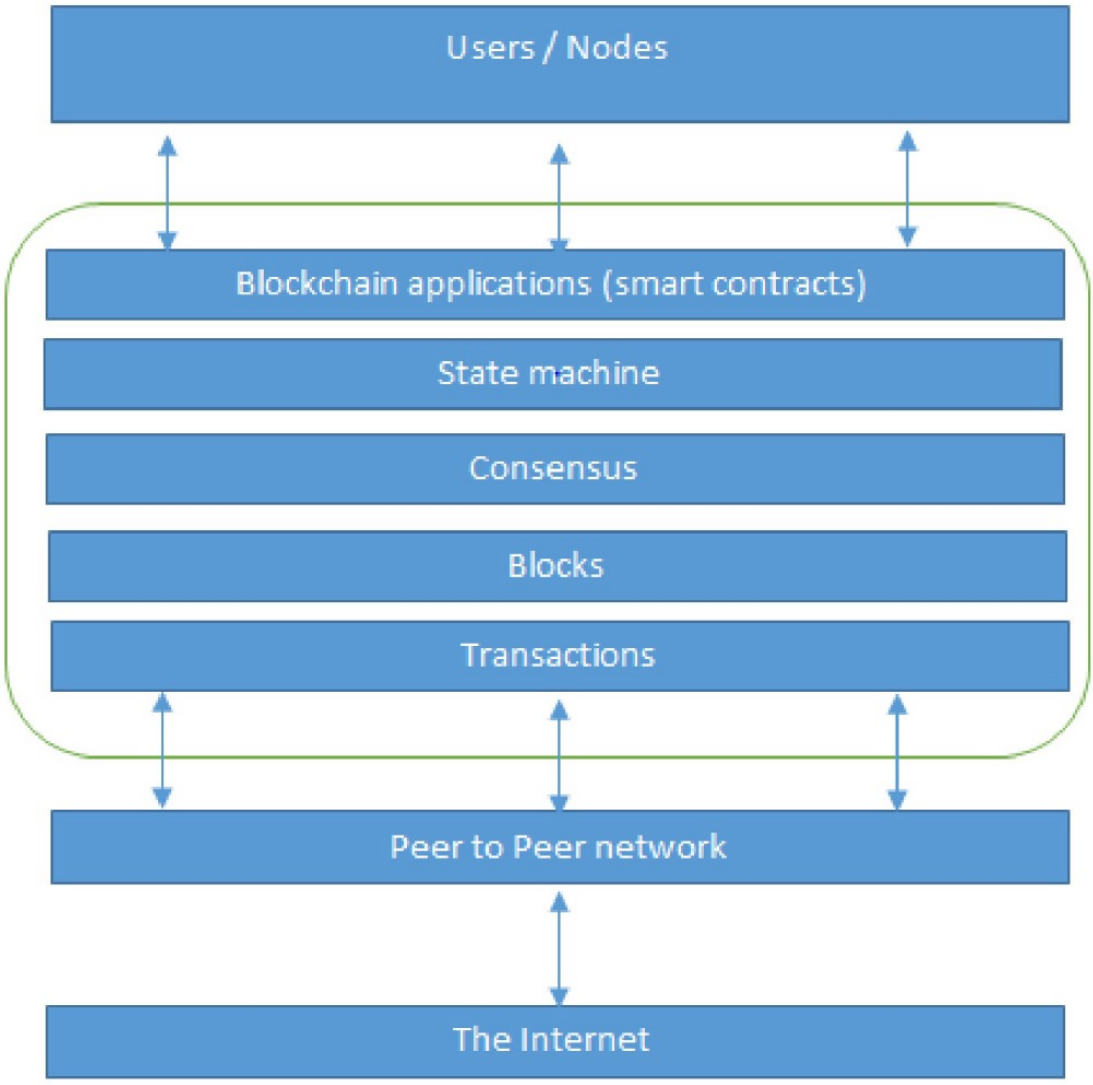
Another property that we encounter is that blockchain is *append-only*, which means that data can only be added to the blockchain in *time-ordered sequential order*. This property implies that once data is added to the blockchain, it is almost impossible to change that data and can be considered practically immutable. Nonetheless, it can be changed in rare scenarios wherein collusion against the blockchain network succeeds in gaining more than 51 percent of the power. There may be some legitimate reasons to change data in the blockchain once it has been added, such as the *right to be forgotten* or *right to erasure* (also defined in **General Data Protection** (**GDPR**) ruling, https://gdpr-info.eu/art-17-gdpr/).

However, those are individual cases that need to be handled separately and that require an elegant technical solution. For all practical purposes, blockchain is indeed immutable and cannot be changed.

Updateable via consensus

Finally, the most critical attribute of a blockchain is that it is *updateable* only via consensus. This is what gives it the power of decentralization. In this scenario, no central authority is in control of updating the ledger. Instead, any update made to the blockchain is validated against strict criteria defined by the blockchain protocol and added to the blockchain only after a consensus has been reached among all participating peers/nodes on the network. To achieve consensus, there are various consensus facilitation algorithms which ensure that all parties are in agreement about the final state of the data on the blockchain network and resolutely agree upon it to be true. Consensus algorithms are discussed later in this chapter and throughout the book as appropriate.

Blockchain can be thought of as a layer of a distributed peer-to-peer network running on top of the internet, as can be seen in the following diagram. It is analogous to SMTP, HTTP, or FTP running on top of TCP/IP.



The network view of a blockchain

* a **blockchain can be defined as a platform** where peers can exchange value / electronic cash using transactions without the need for a centrally-trusted arbitrator. For example, for cash transfers, banks act as a trusted third party

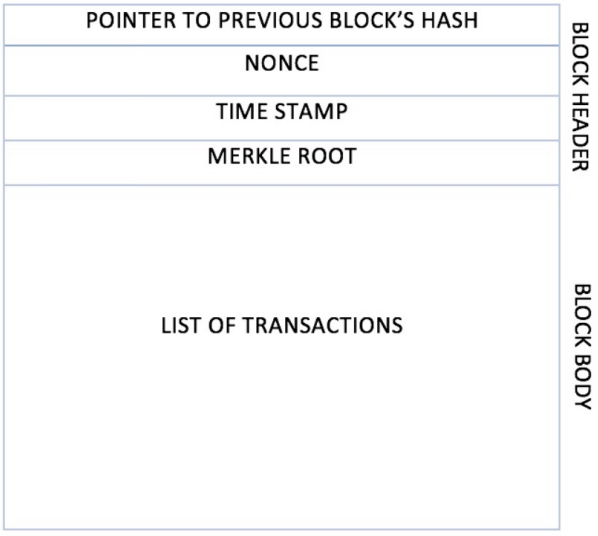
*This disintermediation allows blockchain to be a decentralized consensus mechanism where no single authority is in charge of the database

1. A **block** is merely a selection of transactions bundled together and organized logically

2. A **transaction** is a record of an event, for example, the event of transferring cash from a sender's account to a beneficiary's account. A block is made up of transactions, and its size varies depending on the type and design of the blockchain in use.

3. A **genesis block** is the first block in the blockchain that is hardcoded at the time the blockchain was first started. The structure of a block is also dependent on the type and design of a blockchain.

4. few attributes that are essential to the functionality of a block: the block header, which is composed of pointer to previous block, the timestamp, nonce, Merkle root, and the block body that contains transactions



The generic structure of a block.

A **nonce** is a number that is generated and used only once. A nonce is used extensively in many cryptographic operations to provide replay protection, authentication, and encryption. In blockchain, it's used in PoW consensus algorithms and for transaction replay protection.

Merkle root is a hash of all of the nodes of a Merkle tree. Merkle trees are widely used to validate the large data structures securely and efficiently. In the blockchain world, Merkle trees are commonly used to allow efficient verification of transactions. Merkle root in a

blockchain is present in the block header section of a block, which is the hash of all transactions in a block. This means that verifying only the Merkle root is required to verify all transactions present in the Merkle tree instead of verifying all transactions one by one. We will elaborate further on these concepts in Chapter 4, *Public Key Cryptography*.

Elements of a generic blockchain are described here one by one. These are the elements that you will come across in relation to blockchain:

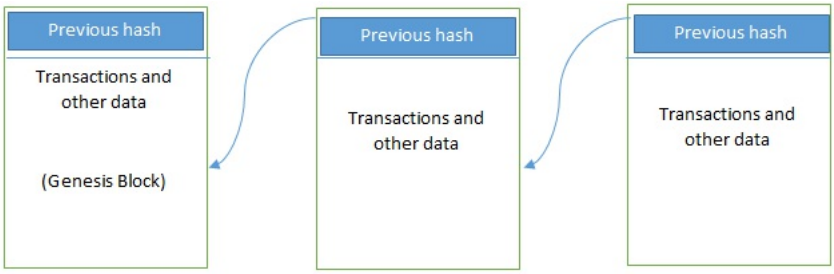
Address: Addresses are unique identifiers used in a blockchain transaction to denote senders and recipients. An address is usually a public key or derived from a public key. While addresses can be reused by the same user, addresses themselves are unique. In practice, however, a single user may not use the same address again and generate a new one for each transaction. This newly-created address will be unique. Bitcoin is, in fact, a pseudonymous system. End users are usually not directly identifiable, but some research in removing the anonymity of Bitcoin users has shown that they can be identified successfully. A good practice is for users to generate a new address for each transaction in order to avoid linking transactions to the common owner, thus preventing identification.

Transaction: A transaction is the fundamental unit of a blockchain. A transaction represents a transfer of value from one address to another.
Block: A block is composed of multiple transactions and other elements, such as the previous block hash (hash pointer), timestamp, and nonce.

Peer-to-peer network: As the name implies, a peer-to-peer network is a network topology wherein all peers can communicate with each other and send and receive messages.
Scripting or programming language: Scripts or programs perform various operations on a transaction in order to facilitate various functions. For example, in Bitcoin, transaction scripts are predefined in a language called **Script**, which consist of sets of commands that allow nodes to transfer tokens from one address to another. Script is a limited language, however, in the sense that it only allows essential operations that are necessary for executing transactions, but it does not allow for arbitrary program development. Think of it as a calculator that only supports standard preprogrammed arithmetic operations. As such, Bitcoin script language cannot be called *Turing complete*. In simple words, Turing complete language means that it can perform any computation. It is named after Alan Turing who developed the idea of Turing machine that can run any algorithm however complex. Turing complete languages need loops and branching capability to perform complex computations. Therefore, Bitcoin's scripting language is not Turing complete, whereas Ethereum's Solidity language is.

To facilitate arbitrary program development on a blockchain, Turing complete programming language is needed, and it is now a very desirable feature of blockchains. Think of this as a computer that allows development of any program using programming languages. Nevertheless, the security of such languages is a crucial question and an essential and ongoing research area. We will discuss this in greater detail in C hapter 5, *Introducing Bitcoin*, Chapter 9, *Smart Contracts*, and Chapter 13, *Development Tools and Frameworks*, later in this book.

Virtual machine: This is an extension of the transaction script introduced earlier. A *virtual machine* allows 51



Turing complete code to be run on a blockchain (as smart contracts); whereas a transaction script is limited in its operation. However, virtual machines are not available on all blockchains. Various blockchains use virtual machines to run programs such as **Ethereum Virtual Machine** (**EVM**) and **Chain Virtual Machine** (**CVM**). EVM is used in Ethereum blockchain, while CVM is a virtual machine developed for and used in an enterprise-grade blockchain called **Chain Core**.

State machine: A blockchain can be viewed as a state transition mechanism whereby a state is modified from its initial form to the next one and eventually to a final form by nodes on the blockchain network as a result of a transaction execution, validation, and finalization process.
Node: A node in a blockchain network performs various functions depending on the role that it takes on. A node can propose and validate transactions and perform mining to facilitate consensus and secure the blockchain. This goal is achieved by following a **consensus protocol** (most commonly PoW). Nodes can also perform other functions such as simple payment verification (lightweight nodes), validation, and many other functions depending on the type of the blockchain used and the role assigned to the node. Nodes also perform a transaction signing function. Transactions are first created by nodes and then also digitally signed by nodes using private keys as proof that they are the legitimate owner of the asset that they wish to transfer to someone else on the blockchain network. This asset is usually a token or virtual currency, such as Bitcoin, but it can also be any real-world asset represented on the blockchain by using tokens.

Smart contract: These programs run on top of the blockchain and encapsulate the business logic to be executed when certain conditions are met. These programs are enforceable and automatically executable. The smart contract feature is not available on all blockchain platforms, but it is now becoming a very desirable feature due to the flexibility and power that it provides to the blockchain applications. Smart contracts have

many use cases, including but not limited to identity management, capital markets, trade finance, record management, insurance, and e-governance. Smart contracts will be discussed in more detail in Chapter 9, *Smart Contracts*.

How blockchain accumulates blocks

Now we will look at a general scheme for creating blocks. This scheme is presented here to give you a general idea of how blocks are generated and what the relationship is between transactions and blocks:

1. A node starts a transaction by first creating and then digitally signing it with its private key. A transaction can represent various actions in a blockchain. Most commonly this is a data structure that represents transfer of value between users on the blockchain network. Transaction data structure usually consists of some logic of transfer of value, relevant rules, source and destination addresses, and other validation information. This will be covered in more detail in specific chapters on Bitcoin and Ethereum later in the book.

2. A transaction is propagated (flooded) by using a flooding protocol, called Gossip protocol, to peers that validate the transaction based on preset criteria. Usually, more than one node are required to verify the transaction.

3. Once the transaction is validated, it is included in a block, which is then propagated onto the network. At this point, the transaction is considered confirmed.

4. The newly-created block now becomes part of the ledger, and the next block links itself cryptographically back to this block. This link is a hash pointer. At this stage, the transaction gets its second confirmation and the block gets its first confirmation.

5. Transactions are then reconfirmed every time a new block is created. Usually, six confirmations in the Bitcoin network are required to consider the transaction final.

It is worth noting that steps 4 and 5 are considered non-compulsory, as the transaction itself is finalized in step 3; however, block confirmation and further transaction reconfirmations, if required, are then carried out in step 4 and step 5.

This completes the basic introduction to blockchain. In the next section, you will learn about the benefits and limitations of this technology.

Benefits and limitations of blockchain

Numerous advantages of blockchain technology have been discussed in many industries and proposed by thought leaders around the world who are participating in the blockchain space. The notable benefits of blockchain technology are as follows:

Decentralization: This is a core concept and benefit of the blockchain. There is no need for a trusted third party or intermediary to validate transactions; instead, a consensus mechanism is used to agree on the validity of transactions.
Transparency and trust: Because blockchains are shared and everyone can see what is on the blockchain, this allows the system to be transparent. As a result, trust is established. This is more relevant in scenarios such as the disbursement of funds or benefits where personal discretion in relation to selecting beneficiaries needs to be restricted.

Immutability: Once the data has been written to the blockchain, it is extremely difficult to change it back. It is not genuinely immutable, but because changing data is so challenging and nearly impossible, this is seen as a benefit to maintaining an immutable ledger of transactions.
High availability: As the system is based on thousands of nodes in a peer-to-peer network, and the data is replicated and updated on every node, the system becomes highly available. Even if some nodes leave the network or become inaccessible, the network as a whole continues to work, thus making it highly available. This redundancy results in high availability.

Highly secure: All transactions on a blockchain are cryptographically secured and thus provide network integrity.
Simplification of current paradigms: The current blockchain model in many industries, such as finance or health, is somewhat disorganized. In this model, multiple entities maintain their own databases and data sharing can become very difficult due to the disparate nature of the systems. However, as a blockchain can serve as a single shared ledger among many interested parties, this can result in simplifying the model by reducing the complexity of managing the separate systems maintained by each entity.

Faster dealings: In the financial industry, especially in post-trade settlement functions, blockchain can play a vital role by enabling the quick settlement of trades. Blockchain does not require a lengthy process of verification, reconciliation, and clearance because a single version of agreed-upon data is already available on a shared ledger between financial organizations.

Cost saving: As no trusted third party or clearing house is required in the blockchain model, this can massively eliminate overhead costs in the form of the fees which are paid to such parties.

As with any technology, some challenges need to be addressed in order to make a system more robust, useful, and accessible. Blockchain technology is no exception. In fact, much effort is being made in both academia and industry to overcome the challenges posed by blockchain technology. The most sensitive blockchain problems are as follows:

Scalability
Adaptability
Regulation
Relatively immature technology Privacy

All of these issues and possible solutions will be discussed in detail in Chapter 18, *Scalability and Other Challenges*.

Features of a blockchain(pigs puttd)

A blockchain performs various functions which are supported by various features. These functions include but are not limited to transfer of value, managing assets and agreements. All of the blockchain tiers described in the previous section perform these functions with the help of features offered by blockchain, but with some exceptions. For example, smart contracts are not supported by all blockchain platforms, such as Bitcoin. Another example is that not all blockchain platforms produce cryptocurrency or tokens, such as Hyperledger Fabric, and MultiChain.

The features of a blockchain are described here:

Distributed consensus: Distributed consensus is the primary underpinning of a blockchain. This mechanism allows a blockchain to present a single version of the truth, which is agreed upon by all parties without the requirement of a central authority.
Transaction verification: Any transactions posted from the nodes on the blockchain are verified based on a predetermined set of rules. Only valid transactions are selected for inclusion in a block.

Platform for smart contracts: A blockchain is a platform on which programs can run to execute business logic on behalf of the users. Not all blockchains have a mechanism to execute *smart contracts*; however, this is a very desirable feature, and it is available on newer blockchain platforms such as Ethereum and MultiChain.

Smart Contracts
Blockchain technology provides a platform for running smart contracts. These are automated, autonomous programs that reside on the blockchain network and encapsulate the business logic and code needed to execute a required function when certain conditions are met. For example, think about an insurance contract where a claim is paid to the traveler if the flight is canceled. In the real world, this process normally takes a significant amount of time to make the claim, verify it, and pay the insurance amount to the claimant (traveler). What if this whole process were automated with cryptographically-enforced trust, transparency, and execution so that as soon as the smart contract received a feed that the flight in question has been canceled, it automatically triggers the insurance payment to the claimant? If the flight is on time, the smart contract pays itself.
This is indeed a revolutionary feature of blockchain, as it provides flexibility, speed, security, and automation for real-world scenarios that can lead to a completely trustworthy system with significant cost reductions. Smart contracts can be programmed to perform any actions that blockchain users need and according to their specific business requirements.

Transferring value between peers: Blockchain enables the transfer of value between its users via tokens. Tokens can be thought of as a carrier of value.
Generation of cryptocurrency: This feature is optional depending on the type of blockchain in use. A blockchain can create cryptocurrency as an incentive to its miners who validate the transactions and spend resources to secure the blockchain. We will discuss cryptocurrencies in great detail in Chapter 5, *Introducing Bitcoin*.

Smart property: It is now possible to link a digital or physical asset to the blockchain in such a secure and precise manner that it cannot be claimed by anyone else. You are in full control of your asset, and it cannot be double-spent or double-owned. Compare this with a

digital music file, for example, which can be copied many times without any controls. While it is true that many **Digital Rights Management** (**DRM**) schemes are being used currently along with copyright laws, but none of them is enforceable in such a way as blockchain based DRM can be. Blockchain can provide DRM functionality in such a way that it can be enforced fully. There are famously broken DRM schemes which looked great in theory but were hacked due to one limitation or another. One example is Oculus hack (http://www.wired.co.uk/article/oculus-rift-drm- hacked).

Another example is PS3 hack, also copyrighted digital music, films and e-books are routinely shared on the internet without any limitations. We have copyright protection in place for many years, but digital piracy refutes all attempts to fully enforce the law on a blockchain, however, if you own an asset, no one else can claim it unless you decide to transfer it. This feature has far-reaching implications, especially in DRM and electronic cash systems where double-spend detection is a crucial requirement. The double- spend problem was first solved without the requirement of a trusted third party in Bitcoin.

Provider of security: The blockchain is based on proven cryptographic technology that ensures the integrity and availability of data. Generally, confidentiality is not provided due to the requirements of transparency. This limitation is the leading barrier to its adoption by financial institutions and other industries that require privacy and confidentiality of transactions. As such, the privacy and confidentiality

of transactions on the blockchain is being researched very actively, and advancements are already being made. It could be argued that, in many situations, confidentiality is not needed and transparency is preferred. For example, with Bitcoin, confidentiality is not an absolute requirement; however, it is desirable in some scenarios. A more recent example is Zcash, which provides a platform for conducting anonymous transactions. This scheme will be discussed in detail in Chapter 8, *Alternative Coins*. Other security services, such as non-repudiation and authentication, are also provided by blockchain, as all actions are secured using private keys and digital signatures.

Immutability: This is another critical feature of blockchain: once records are added to the blockchain, they are immutable. There is the remote possibility of rolling back changes, but this is to be avoided at all costs as doing so would consume an exorbitant amount of computing resources. For example, with Bitcoin if a malicious user wants to alter previous blocks, then it would require computing the PoW once again for all those blocks that have already been added to the blockchain. This difficulty makes the records on a blockchain essentially immutable.

Uniqueness: This blockchain feature ensures that every transaction is unique and has not already been spent (double-spend problem). This feature is especially relevant with cryptocurrencies, where detection and avoidance of double spending are a vital requirement.

Types of blockchain(ppp sss tt)— BC TYPES Are proved so talented

Based on the way that blockchain has evolved over the last few years, it can be divided into multiple categories with distinct though sometimes partially-overlapping attributes. You *should* note that the tiers described earlier in the chapter are a different concept whereby the logical categorization of blockchain based on its evolution and usage is presented.

In this section, we will examine the different types of blockchains from a technical and business usage perspective. These blockchain types can occur on any blockchain tier, as there is no direct relationship between those tiers and the various types of blockchain.

In this section we'll examine:

Distributed ledgers
Distributed Ledger Technology (DLT) Blockchains
Ledgers

Distributed ledgers

First, I need to clarify an ambiguity. It should be noted that a *distributed ledger* is a broad term describing shared databases; hence, all blockchains technically fall under the umbrella of shared databases or distributed ledgers. Although all blockchains are fundamentally distributed ledgers, all distributed ledgers are not necessarily a blockchain.

A critical difference between a distributed ledger and blockchain is that a distributed ledger does not necessarily consist of blocks of transactions to keep the ledger growing.

Rather, a blockchain is a special type of shared database that is comprised of blocks of transactions. An example of a distributed ledger that does not use blocks of transactions is R3's Corda. Corda is a distributed ledger which is developed to record and manage agreements and is especially focused on financial services industry. On the other hand, more widely-known blockchains like Bitcoin and Ethereum make use of blocks to update the shared database.

As the name suggests, a distributed ledger is distributed among its participants and spread across multiple sites or organizations. This type of ledger can be either private or public. The fundamental idea here is that, unlike many other blockchains, the records are stored contiguously instead of being sorted into blocks. This concept is used in Ripple which is a blockchain and cryptocurrency based global payment network

Distributed Ledger Technology

It should be noted that over the last few years, the terms distributed ledger or **Distributed Ledger Technology** (**DLT**) have grown to be commonly used to describe blockchain in finance industry. Sometimes, blockchain and DLT are used interchangeably. Though this is not entirely accurate, it is how the term has evolved recently, especially in the finance sector. In fact, DLT is now a very active and thriving area of research in the financial sector. From a financial sector point of view, DLTs are permissioned blockchains that are shared and used between known participants. DLTs usually serve as a shared database, with all participants known and verified. They do not have a cryptocurrency or do not require mining to secure the ledger.

Public blockchains

public blockchains are not owned by anyone.

They are open to the public, and anyone can participate as a node in the decision-making process.

Users may or may not be rewarded for their participation. All users of these *permissionless* or *unpermissioned* ledgers maintain a copy of the ledger on their local nodes and use a distributed consensus mechanism to decide the eventual state of the ledger. **Bitcoin and Ethereum are both considered public blockchains.**

Private blockchains

private blockchains are just that—private.

they are open only to a group of individuals or organizations who have decided to share the ledger among themselves.

HydraChain and Quorum. Optionally, both of these blockchains can also run in public mode if required, but their primary purpose is to provide a private blockchain.

Semiprivate blockchains

This hybrid model where the private part of the blockchain remains internal and shared among known participants,

while the public part of the blockchain can still be used by anyone, optionally allowing mining to secure the blockchain.

This way, the blockchain as a whole can be secured using PoW,

semi-decentralized **model,** where it is controlled by a single entity but still allows for multiple users to join the network by following appropriate procedures.

Sidechains

More precisely known as *pegged sidechains*, this is a concept whereby coins can be moved from one blockchain to another and moved back again. Typical uses include the creation of new *altcoins* (alternative cryptocurrencies) whereby coins are burnt as a proof of an adequate stake. *Burnt* or *burning the coins* in this context means that the coins are sent to an address which is unspendable and this process makes the *burnt* coins irrecoverable. This mechanism is used to bootstrap a new currency or introduce scarcity which results in increased value of the coin.

This mechanism is also called **Proof of Burn** (**PoB**) and is used as an alternative method for distributed consensus to PoW and **Proof of Stake** (**PoS**). The aforementioned example for burning coins applies to a **one- way pegged sidechain**. The second type is called a **two-way pegged sidechain**, which allows the movement of coins from the main chain to the sidechain and back to the main chain when required.

This process enables the building of smart contracts for the Bitcoin network. Rootstock is one of the leading examples of a sidechain, which enables smart contract development for Bitcoin using this paradigm. It works by allowing a two-way peg for the Bitcoin blockchain, and this results in much faster throughput.

Permissioned ledger

A *permissioned ledger* is a blockchain where participants of the network are already known and trusted. Permissioned ledgers do not need to use a distributed consensus mechanism; instead, an agreement protocol is used to maintain a shared version of the truth about the state of the records on the blockchain. In this case, for verification of transactions on the chain, all verifiers are already preselected by a central authority and typically there is no need for a mining mechanism.

By definition, there is also no requirement for a permissioned blockchain to be private, as it can be a public blockchain but with regulated access control. For example, Bitcoin can become a permissioned ledger if an access control layer is introduced on top of it that verifies the identity of a user and then allows access to the blockchain.

Shared ledger

This is a generic term that is used to describe any application or database that is shared by the public or a consortium. Generally, all blockchains, fall into the category of a shared ledger.

Tokenized blockchains

These blockchains are standard blockchains that generate cryptocurrency as a result of a consensus process via mining or initial distribution. Bitcoin and Ethereum are prime examples of this type of blockchain.

Tokenless blockchains

These blockchains are designed in such a way that they do not have the basic unit for the transfer of value. However, they are still valuable in situations where there is no need to transfer value between nodes and only the sharing of data among various trusted parties is required. This is similar to full private blockchains, the only difference being that use of tokens is not required. This can also be thought of as a shared distributed ledger used for storing data. It does have its benefits when it comes to immutability, security, and consensus driven updates but are not used for common blockchain application of value transfer or cryptocurrency.

This ends our examination of the various type of blockchain, we'll now move in the next section to discuss the concept of census.

Consensus is a process of agreement between distrusting nodes on the final state of data. To achieve consensus, different algorithms are used. It is easy to reach an agreement between two nodes (in client-server systems, for example), but when multiple nodes are participating in a distributed system and they need to agree on a single value, it becomes quite a challenge to achieve consensus. This process of attaining agreement common state or value among multiple nodes despite the failure of some nodes is known as **distributed consensus**.

Consensus is the backbone of a blockchain and, as a result, it provides decentralization of control through an optional process known as **mining**. The choice of the **consensus algorithm** is also governed by the type of blockchain in use; that is, not all consensus mechanisms are suitable for all types of blockchains. For example, in public permissionless blockchains, it would make sense to use PoW instead of a simple agreement mechanism that is perhaps based on proof of authority. Therefore, it is essential to choose an appropriate consensus algorithm for a particular blockchain project.

Consensus mechanism

A **consensus mechanism** is a set of steps that are taken by most or all nodes in a blockchain to agree on a proposed state or value. For more than three decades, this concept has been researched by computer scientists in industry and academia. Consensus mechanisms have most recently come into the limelight and gained considerable popularity with the advent of blockchain and Bitcoin.

There are various requirements that must be met to provide the desired results in a consensus mechanism. The following describes these requirements:

Agreement: All honest nodes decide on the same value
Termination: All honest nodes terminate execution of the consensus process and eventually reach a decision
Validity: The value agreed upon by all honest nodes must be the same as the initial value proposed by at least one honest node
Fault tolerant: The consensus algorithm should be able to run in the presence of faulty or malicious nodes (Byzantine nodes)
Integrity: This is a requirement that no node can make the decision more than once in a single consensus cycle

Types of consensus mechanisms

All consensus mechanisms are developed to deal with faults in a distributed system and to allow distributed systems to reach a final state of agreement. There are two general categories of consensus mechanisms. These categories deal with all types of faults (fail stop type or arbitrary). These common types of consensus mechanisms are as follows:

Traditional Byzantine Fault Tolerance (BFT)-based: With no compute-intensive operations, such as partial hash inversion (as in Bitcoin PoW), this method relies on a simple scheme of nodes that are publisher-signed messages. Eventually, when a certain number of messages are received, then an agreement is reached.

Leader election-based consensus mechanisms: This arrangement requires nodes to compete in a leader- election lottery, and the node that wins proposes a final value. For example, the PoW used in Bitcoin falls into this category.

Consensus in blockchain

Consensus is a distributed computing concept that has been used in blockchain in order to provide a means of agreeing to a single version of the truth by all peers on the blockchain network. This concept was previously discussed in the distributed systems section of this chapter. In this section, we will address consensus in the context of blockchain technology. Some concepts presented here are still relevant to distributed systems theory, but they are explained from a blockchain perspective.

Roughly, the following describes the two main categories of consensus mechanisms:

Proof-based, leader-election lottery based, or the Nakamoto consensus whereby a leader is elected at random (using an algorithm) and proposes a final value. This category is also referred to as the *fully decentralized* or *permissionless* type of consensus mechanism. This type is well used in the Bitcoin and Ethereum blockchain in the form of a PoW mechanism.

BFT-based is a more traditional approach based on rounds of votes. This class of consensus is also known as the *consortium* or *permissioned* type of consensus mechanism.

BFT-based consensus mechanisms perform well when there are a limited number of nodes, but they do not scale well. On the other hand, leader-election lottery based (PoW) type consensus mechanisms scale very well but perform very slowly. As there is significant research being conducted in this area, new types of consensus mechanism are also emerging, such as the semi-decentralized type, which is used in the Ripple network. Ripple network will be discussed in detail in Chapter 16, *Alternative Blockchains*. There are also various other proposals out there, which are trying to find the right balance between scalability and performance. Some notable projects include PBFT, Hybrid BFT, BlockDAG, Tezos, Stellar, and GHOST.

The consensus algorithms available today, or that are being researched in the context of blockchain, are presented here. The following is not an exhaustive list, but it includes all notable algorithms.

Proof of Work (PoW): This type of consensus mechanism relies on proof that adequate computational resources have been spent before proposing a value for acceptance by the network. This scheme is used in Bitcoin, Litecoin, and other cryptocurrency blockchains. Currently, it is the only algorithm that has proven to be astonishingly successful against any collusion attacks on a blockchain network, such as the Sybil attack. The Sybil attack will be discussed in Chapter 5, *Introducing Bitcoin*.

Proof of Stake (PoS): This algorithm works on the idea that a node or user has an adequate stake in the system; that is, the user has invested enough in the system so that any malicious attempt by that user would outweigh the benefits of performing such an

attack on the network. This idea was first introduced by Peercoin, and it is going to be used in the Ethereum blockchain version called *Serenity*. Another important concept in PoS is **coin age**, which is a criterion derived from the amount of time and number of coins that have not been spent. In this model, the chances of proposing and signing the next block increase with the coin age.

Delegated Proof of Stake (DPoS): This is an innovation over standard PoS, whereby each node that has a stake in the system can delegate the validation of a transaction to other nodes by voting. It is used in the BitShares blockchain.
Proof of Elapsed Time (PoET): Introduced by Intel in 2016, PoET uses a **Trusted Execution Environment** (**TEE**) to provide randomness and safety in the leader election process via a guaranteed wait time. It requires the Intel **Software Guard Extensions** (**SGX**) processor to provide the security guarantee for it to be secure. This concept is discussed in more detail in Chapter 15, *Hyperledger*, in the context of the Intel's *Sawtooth Lake* blockchain project.

Proof of Deposit (PoD): In this case, nodes that wish to participate in the network have to make a security deposit before they can mine and propose blocks. This mechanism is used in the Tendermint blockchain. **Proof of Importance (PoI)**: This idea is significant and different from PoS. PoI not only relies on how large a stake a user has in the system, but it also monitors the usage and movement of tokens by the user in order to establish a level of trust and importance. It is used in the NEM coin blockchain. More information about this coin is available at NEM's website https://nem.io.

Federated consensus or federated Byzantine consensus: This mechanism is used in the stellar consensus protocol. Nodes in this protocol retain a group of publicly-trusted peers and propagate only those

transactions that have been validated by the majority of trusted nodes.
Reputation-based mechanisms: As the name suggests, a leader is elected by the reputation it has built over time on the network. It is based on the votes of other members.
PBFT: This mechanism achieves state machine replication, which provides tolerance against Byzantine nodes. Various other protocols including PBFT, PAXOS, RAFT, and **Federated Byzantine Agreement** (**FBA**) are also being used or have been proposed for use in many different implementations of distributed systems and blockchains.
Proof of Activity (**PoA**): This scheme is a combination of PoS and PoW, which ensures that a stakeholder is selected in a pseudorandom but uniform fashion. This is a comparatively more energy-efficient mechanism as compared to PoW. It utilizes a new concept called *Follow the Satoshi*. In this scheme, PoW and PoS are combined together to achieve consensus and good level of security. This scheme is more energy efficient as PoW is used only in the first stage of the mechanism, after the first stage it switches to PoS which consumes negligible energy. We will discuss these ideas further in Chapter 6, *Bitcoin Network and Payments* where protocols are reviewed in the context of advanced Bitcoin protocols.

Proof of Capacity (PoC): This scheme uses hard disk space as a resource to mine the blocks. This is different from PoW, where CPU resources are used. In in PoC, hard disk space is utilized for mining and as such is also known as *hard drive mining*. This concept was first introduced in the Burstcoin cryptocurrency. **Proof of Storage (PoS)**: This scheme allows for the outsourcing of storage capacity. This scheme is based on the concept that a particular piece of data is probably stored by a node *which* serves as a means to participate in the consensus mechanism. Several variations of this scheme have been proposed, such as Proof of Replication, Proof of Data Possession, Proof of Space, and Proof of Space-Time.

CAP theorem and blockchain

CAP theorem, also known as Brewer's theorem, was introduced by Eric Brewer in 1998 as conjecture. In 2002, it was proven as a theorem by Seth Gilbert and Nancy Lynch. The theory states that any distributed system cannot have consistency, availability, and partition tolerance simultaneously:

Consistency is a property which ensures that all nodes in a distributed system have a single, current, and identical copy of the data.
Availability means that the nodes in the system are up, accessible for use, and are accepting incoming requests and responding with data without any failures as and when required. In other words, data is available at each node and the nodes are responding to requests.

Partition tolerance ensures that if a group of nodes is unable to communicate with other nodes due to network failures, the distributed system continues to operate correctly. This can occur due to network and node failures.

It has been proven that a distributed system cannot have consistency, availability, and partition tolerance simultaneously. This is explained with the following example. Let's imagine that there is a distributed system with two nodes. Now let us apply the three theorem properties on this smallest of possible distributed systems only with two nodes.

Consistency is achieved if both nodes have the same shared state; that is, they have the same up-to-date copy of the data.
Availability is achieved if both nodes are up and running and responding with the latest copy of data. **Partition tolerance** is achieved if communication does not break down between two nodes (either due to network issues, Byzantine faults, and so forth), and they are able to communicate with each other.

Now think of scenario where a partition occurs and nodes can no longer communicate with each other. If no new updated data comes in, it can only be updated on one node only. In that case, if the node accepts the update, then only that one node in the network is updated and therefore consistency is lost. Now, if the update is rejected by the node, that would result in loss of availability. In that case due to partition tolerance, both availability and consistency are unachievable.

This is strange because somehow blockchain manages to achieve all of these properties—or does it? This will be explained shortly. To achieve fault tolerance, replication is used. This is a standard and widely-used method to achieve fault tolerance. Consistency is achieved using consensus algorithms in order to ensure that all nodes have the same copy of the data. This is also called **state machine replication**. The blockchain is a means for achieving state machine replication. In general, there are two types of faults that a node can experience. Both of these types fall under the broader category of faults that can occur in a distributed system:

Fail-stop fault: This type of fault occurs when a node merely has crashed. Fail-stop faults are the easier ones to deal with of the two fault types. Paxos protocol, introduced earlier in this chapter, is normally used to deal with this type of fault. These faults are simple to deal with

Byzantine faults: The second type of fault is one where the faulty node exhibits malicious or inconsistent behavior arbitrarily. This type is difficult to handle since it can create confusion due to misleading information. This can be a result of an attack by adversaries, a software bug, or data corruption. State machine replication protocols such as PBFT was developed to address this second type of faults.

Strangely, it seems that the CAP theorem is violated in the blockchain, especially in its most successful implementation, Bitcoin. However, this is not the case. In blockchains, consistency is sacrificed in favor of availability and partition tolerance. In this scenario, **Consistency** (**C**) on the blockchain is not achieved simultaneously with **Partition tolerance** (**P**) and **Availability** (**A**), but it is achieved over time. This is called eventual consistency, where consistency is achieved as a result of validation from multiple nodes over time. The concept of mining was introduced in Bitcoin for this purpose. **Mining** is a process that facilitates the achievement of consensus by using the PoW consensus algorithm. At a higher level, mining can be defined as a process that is used to add more blocks to the blockchain. More on this later in Chapter 5, *Introducing Bitcoin*.

Elliptic curves

The **elliptic curves algorithm** is based on the discrete logarithm problem discussed earlier but in the context of elliptic curves. An **elliptic curve** is an algebraic cubic curve over a field, which can be defined by the following equation. The curve is non-singular, which means that it has no cusps or self-intersections. It has two variables *a* and *b*, as well as a point of infinity.

Here, *a* and *b* are integers whose values are elements of the field on which the elliptic curve is defined. Elliptic curves can be defined over real numbers, rational numbers, complex numbers, or finite fields. For cryptographic purposes, an elliptic curve over prime finite fields is used instead of real numbers. Additionally, the prime should be greater than 3. Different curves can be generated by varying the value of *a* and/or *b*.

The most prominently used cryptosystems based on elliptic curves are the **Elliptic Curve Digital Signature Algorithm** (**ECDSA**) and the **Elliptic Curve Diffie-Hellman** (**ECDH**) key exchange.

To understand public key cryptography, the key concept that needs to be explored is the concept of public and private keys.

Elliptic Curve Cryptography

Elliptic Curve Cryptography (**ECC**) is based on the discrete logarithm problem founded upon elliptic curves over finite fields (Galois fields). The main benefit of ECC over other types of public key algorithms is that it requires a smaller key size while providing the same level of security as, for example, RSA. Two notable schemes that originate from ECC are ECDH for key exchange and ECDSA for digital signatures.

ECC can also be used for encryption, but it is not usually used for this purpose in practice. Instead, it is used for key exchange and digital signatures commonly. As ECC needs less space to operate, it is becoming very popular on embedded platforms and in systems where storage resources are limited. By comparison, the same level of security can be achieved with ECC only using 256-bit operands as compared to 3072-bits in RSA.

Elliptic curve cryptography (ECC) is a type of public-key cryptography based on the mathematical properties of elliptic curves over finite fields. In the context of blockchain technology, ECC is often used for digital signatures and encryption to secure transactions and data on the blockchain. Here's how ECC works in the context of blockchain:

1. **Key Generation**: In ECC, each participant generates a pair of cryptographic keys: a public key and a private key. The private key is kept secret and used to create digital signatures or decrypt messages, while the public key is shared openly and used to verify signatures or encrypt messages.

2. **Elliptic Curve Parameters**: The choice of elliptic curve parameters (curve equation, base point, prime modulus) is critical for the security of ECC. Blockchain protocols typically specify standardized elliptic curves with well-established security properties, such as the secp256k1 curve used in Bitcoin.

3. **Digital Signatures**: To sign a transaction on the blockchain, the sender uses their private key to generate a digital signature over the transaction data. The recipient can verify the signature using the sender's public key, ensuring that the transaction was created by the rightful owner of the private key.

4. **Public Key Cryptography**: ECC enables secure communication between participants on the blockchain network. Messages can be encrypted using the recipient's public key and decrypted only by the recipient using their private key, providing confidentiality and integrity of data transmission.

5. **Security Properties**: ECC offers several security advantages compared to other cryptographic algorithms, including smaller key sizes for equivalent security levels, faster computation times, and resistance to quantum computing attacks.

6. **Transaction Verification**: Each node on the blockchain network independently verifies the validity of transactions using ECC-based digital signatures. Nodes validate that transactions are correctly signed by the sender and adhere to consensus rules before adding them to the blockchain.

7. **Wallet Addresses**: In blockchain systems, wallet addresses are derived from public keys using cryptographic hash functions. Users can receive funds by sharing their wallet addresses publicly, while keeping their private keys secure to authorize transactions.

Overall, elliptic curve cryptography plays a crucial role in securing blockchain networks by providing efficient and robust cryptographic primitives for key generation, digital signatures, and encryption. Its adoption has enabled the widespread deployment of secure and decentralized blockchain-based systems for various applications, including cryptocurrencies, smart contracts, and decentralized finance (DeFi).

Secure Hash Algorithms

The following list describes the most common **Secure Hash Algorithms** (**SHAs**):

SHA-0: This is a 160-bit function introduced by NIST in 1993.
SHA-1: SHA-1 was introduced in 1995 by NIST as a replacement for SHA-0. This is also a 160-bit hash function. SHA-1 is used commonly in SSL and TLS implementations. It should be noted that SHA-1 is now considered insecure, and it is being deprecated by certificate authorities. Its usage is discouraged in any new implementations.
SHA-2: This category includes four functions defined by the number of bits of the hash: SHA-224, SHA- 256, SHA-384, and SHA-512.

SHA-3: This is the latest family of SHA functions. SHA-3-224, SHA-3-256, SHA-3-384, and SHA-3-512 are members of this family. SHA-3 is a NIST-standardized version of Keccak. Keccak uses a new approach called **sponge construction** instead of the commonly used Merkle-Damgard transformation.
RIPEMD: RIPEMD is the acronym for **RACE Integrity Primitives Evaluation Message Digest**. It is based on the design ideas used to build MD4. There are multiple versions of RIPEMD, including 128-bit, 160-bit, 256-bit, and 320-bit.

Whirlpool: This is based on a modified version of the Rijndael cipher known as *W*. It uses the Miyaguchi- Preneel compression function, which is a type of one-way function used for the compression of two fixed- length inputs into a single fixed-length output. It is a single block length compression function.

Hash functions have many practical applications ranging from simple file integrity checks and password storage to use in cryptographic protocols and algorithms. They are used in hash tables, distributed hash tables, bloom filters, virus fingerprinting, peer-to-peer file sharing, and many other applications.

Hash functions play a vital role in blockchain. Especially, The PoW function in particular uses SHA-256 twice in order to verify the computational effort spent by miners. RIPEMD 160 is used to produce Bitcoin addresses. This will be discussed further in later chapters.

In the next section, the design of the SHA algorithm is introduced.

Design of SHA-256

SHA-256 has the input message size < 2^{64}-bits. Block size is 512-bits, and it has a word size of 32-bits. The output is a 256-bit digest.

The compression function processes a 512-bit message block and a 256-bit intermediate hash value. There are two main components of this function: the compression function and a message schedule.

The algorithm works as follows, in eight steps: 1. **Preprocessing**:

1. Padding of the message is used to adjust the length of a block to 512-bits if it is smaller than the required block size of 512-bits.

2. Parsing the message into message blocks, which ensures that the message and its padding is divided into equal blocks of 512-bits.

3. Setting up the initial hash value, which consists of the eight 32-bit words obtained by taking the first 32-bits of the fractional parts of the square roots of the first eight prime numbers. These initial values are randomly chosen to initialize the process, and they provide a level of confidence that no backdoor exists in the algorithm.

2. **Hash computation**:

4. Each message block is then processed in a sequence, and it requires 64 rounds to compute the full hash output. Each round uses slightly different constants to ensure that no two rounds are the same.

5. The message schedule is prepared.

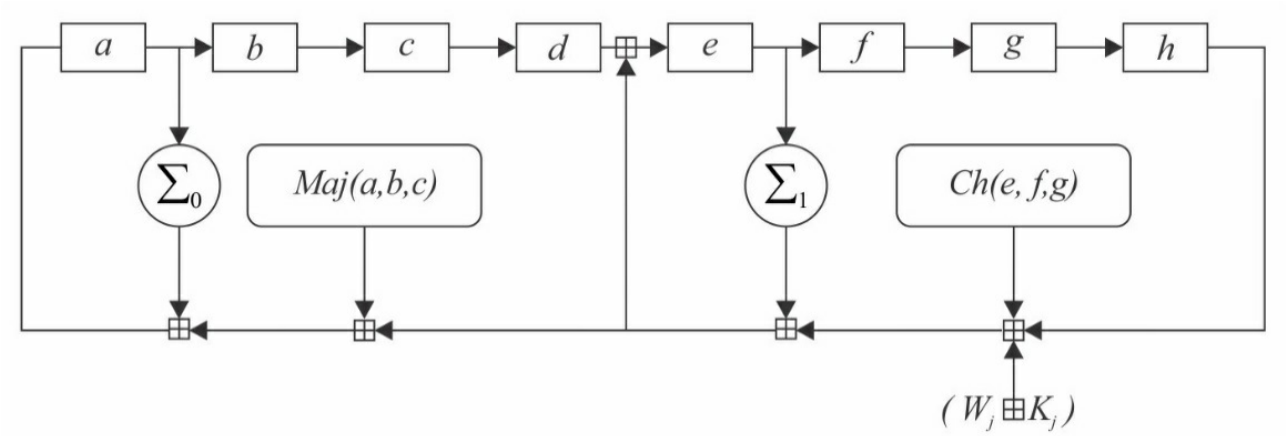
6. Eight working variables are initialized.

7. The intermediate hash value is calculated.

8. Finally, the message is processed, and the out put hash is produced:
One round of a SHA-256 compression function

In the preceding diagram, **a**, **b**, **c**, **d**, **e**, **f**, **g**, and **h** are the registers. *Maj* and *Ch* are applied bitwise. $\mathbf{\Sigma_0}$ and $\mathbf{\Sigma_1}$

performs bitwise rotation. Round constants are $\mathbf{W_j}$ and $\mathbf{K_j}$, which are added, *mod* 2^{32}.



193

SHA stands for "Secure Hash Algorithm," and it's a family of cryptographic hash functions designed by the National Security Agency (NSA) in the United States. The primary purpose of a hash function like SHA is to take input data and produce a fixed-size hash value, typically a string of numbers and letters.

Purpose of SHA:-SIP

- **Data Integrity**: SHA algorithms are used to verify the integrity of data. If even a small part of the input data changes, the hash value will change significantly, indicating that the data has been altered.

- **Digital Signatures**: They are used in digital signatures, where the hash of a message is encrypted with a private key to create a signature. The recipient can use the sender's public key to decrypt and verify the hash.

- **Password Hashing**: SHA algorithms are also used to hash passwords securely. Instead of storing the password itself, a system stores its hash value. When a user enters their password, it's hashed and compared to the stored hash.

Types of SHA Algorithms:

1. **SHA-1 (Secure Hash Algorithm 1)**:

   - SHA-1 was one of the earliest members of the SHA family, producing a 160-bit hash value.

   - It's no longer considered secure against well-funded attackers due to vulnerabilities, and it's recommended to use stronger alternatives.

2. **SHA-2 (Secure Hash Algorithm 2)**:

   - SHA-2 includes a family of hash functions with various output sizes, such as SHA-256, SHA-384, SHA-512, etc.

   - **SHA-256**: Produces a 256-bit hash value. Widely used for data integrity, digital signatures, and password hashing.

   - **SHA-384**: Produces a 384-bit hash value. Offers stronger security but slower performance compared to SHA-256.

   - **SHA-512**: Produces a 512-bit hash value. Similar to SHA-384 but with a longer output.

3. **SHA-3 (Secure Hash Algorithm 3)**:

- SHA-3 is the latest member of the SHA family, designed as the new standard after SHA-2.

- It was chosen through a public competition and provides improved security features.

- SHA-3 includes hash functions with different output sizes, such as SHA3-224, SHA3-256, SHA3-384, SHA3-512.

How SHA Algorithms Work:

- **Message Padding**: Input messages are padded to meet specific block sizes.

- **Block Processing**: The padded message is divided into blocks, and each block goes through a series of transformations.

- **Compression Function**: Each block is processed using a compression function that combines the current block with the previous hash.

- **Final Hash**: After processing all blocks, a final hash value is produced, which is a fixed-size output.

Security:

- The security of SHA algorithms relies on properties like collision resistance, where it's computationally infeasible to find two different inputs that produce the same hash.

- Preimage resistance ensures that given a hash value, it's hard to find an input that hashes to that value.

- These properties make it difficult for attackers to reverse-engineer the original data from its hash.

Conclusion:

SHA algorithms are fundamental in modern cryptography and are used in various applications to ensure data integrity, provide digital signatures, and securely store passwords. The choice of which SHA algorithm to use depends on factors like security requirements, performance considerations, and compatibility with existing systems. It's important to stay updated on the latest recommendations for cryptographic best practices.

SHA-256 (Secure Hash Algorithm 256-bit) is another member of the SHA-2 family of cryptographic hash functions. It produces a fixed-size (256-bit) hash value from input data. Here's a detailed explanation of the SHA-256 algorithm:

Overview:

SHA-256 operates on 32-bit words and uses 64 rounds of hashing. It takes an input message and produces a fixed-size (256-bit) hash value. The main steps involved are:

1. **Padding**: The input message is padded to a length that is a multiple of 512 bits.

2. **Processing**: The padded message is processed in blocks of 512 bits each.

3. **Compression**: Each block goes through a series of transformations to produce the final hash.

Step-by-Step Explanation:

1. **Padding**:

   - The input message is padded to ensure its length is a multiple of 512 bits.

   - A '1' bit is appended to the message.

   - Then '0' bits are appended until the length of the padded message is 448 bits modulo 512.

   - Finally, the original message length (before padding) is represented in binary and appended as a 64-bit big-endian integer.

   For example, if the original message length is 400 bits:

   - Append a '1' bit: `... 01`

   - Append '0' bits: `... 0100 0000`

   - Append the 64-bit message length: `... 0100 0000 0000 0000 0000 0000 0000 0000`

2. **Processing**:

   - Divide the padded message into 512-bit blocks.

   - Initialize eight 32-bit words (A, B, C, D, E, F, G, H) with specific constants

- Break each 512-bit block into 16 32-bit words, creating a 64-word message schedule array, W[0..63].

3. **Compression**:

- For each block, perform 64 rounds of hashing.

- Each round updates the values of A, B, C, D, E, F, G, and H.

- The operations in each round are a combination of logical functions, bitwise operations (such as AND, OR, XOR), addition modulo 2^32, and shifting.

- The values of A, B, C, D, E, F, G, and H are updated based on these operations and the current word from the message schedule array.

4. **Final Hash**:

- After processing all blocks, concatenate the final values of A, B, C, D, E, F, G, and H to get the 256-bit hash.

- The hash is the concatenation of the hexadecimal representations of these eight 32-bit words.

Security:

SHA-256 is designed to be collision-resistant, meaning it is computationally infeasible to find two different inputs that produce the same hash output. It also provides resistance against preimage and second preimage attacks, making it a strong choice for secure hashing.

Usage:

SHA-256 is widely used in various security applications, such as digital signatures, certificate generation, password hashing, and integrity verification. It's also used in blockchain technologies like Bitcoin for mining and transaction verification.

This explanation provides a high-level overview of the SHA-256 algorithm. For a more in-depth understanding, it's beneficial to study the specific logical and bitwise operations involved in each step of the algorithm.

SHA-512 (Secure Hash Algorithm 512-bit) is one of the family of Secure Hash Algorithms (SHA) developed by the National Security Agency (NSA) in the United States. It is widely used in various security applications and cryptographic protocols. Here's a detailed explanation of the SHA-512 algorithm:

Overview:

SHA-512 operates on 64-bit words and uses 80 rounds of hashing. It takes an input message and produces a fixed-size (512-bit) hash value. The main steps involved are:

1. **Padding**: The input message is padded to a length that is a multiple of 1024 bits.

2. **Processing**: The padded message is processed in blocks of 1024 bits each.

3. **Compression**: Each block goes through a series of transformations to produce the final hash.

Step-by-Step Explanation:

1. **Padding**:

   - The input message is padded to ensure its length is a multiple of 1024 bits.

   - A '1' bit is appended to the message.

   - Then '0' bits are appended until the length of the padded message is 896 bits modulo 1024.

   - Finally, the original message length (before padding) is represented in binary and appended as a 128-bit big-endian integer.

   For example, if the original message length is 600 bits:

   - Append a '1' bit: `... 01`

   - Append '0' bits: `... 0100 0000`

   - Append the 128-bit message length: `... 0100 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 1011 1000`

2. **Processing**:

   - Divide the padded message into 1024-bit blocks.

   - Initialize eight 64-bit words (A, B, C, D, E, F, G, H) with specific constants (the first 64 bits of the fractional parts of the square roots of the first 8 primes 2-19).

   - Break each 1024-bit block into 64 64-bit words, creating a 64-word message schedule array, W[0..63].

3. **Compression**:

   - For each block, perform 80 rounds of hashing.

   - Each round updates the values of A, B, C, D, E, F, G, and H.

   - The operations in each round are a combination of logical functions, bitwise operations (such as AND, OR, XOR), addition modulo 2^64, and shifting.

   - The values of A, B, C, D, E, F, G, and H are updated based on these operations and the current word from the message schedule array.

4. **Final Hash**:

   - After processing all blocks, concatenate the final values of A, B, C, D, E, F, G, and H to get the 512-bit hash.

   - The hash is the concatenation of the hexadecimal representations of these eight 64-bit words.

Security:

SHA-512 is designed to be collision-resistant, meaning it is computationally infeasible to find two different inputs that produce the same hash output. It also provides resistance against preimage and second preimage attacks, making it a strong choice for secure hashing.

Usage:

SHA-512 is commonly used in various security applications, such as digital signatures, certificate generation, and integrity verification. It's also used in cryptocurrencies like Bitcoin for mining and transaction verification.

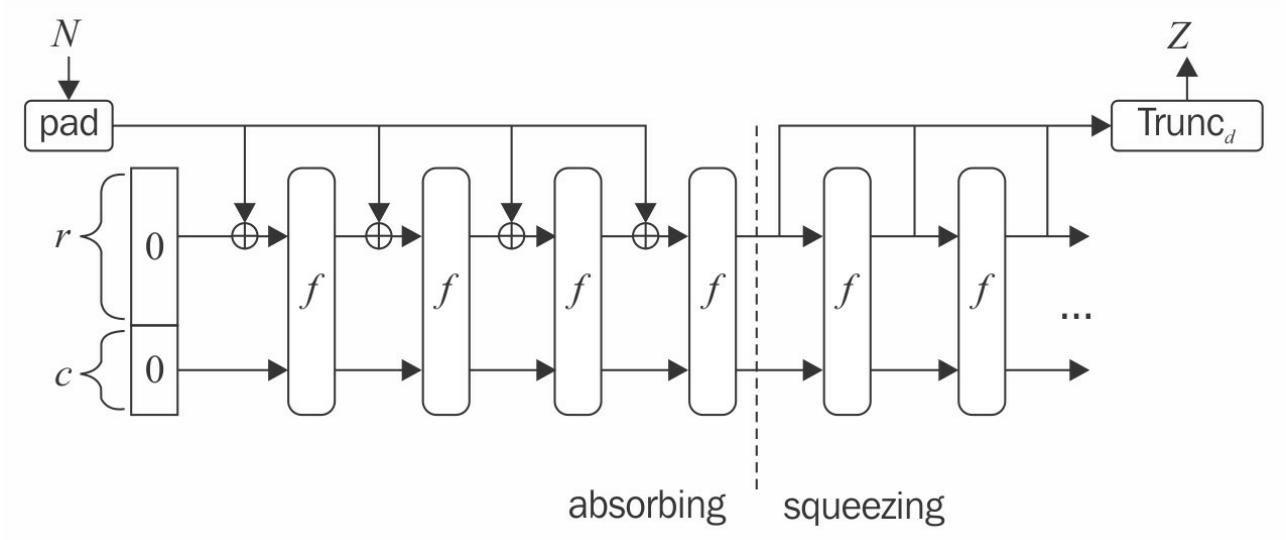
This explanation provides a high-level overview of the SHA-512 algorithm. For a more in-depth understanding, it's beneficial to study the specific logical and bitwise operations involved in each step of the algorithm.

Design of SHA-3 (Keccak)

The structure of SHA-3 is very different from that of SHA-1 and SHA-2. The key idea behind SHA-3 is based on unkeyed permutations, as opposed to other typical hash function constructions that used keyed permutations. Keccak also does not make use of the Merkle-Damgard transformation that is commonly used to handle arbitrary-length input messages in hash functions. A newer approach called **sponge and squeeze construction** is used in Keccak. It is a random permutation model. Different variants of SHA-3 have been standardized, such as SHA-3-224, SHA-3-256, SHA-3-384, SHA-3-512, SHAKE-128, and SHAKE-256. SHAKE-128 and SHAKE-256 are **Extendable Output Functions** (**XOFs**), which are also standardized by NIST. XOFs allow the output to be extended to any desired length.

The following diagram shows the sponge and squeeze model, which is the basis of SHA-3 or Keccak. Analogous to a sponge, the data is first absorbed into the sponge after applying padding. There it is then changed into a subset of permutation state using XOR, and then the output is squeezed out of the sponge function that represents the transformed state. The rate is the input block size of a sponge function, while capacity determines the general security level:

SHA-3 absorbing and squeezing function



Elliptic Curve Cryptography (ECC)

Introduction to ECC:

Elliptic Curve Cryptography (ECC) is a public-key cryptographic system based on the algebraic structure of elliptic curves over finite fields. It offers strong security with shorter key lengths compared to other public-key systems like RSA. ECC is widely used in applications where efficiency and security are crucial, such as in mobile devices and Internet of Things (IoT) devices.

Key Components of ECC:

1. **Elliptic Curves**:

   - An elliptic curve is a set of points that satisfy a specific mathematical equation. The equation for an elliptic curve in Weierstrass form is:

     ```

     y^2 = x^3 + ax + b

     ```

     Where `a` and `b` are constants.

   - The curve is symmetric around the x-axis and may intersect the x-axis at one or more points.

   - Points on the curve are often represented as (x, y) coordinates.

2. **Point Addition on Elliptic Curves**:

- ECC uses a special operation called "point addition" to add two points on the curve to get a third point.

- Point addition involves drawing a line through the two points and finding the third point where the line intersects the curve. This intersection point is reflected over the x-axis to get the result.

- If the line is vertical (i.e., the two points have the same x-coordinate), the result is the point at infinity (the identity element).

3. **Scalar Multiplication**:

- Scalar multiplication is repeatedly adding a point to itself a certain number of times (scalar multiplication).

- For example, `k * P` means adding point `P` to itself `k` times.

- Scalar multiplication is a fundamental operation in ECC and forms the basis of the cryptographic system.

Key Generation and Public/Private Keys:

- ECC uses the concept of key pairs: public and private keys.

- **Key Generation**:

  - A user's private key (`d`) is a randomly chosen number within a specific range.

  - The corresponding public key (`Q`) is calculated by multiplying a special point on the curve (called the base point or generator point) `G` by the private key:

    ```

    Q = d * G

    ```

- **Public Key**: The public key is used for encryption and verification.

- **Private Key**: The private key is kept secret and used for decryption and signing.

Security and Strengths of ECC:

- **Security**: ECC's security is based on the difficulty of the Elliptic Curve Discrete Logarithm Problem (ECDLP). This problem involves finding `d` given `Q` and `G` in the equation `Q = d * G`, which is believed to be computationally infeasible.

- **Key Length**: ECC offers equivalent security to RSA with much shorter key lengths. For example, a 256-bit ECC key provides similar security to a 3072-bit RSA key.

- **Efficiency**: ECC operations are faster and require less computational power compared to other public-key systems. This makes it ideal for resource-constrained devices.

Applications of ECC:

- **Secure Communication**: ECC is used in protocols like TLS/SSL for securing web communications.

- **Digital Signatures**: ECC can be used for creating and verifying digital signatures, ensuring data integrity and authenticity.

- **Encryption**: ECC can be used for encrypting and decrypting data securely.

- **Mobile Devices and IoT**: ECC's efficiency makes it well-suited for devices with limited resources, such as smartphones, smart cards, and IoT devices.

Conclusion:

Elliptic Curve Cryptography (ECC) is a secure ,powerful and efficient public-key cryptographic system based on the mathematics of elliptic curves.

It offers strong security with shorter key lengths, making it ideal for a wide range of applications

ECC involves grasping the concepts of

elliptic curves,

point addition,

scalar multiplication,

key generation,

security principles

Advantages of ECC

The advantages of ECC are as follows:

Shorter key lengths: ECC provides a greater level of security while having a shorter key size. The energy required to break a 3072-length key generated by the RSA encryption method is the same amount of energy required to break a 256-length key generated by ECC. This shows that the 256-length key generated by the ECC is as cryptographically as strong as the 3072-length one generated by the RSA encryption.

Less computational power: As the key sizes are shorter, it is computationally less expensive to encrypt and decrypt the data.

Fast key generation: The key generation process is relatively simple and computationally less expensive as it involves securely

generating a random integer within a specified range. This makes it extremely fast compared to the RSA encryption algorithm.

Fast signatures: The ECC uses the ECDSA, which is extremely fast while generating the digital signature as it involves the simple step of multiplying a point on the curve.

Disadvantages of ECC

The disadvantages of ECC are as follows:

Complicated: The ECC is quite complicated to implement, making it more prone to errors, thus compromising the system's overall security if not implemented properly.

Random number generator: The system's security is compromised if a broken random number generator is used at the time of private key selection.

Patents: It is one of the main factors restricting the widespread use of the ECC algorithm. Certicom and National Security Agency NSA own some of the main patents.

Signature verification: Although the signature generation process is fast, the verification process takes some time due to its computationally intensive calculations.

Backdoor: There are growing concerns that NSA may have implemented a backdoor into the ECC algorithms allowing them to monitor the encrypted messages being sent.

Elliptic Curve Cryptography (ECC) plays a crucial role in the realm of blockchain technology, particularly in

securing transactions,

generating addresses, and

providing cryptographic signatures.

Key Generation and Address Generation:

- **Public and Private Keys**: In blockchain systems like Bitcoin, users have a pair of cryptographic keys
- - **ECC for Key Generation**: ECC is used to generate these keys. A private key is a randomly chosen number, and the corresponding public key is derived through ECC operations.

- **Public Key Hash**: To create a user's public address, the public key is hashed using a specific algorithm (like SHA-256 or RIPEMD-160). This hash is then encoded to create the public address, which is the address used for receiving funds.

Digital Signatures:

- **Transaction Signing**: When a user wants to send cryptocurrency from their address, they create a transaction and sign it with their private key.

- **ECC for Signatures**: ECC is used to create the digital signature. The signature is a mathematical operation involving the private key and the transaction data.

- **Verification**: Others can verify the signature using the sender's public key and the signed transaction. If the verification is successful, it means the transaction was indeed signed by the owner of the private key.

Security and Efficiency:

- **Security Strength**: ECC's strength in blockchain lies in its security properties, such as the difficulty of solving the Elliptic Curve Discrete Logarithm Problem (ECDLP).

- **Shorter Key Length**: ECC's shorter key lengths compared to RSA mean that blockchain networks can achieve the same level of security with smaller data sizes.

- **Efficiency**: ECC's efficiency in key generation and signature verification is crucial for the performance of blockchain networks, especially in systems where many transactions need to be processed quickly.

Bitcoin and ECC:

- **Bitcoin's Use of ECC**: Bitcoin, the most well-known blockchain system, uses ECC extensively.

- **Address Format**: Bitcoin addresses are derived from the public key through ECC and hashing. Users share their addresses to receive funds.

- **Transaction Verification**: Transactions in the Bitcoin network are verified using digital signatures created through ECC.

- **Mining Rewards**: Miners in the Bitcoin network use ECC for their mining operations, including creating blocks and securing the network.

Ethereum and ECC:

- **Smart Contracts**: Ethereum, another popular blockchain platform, also uses ECC.

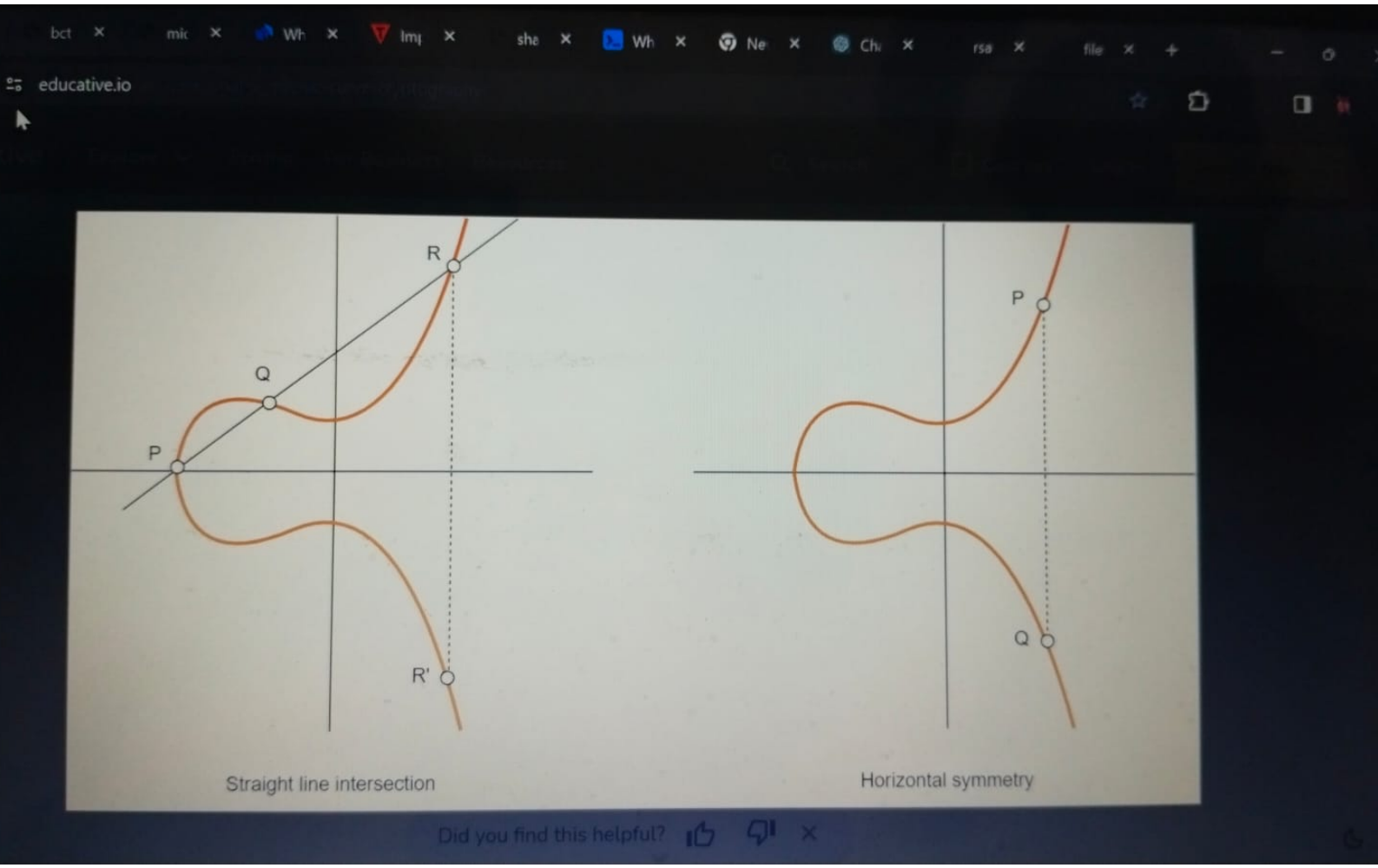
- **Public and Private Keys**: Ethereum users have ECC key pairs for interacting with the network, including deploying smart contracts.

- **Transaction Signing**: Transactions on the Ethereum network are signed with ECC-based digital signatures.

- **Address Generation**: Ethereum addresses are derived from the public key using ECC operations.

Conclusion:

Elliptic Curve Cryptography (ECC) is an essential component of blockchain technology, providing the cryptographic security needed for key generation, address creation, and transaction signing. Its efficiency and strength make it a popular choice for securing blockchain networks like Bitcoin and Ethereum. Understanding ECC is crucial for anyone working with or developing applications for blockchain, as it forms the backbone of security and trust within these decentralized systems.



10) Tokenless blockchains

- not real
- doesn't do any transaction but share data btw trusted parties.

Types of Consensus mechanisms

1) Proof of Work (Bit Coin)
- relies on proof that enough computational resources have been spent before proposing a value for acceptance by the network.

2) Proof of stake (Peer Coin) (geeksforgeeks)
- user has enough stake in the system; for example the user has invested enough in the system so that any malicious attempt would outweigh the benefits of performing an attack on the system.
- used in peer coin & Etherium blockchain
- stake is value/money we bet on a certain outcome.
- Coin-age based selection
- Random Block Selection (lowest hash value & highest stake)

Advantages
- Energy efficient
- Decentralization
- Security

3) Delegated Proof-of-stake

4) Proof of elapsed time
- uses TEE (Trusted Execution Environment)
- used in Intel Sawtooth Lake Blockchain project.

5) Deposit-based Consensus

6) Proof of importance
- monitors the usage & movement of tokens by the user to establish a level of trust and importance.
- used in Nemcoin

7) Federated Concensus (or) Federated Byzantine Concensus

8) Reputation based Mechanisms

Benefits & Limitations of Blockchain

Benefits
- Decentralization
- Transparency & Trust
- Immutability
- High availability
- Highly Secure
- Simplification of current paradigms
- Faster dealings
- Cost saving

challenges / Limitations
- Scalability
- Adaptability
- Regulation
- Relatively immature technology
- Privacy

5/3/24

Mechanisms & Services

Authentication
- Entity Authentication
- Data origen authentication
  - Methods — Digital Signature
  - Msg Authentication Codes (Macs)

Generic elements of a block chain

- Addresses
- Transaction
- Block
- Peer-to-peer network
- Scripting or programming
- Virtual machine
- State machine
- Nodes
- Smart Contracts

20/2/24

Features of Blockchain

→ Distributed Consensus
→ Transaction Verification
→ Transferring value btw Peers
→ Generating Crypto Currency
→ Smart preoperty
→ Provider of security
→ Immutability
→ Uniqueness
→ Smart Contracts

direct tagged to next

Computingability & Proof of work

Proof of work Consensus

The three pillars of Distributed Concensus

- Distributed System
- Principles of Cryptography
- Economic models

Majority Voting

FLP Impossibility Theorem

* Blockchain → Hash chain + Proof of work (Puzzle solving)

* Classical distributed Consensus can be applied on blockchain for cryptocurrencies.

Applications of Block chain

- Supply chain management
- Health records
- Agriculture
- Design of database, websites

How blockchains accumulate blocks

1) A node starts a transaction by signing it with its private key.
2) the transaction is propagated (flooded) by using much desirable Gossip protocol to peers, which validates the transaction based on pre-set Criteria. More than one node is required to validate the transactions
3) Once transaction is validated, it is included in a block, which is then propagated on to the network. At this point, the transaction is said to be confirmed.
4) Second Confirmation
5) Total 6 Confirmations are required } Non Compulsory steps

8/2/24

BCT

TB - Bashir, Mastering blockchain

simplilearn.com
geekflare.com
edx.org/learn/blockchain

- Byzantine Node
- CAP Theorm
  - Consistency, Availability, Partition tolerance
- Consensus mechanism
  - Agreement
- State machine replication

⇒ Blockchain is a method to achieve Statemachine Replication.

→ Byzantine Generals Problem

→ Proof of Work (PoW) is a mechanism to achieve Consensus algorithm

Consensus Mechanism Requirements

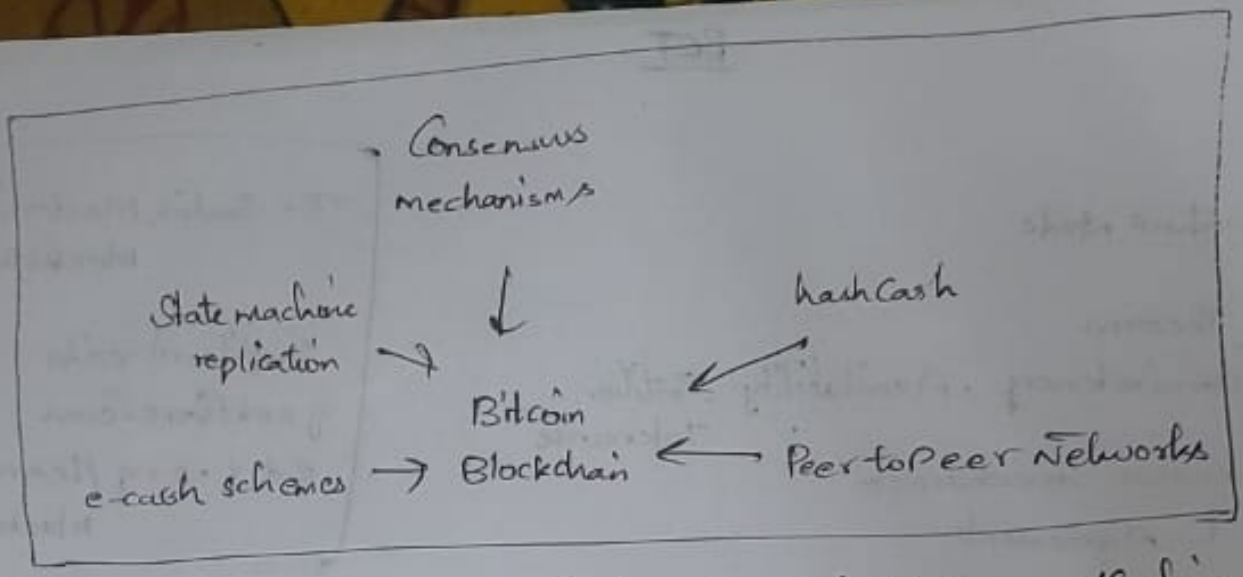
- Agreement
- Termination
- Validity
- Fault tolerant
- Integrity.

Types of Consensus Mechanisms

- Byzantine fault-tolerance based
- Leader based Consensus mechanisms

→ Proposer, acceptor, learner
→ Candidate or leader

→ Hash Cash used in Bitcoin mining process



Def: Bitcoin is a new electronic cash system that is fully peer-to-peer with no trusted third party

Blockchain is a trustless decentralization network

Blockchain at its core is a peer-to-peer distributed ledger that is cryptographically secure, append-only, immutable (extremely hard to change), and updateable only via consensus or agreement among ~~peers~~ peers

Block consists of ordered set of transactions

→ first block in the block chain is called as genesis block

Previous hash

Transactions & other data

(Genesis block)

-> A block is a container data structure that contains series of transactions.
-> A block may contain more than 500 transactions on average.

Merkle root

Bits are written in Hex, ex:- 0x170e
First byte is index & next three bytes form coefficient
Target = Coeff * 2^(8 * (index - 3))

Block Generation Cost

=> Energy efficiency . ~0.095 J/GH = ~100 J/TH
=> ASIC Hardware for bitcoin can perform about 2.5TH/s.
=> Hash rate approx. 100MTH/s
=> Network Consumes about 80TW-hours

Transactions in a block

=> there are organised as Merkle Tree.
=> The merkle root is used to construct the block hash
=> If you change a transaction, you need to change all the ~~block~~ subsequent block hashes

Bitcoin Scripts

-> Simple, compact, stack-based and processed left to right
-> FORTH like language
-> Not turing Complete (no loops)
-> Halting prblm is not there.

a paper on bitcoin, Bitcoin: A peer-to-peer Electronic Cash System by Satoshi Nakamoto

Accountability

Cryptographic primitives

Message Authentication modes,

Cryptographic hashes

Elliptic Curves
- non singular linear curve
- ECDSA - Elliptic Curve digital signature algorithm.

Hash functions

Collision Resistance

Blockchaindemo.io

Bitcoin - protocol
bitcoin - currency

12/3/24

Bitcoin

=> Bitcon is a fully decentralized digital currency.
-> It is the first application of bct.
=> Extremely secure & stable
=> There are special nodes called Miners
=> Miners propose new blocks - solve the puzzle and add the soln as a proof of solving the challenge to be the leader.

Distributed Ledger Technology (DLT) - Blockchain 1.0

Dapps (Smart Contracts)

=> Block chain is a platform for executing transactional services.

Block consists of 2 pairs - header (PoW, Previousblock (pointer)), Transactions

Blockchain technology has a wide range of applications across various industries due to its security, transparency, and decentralization. Here are some notable applications:

1. **Cryptocurrencies**: The most famous application of blockchain is in cryptocurrencies like Bitcoin and Ethereum. Blockchain enables secure, transparent, and decentralized digital currencies.

2. **Supply Chain Management**: Blockchain can track goods as they move through a supply chain. This transparency helps in verifying the authenticity and quality of products, reducing fraud, and ensuring ethical sourcing.

3. **Smart Contracts**: These are self-executing contracts with the terms directly written into code. They automatically enforce and execute the terms of agreements when predefined conditions are met. This finds application in various fields like real estate, insurance, and legal contracts.

4. **Healthcare**: Blockchain can securely store patient records, ensuring privacy and easy access by authorized personnel. It also helps in tracking the authenticity of drugs in the pharmaceutical supply chain.

5. **Voting Systems**: Blockchain can create transparent and secure voting systems. It ensures the integrity of the electoral process by providing an immutable record of votes.

6. **Identity Management**: Blockchain can be used to create digital identities that are secure and verifiable, reducing identity theft and fraud.

7. **Finance and Banking**: Apart from cryptocurrencies, blockchain can be used for faster and more secure cross-border payments, trade finance, and reducing the need for intermediaries in financial transactions.

8. **Gaming**: In the gaming industry, blockchain can enable ownership of in-game assets, creating a transparent and secure environment for buying, selling, and trading virtual items.

9. **Real Estate**: Blockchain can streamline property transactions by providing transparent records of ownership, reducing fraud and the need for intermediaries.

10. **Energy Management**: It can help in managing energy grids more efficiently by tracking energy generation, distribution, and consumption in a transparent manner.

These are just a few examples, and the potential applications of blockchain continue to expand as the technology matures and innovators find new ways to utilize its capabilities.