UNIT I
INTRODUCTION TO BLOCKCHAIN

BLOCKCHAIN

Blockchain is a shared, immutable ledger that facilitates the process of recording transactions and tracking assets in a business network.

A blockchain is "a distributed database that maintains a continuously growing list of ordered records, called blocks." These blocks "are linked using cryptography. Each block contains a cryptographic hash of the previous block, a timestamp, and transaction data. A blockchain is a decentralized, distributed and public digital ledger that is used to record transactions across many computers so that the record cannot be altered retroactively without the alteration of all subsequent blocks and the consensus of the network.

PUBLIC LEDGERS

The ledger is the book in which the user's transactions on the network are recorded. The ledger is like a book where everything is recorded to maintain security, privacy, and transparency in the network. It is shared among all the users on the network.

a ledger is a database or a list of every transaction that has ever taken place on the network. This decentralized ledger, known as a blockchain, is maintained by a network of computers, or nodes, who work together to verify and record transactions.

- A consensus mechanism, like Proof of Work or Proof of Stake, verifies each transaction on the blockchain to make sure it is legitimate and not fraudulent. The transaction is added to a block and then added to the blockchain when it has been confirmed.
- A transaction that has been added to the blockchain cannot be changed or removed since the ledger is immutable. By doing this, fraud is avoided and the ledger's integrity is guaranteed.
- Users may follow the movement of cryptocurrency from one address to another using the ledger. This makes it possible to confirm the legitimacy of transactions and their owner, as well as to prevent double-spending.

BLOCKCHAIN AS PUBLIC LEDGERS

A blockchain ledger is a distributed database that is maintained by a network of computers, which uses cryptography to ensure that each transaction is secure and cannot be altered retroactively without the consensus of the network.

- The ledger is made up of blocks, which contain a list of transactions, a timestamp, and a cryptographic hash of the previous block.
- Each new block is added to the chain in a linear, chronological sequence, forming a permanent, immutable record of all transactions that have occurred on the network.
- The decentralized nature of the blockchain ledger ensures that it is resistant to tampering, censorship, and fraud, making it an ideal platform for secure and transparent transactions.
- Blockchain technology is most commonly associated with cryptocurrencies like Bitcoin, but it has many other potential applications, such as supply chain management, voting systems, and digital identity verification.

- The decentralized nature of the blockchain ledger ensures that it is resistant to tampering, censorship, and fraud, making it an ideal platform for secure and transparent transactions.
- Blockchain technology is most commonly associated with cryptocurrencies like Bitcoin, but it has many other potential applications, such as supply chain management, voting systems, and digital identity verification.

Parameter	Blockchain Ledger	Distributed Ledger
Consensus mechanism	A consensus method, such as proof of work or proof of stake, is used by blockchain ledgers to validate transactions and add new blocks to the chain.	Distributed ledgers, on the other hand, employ a variety of consensus algorithms, including Paxos, Raft, and Byzantine Fault Tolerance.
Scalability	Due to the consensus method and the requirement that every node validates every transaction, blockchain ledgers have come under fire for their poor scalability.	On the other side, distributed ledgers can be more scalable since they leverage an efficient network of nodes to validate transactions.
Applications	Bitcoin and other cryptocurrencies as well as other financial applications frequently employ blockchain ledgers.	Several sectors, including supply chain management, healthcare, and government, use distributed ledgers.
Permission	Blockchain ledgers come in both public and private varieties, with private blockchains only accessible to a few users.	Distributed ledgers can either be public or private, however, they are typically utilized in business contexts and require access rights.
Structure	Blockchain is a distributed ledger that uses blocks of data connected in a chain using encryption, making it difficult to change.	On the other hand, distributed ledgers use a network of nodes rather than a chain of blocks to jointly maintain a ledger.

BLOCK IN A BLOCKCHAIN

Blocks are files stored by a blockchain, where transaction data are permanently recorded. A block records some or all of the most recent transactions not yet validated by the network.

Once the data are validated, the block is closed. Then, a new block is created for new transactions to be entered into and validated.

A block is thus a permanent store of records that, once written, cannot be altered or removed without changing all preceding or following blocks.

How a Block (Blockchain Block) Works

A blockchain network witnesses a great deal of transaction activity. The transactions made during a period are recorded into a file called a block, which is the basis of the blockchain network.

A block stores information. There are many pieces of information included within a block, but it doesn't occupy a large amount of storage space. Blocks generally include these elements, but it might vary between different types. For example, Bitcoin block structures include:

- **Blocksize**: Sets the size limit on the block so that only a specific amount of information can be written in it
- **Block header**: Contains information about the block
- **Transaction counter**: A field that lists how many transactions are stored in the block
- **Transactions**: A list of all of the transactions within a block

The transaction element is the largest because it contains the most information. It is followed in storage size by the block header, which includes these sub-elements:

- **Version**: The cryptocurrency version being used, also called the magic number
- **Previous block hash**: Contains a hash (encrypted number) of the previous block's header
- **Merkle root**: Hash of transactions in the Merkle tree of the current block
- **Timestamp**: A timestamp to place the block in the blockchain
- **Difficulty Target**: The difficulty rating of the target hash, signifying the difficulty in generating a hash that is equal to or less than the target.
- **Nonce**: A number the miner increases incrementally when hashing

One 32-bit (four bytes) number in the header is called a nonce (number used once). The nonce is increased with every guess by a value of one. In Bitcoin's case, the nonce can only go up to a value of about 4 billion before it rolls over to zero again (which takes less than one second). Once this happens, another value called the extra nonce is changed, and the nonce starts over again at zero.

Once the right combination of block data, nonce, and extra nonce are found, the miner wins the competition. The network closes that block, generates a new one with a header, and the process repeats.

Different mechanisms are used to reach a consensus; the most popular for cryptocurrency is proof-of-work (PoW), with proof-of-stake (PoS) becoming more so because of the reduced energy consumption compared to PoW.

The XRP Ledger uses a mechanism similar to proof-of-history (PoH), where a deterministic transactional ordering algorithm prevents double-spending.

Mining's Relationship to Blocks

Mining is the term used for solving the cryptographic puzzle, which acts as the first verification for transactions and proof that work was done to do the verification.

Cryptocurrency mining is commonly thought to be solving a complex mathematical problem; mining is actually sending block data through a hashing algorithm (which uses complicated math functions) to generate a hexadecimal number.

For example, Bitcoin uses the hashing algorithm SHA-256. For a miner to generate the "winning" number, the mining program sends the block header, nonce, and the extra nonce

through SHA-256. The two nonces are adjusted incrementally until a certain value is reached. This value is another hexadecimal number that requires the network to spend about 10 minutes generating hashes.

Other Block and Blockchain Uses

Because most blockchain definitions refer to Bitcoin because it was the first cryptocurrency to use one, many people associate blocks and blockchains with Bitcoin. However, other cryptocurrencies use blocks and blockchains as well. It's important to note that Ethereum's network also uses blocks and blockchain.

However, Ethereum and its blockchain were designed for multiple uses that extend to much more than cryptocurrency. For example, non-fungible tokens, smart contracts, decentralized finance applications, and more have been developed using Ethereum.

Block and Blockchain Developments

Cryptocurrency introduced blockchains to the mainstream, but these distributed ledgers have been under development for many years. Blocks are files that store information, so much more can be done with them than issuing a token that can be used as payment or to invest in.

Blocks and blockchains create unchangeable records. Public blockchains can be widely distributed and nearly impossible to alter, while private blockchains can be secured using accounts and permissions. Permissioned public blockchains are a combination of public and private blockchains. These three types set the stage for what some believe to be the next iteration of the way information and content are stored, accessed, controlled, and secured.

The backbone of the modern world is connectivity and the internet. Information and content are hosted and stored on centralized servers controlled by businesses and governments. This allows them to censor what you are exposed to, monetize (or demonetize) any content you create, track your spending and viewing habits, set restrictions on what you can and can't do, and much more.

Blockchain developers are developing systems designed to replace the structure of the web. This emerging structure is commonly called Web 3. Web 3 is a concept that combines tokens, wallets, cryptocurrencies, and blockchains to overhaul the web's infrastructure and transition ownership of digital content and assets back to creators.

Blockchain Impacts for Individuals

One of the most significant impacts of blockchains on individuals is access to financial services. Even in developed countries, some people do not have access to personal or business loans, credit, or other financial instruments that could help them achieve financial independence and stability.

Another important expectation for blockchain and Web 3 development is that they will allow you to control how your personal information is used. Information such as your age, income, what you watch, and what you're shopping for is valuable to businesses. Businesses pay other companies for your information so that they can market specifically to you and others who share your interests.

If you controlled your information using tokens, you could sell it to a business and make money from it rather than someone else using your information to earn revenue. Even better, imagine placing only the information you want businesses to have in a smart contract. If an entity wanted it, they would have to pay you for it using cryptocurrency. The blockchain would release it, you'd be paid, and the smart contract could keep the business from reselling it to another company without your permission.

Blockchain Impacts for Businesses

Businesses in all sectors are exploring ways to use blockchains. Insurance companies can use it to facilitate insurance claims and fraud investigations, and the legal industry can use it to reduce time and costs.

Intellectual property rights have always been an issue, with thefts, piracy, and fraud costing billions of dollars. Creators, media, and entertainment providers can tokenize and track content to protect their interests.

Financial service providers can have access to more individuals. Businesses can use blockchain to accurately and automatically record data, trusting that it cannot be altered by anyone. Data in these cases would be more transparent and actionable, allowing businesses to make more informed and strategic decisions.

Many international businesses use blockchain solutions to make faster and cheaper cross-border payments. Using blockchain and cryptocurrency can reduce overall costs for enterprises, which can translate into lower costs for consumers.

What Exactly Does a Blockchain Do?

A blockchain is a distributed record of information that uses cryptography and scale to secure it. Blockchains can be used for any purpose that requires recording and securing data.

How Do I Identify a Block in a Blockchain?

In some blockchains, blocks have a number called block height. This is the sequential number of the block on a chain, such as Block 1, Block 2, and so on. Others might use a unique number called a block header, ledger header, or other hexadecimal number.

What Is Blockchain For Beginners?

A blockchain is a collection of files (blocks) chained together by encrypting data in one block and inserting the result into the next file. Each block is encrypted this way, which chains them together and makes them immutable after a certain number of confirmations.

The Bottom Line

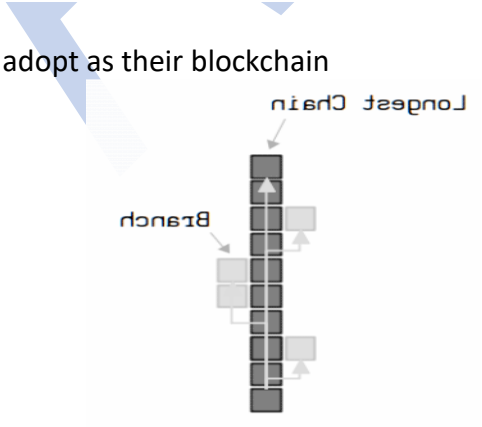
Blocks are files containing data and information about transactions and previous blocks. They are encrypted, proposed, and stored using various methods, but they are the basis for every blockchain.

Blocks and blockchains are most commonly associated with cryptocurrency, but their eventual, hypothetical, and emerging uses in the world economy are what make them more interesting, especially for investors.

TRANSACTIONS- THE CHAIN AND THE LONGEST CHAIN

Longest Chain

The chain of blocks nodes adopt as their blockchain

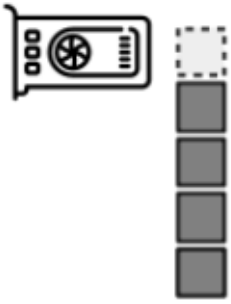


Current Longest Chain:

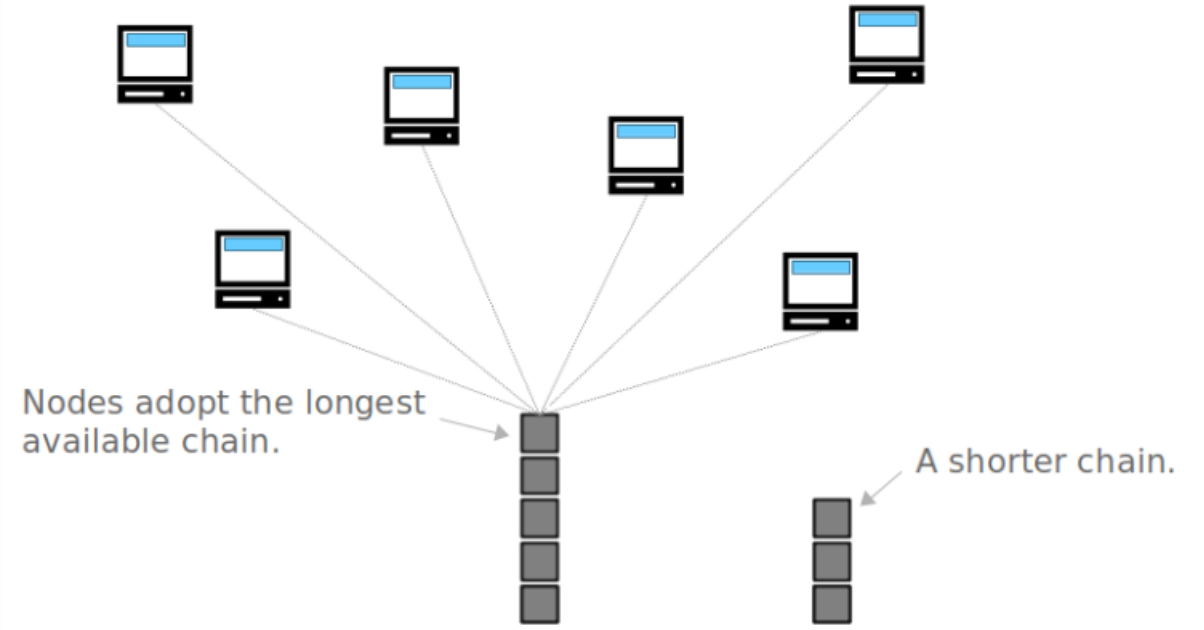
- **Height:** 845241

- **Chainwork:** 0x0000000000000000000000000000000000000000007b75a880303be329e48ddf3c

The *longest chain* is what Bitcoin nodes accept as the **valid version** of the blockchain.
The longest chain rule **allows every node on the network to agree** on what the blockchain looks like, and therefore agree on the same transaction history.
What is the longest chain?
The longest chain is the chain of blocks that took the **most effort to build**.
In short, to add a new block to the blockchain you need to use processing power, which means that every block on the blockchain required a certain amount of *energy* to get there.



Blocks are mined using processing power.
Therefore, a blockchain with *more blocks* in it will have taken *more energy* to build than a chain with fewer blocks in it, and as a rule nodes will always adopt this chain over a "shorter" one.



As a result, nodes will always adopt the chain that took the most energy to build, and this is what we mean when we refer to the "longest chain".

PERMISSIONED MODEL OF BLOCKCHAIN

Permissioned blockchains are blockchains that are closed (i.e., not publicly accessible) or have an access control layer. This additional layer of security means that the blockchain can only be accessed by users with permissions. Permissioned users are only able to perform blockchain operations within the strict confines of roles assigned to them by the ledger administrators and

require that they authenticate themselves through certificates or digital identifier methods. In addition, the roles would dictate what information a user would be able to access.

Aspects of a permissioned blockchain

Decisions are authorized by a private group

Decisions are made by the owners of the network through a central, pre-defined level.

Security

Permissioned blockchains provide the operating organization granular control over permissions, data access, and the scope of user roles.

Decentralization isn't fixed

Permissioned blockchains can either be fully centralized or partially decentralized. Its members typically decide on the network's level of decentralization and the mechanisms for consensus.

Transparency is not required

Unlike permissionless blockchains, permissioned blockchains do not need to be transparent. Transparency is optional, as most permissioned blockchain networks are specifically intended to not be transparent for security purposes. Levels of transparency usually depend on the goals of the organization running the blockchain network.

In the meantime, the ledger maintains a record of every transaction and the identities of the participating parties.

Lack of anonymity

Access to the identify of every transactional participant can be crucial information for private entities concerned with accountability and a provable chain of custody. Every change is tracked to a specific user, so network administrators can have instantaneous access to has made a change to the system and when.

Consensus mechanisms

Because of the structure of permissioned blockchains, they don't use the same types of consensus protocols as permissionless ones. Most commonly, organizations that deploy permissioned blockchains use one (or more) of the following three protocols: Practical Byzantine Fault Tolerance (PBFT), federated, or round-robin consensus.

PBFT – PBFT is an improved version of the original BFT protocol where all voting nodes much reach a consensus, but one or more parties are considered unreliable. In this model, a network's safety and stability are guaranteed so long as the required minimum percentage of nodes are behaving honestly and properly.

Federated (or Federated Byzantine Consensus) - In a federated consensus, there's a set of transaction validators trusted by each node in the blockchain that receives and sorts the transactions. Once a minimum number of these validators agree, a consensus is reached.

Round-robin - In a round-robin consensus, nodes are selected pseudo-randomly to create blocks. Once chosen, a node must pass through a cooling-off period before it can reenter the pool and be available again for consensus participation.

Advantages of permissioned blockchain

One of the most significant advantages of permissioned blockchains is the high level of privacy and security they can provide. Without a verified set of credentials and access, no user can access or alter transaction information without permission.

Another advantage is flexibility when it comes to decentralization. It can be incremental or fully centralized, giving businesses more freedom to participate without having to worry about the risks associated with a highly centralized network.

Permissioned blockchains are also highly customizable and can accommodate configurations and integrations based on an organization's needs. And with knowledge of every user and their actions on the network, a verifiable chain of custody can be established for every transaction.
Lastly, these types of blockchains are both scalable and highly performant due to the limited number of nodes needed to manage transaction verifications.
Disadvantages of permissioned blockchain
While lack of transparency can be a potential point of concern for permissioned blockchains, the issue is usually mitigated by the implicit trust placed in the governing authority. In a business context, consensus mechanisms and the smart contracts that moderate transactions on the network are agreed upon by the participating parties and maintained in secure, isolated containers. With this additional layer of computational security and measure of implicit trust, a properly provisioned permissioned blockchain can offset the security risk posed by bad actors.
Why permissioned blockchains are ideal for business applications
Many enterprise use cases require performance characteristics that permissionless blockchain technologies are presently unable to deliver because of limitations due to inefficiency and scalability. Additionally, in instances where permissioned blockchains are replacing existing secure, centralized networks, the identity of the participants is an essential requirement, such as in the case of financial transactions where Know-Your-Customer (KYC), Anti-Money Laundering (AML), and supply-chain provenance regulations must be followed.
In general, then, for a blockchain network to be ready for enterprise use, it should possess the following requirements:

- Participants must be identified/identifiable
- Networks need to be permissioned
- High transaction throughput performance
- Low latency of transaction confirmation
- Privacy and confidentiality of transactions and data pertaining to business transactions

Business value
Let's quickly review and see how permissioned blockchains stack up against these requirements. In terms of added value, permissioned blockchains:

- Increase business velocity by accelerating transactions, enabling new business models and revenue streams
- Automate multi-party business processes
- Reduce the cost and risk of using intermediaries
- Reduce the cost of fraud and regulatory compliance
- Improve data quality and timeliness by avoiding offline reconciliation and manual exception handling
- Increase auditability and trust; reduce audit costs

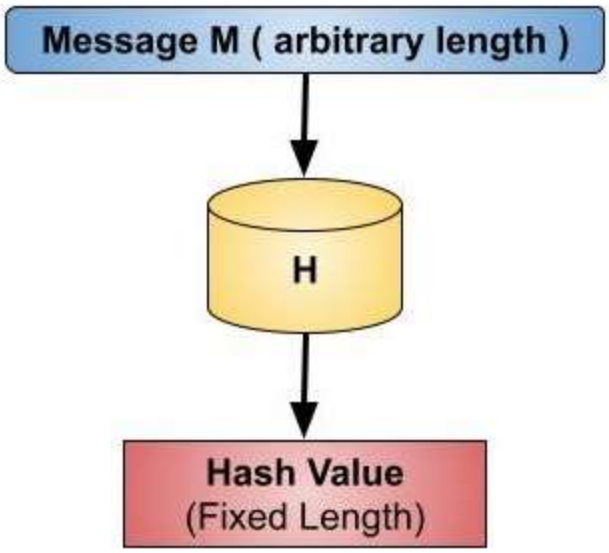
Comparing the two, permissioned blockchains are well positioned to achieve all the stated business requirements.
Use case examples
So, how are permissioned blockchains being used by businesses? While still an emerging business model, they have already found a wide variety of applications. Permissioned blockchains have been used to manage supply chains, create contracts, handle claims, verify payment between parties, and administer user identity.

CRYPTOGRAPHIC HASH FUNCTION

hash function in cryptography is like a mathematical function that takes various inputs, like messages or data, and transforms them into fixed-length strings of characters. Means the input to the hash function is of any length but output is always of fixed length. This is like compressing a large balloon into a compact ball.

The importance of this process lies in its generation of a unique "fingerprint" for each input. Any minor alteration in the input results in a substantially different fingerprint, a quality known as "collision resistance."

Hash functions play a crucial role in various security applications, including password storage (hash values instead of passwords), digital signatures, and data integrity checks. Hash values, or message digests, are values that a hash function returns. The hash function is shown in the image below –



Key Points of Hash Functions

- Hash functions are mathematical operations that "map" or change a given collection of data into a fixed-length bit string that is referred to as the "hash value."
- Hash functions have a variety of complexity and difficulty levels and are used in cryptography.
- Cryptocurrency, password security, and communication security all use hash functions.

Operation of Cryptographic Hash Functions

In computing systems, hash functions are frequently used data structures for tasks like information authentication and message integrity checks. They are not easily decipherable, but because they can be solved in polynomial time, they are regarded as cryptographically "weak".

Typical hash functions have been improved with security characteristics by cryptographic hash functions, which make it more challenging to decipher message contents or recipient and sender information.

Specifically, cryptographic hash functions display the following three characteristics –

- The hash function are called as "collision-free." As a result, no two input hashes should be equal to the same output hash.
- They are hidden. A hash function's output should make it difficult to figure out the input value from it.

- They should to be friendly to puzzles. The selection of an input that generates a predetermined result needs to be difficult. As such, the input needs to be taken from as wide as possible.

Properties of hash functions

To be a reliable cryptographic tool, the hash function should have the following properties −

Pre-Image Resistance

- According to this feature, reversing a hash function should be computationally difficult.
- In other words, if a hash function h generates a hash value z, it should be difficult to identify an input value x that hashes to z.
- This feature defends against an attacker attempting to locate the input with just the hash value.

Second Pre-Image Resistance

- This property says that given an input and its hash, it should be difficult to find another input with the same hash.
- In other words, it should be challenging to find another input value y such that h(y) equals h(x) if a hash function h for an input x returns the hash value h(x).
- This feature of the hash function protects against an attacker who wants to replace a new value for the original input value and hash, but only holds the input value and its hash.

Collision Resistance

- This feature says that it should be difficult to identify two different inputs of any length that produce the same hash. This characteristic is also known as a collision-free hash function.
- In other words, for a hash function h, it is difficult to identify two distinct inputs x and y such that h(x)=h(y).
- A hash function cannot be free of collisions because it is a compression function with a set hash length. The collision-free condition simply indicates that these collisions should be difficult to locate.
- This characteristic makes it very hard for an attacker to identify two input values that have the same hash.
- Furthermore, a hash function is second pre-image resistant if it is collision-resistant.

Efficiency of Operation

- Computation of h(x) for any hash function h given input x can be an easy process.
- Hash functions are computationally considerably faster than symmetric encryption.

Fixed Output Size

Hashing generates an output of a specific length, regardless of the input size, and helps to make an output of the same size from different input sizes.

Deterministic

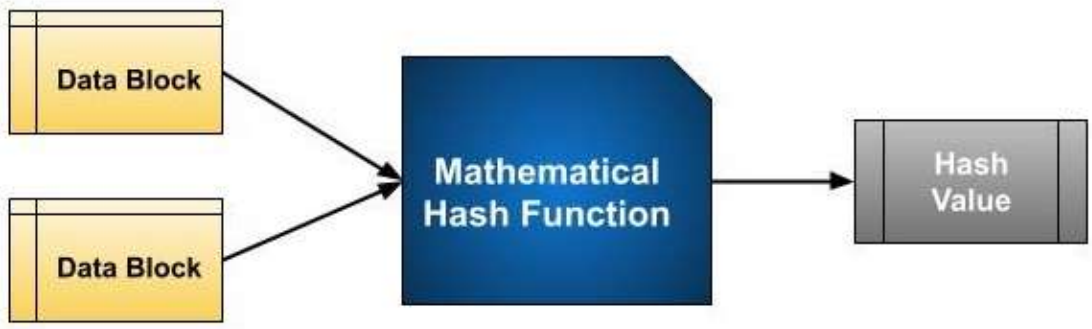
For a given input, the hash function consistently produces the same output, like a recipe that always yields the same dish when followed precisely.

Fast Computation

Hashing operations occur rapidly, even for large amounts of data sets.

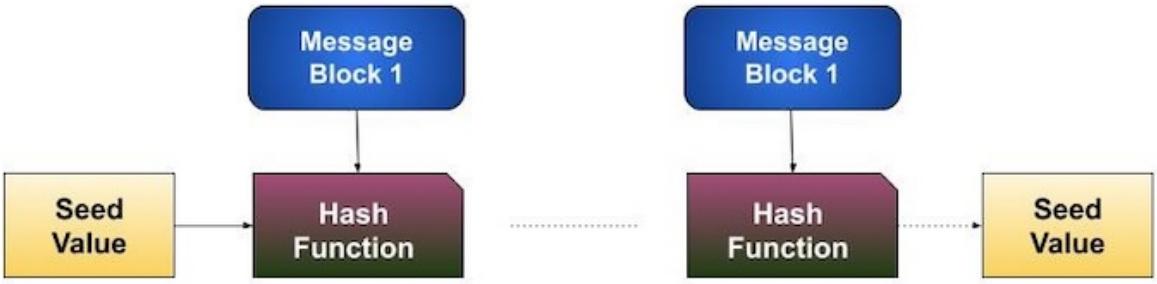
Design of Hashing Algorithms

Hashing essentially involves a mathematical function that takes two data blocks of fixed size and converts them into a hash code. The function is a key part of the hashing algorithm. The length of these data blocks differ according to the algorithm used. Usually, they range from 128 bits to 512 bits. Below is an example of a hash function −



Hashing algorithms use a sequence of rounds, similar to a block cipher, to process a message. In each round, a fixed-size input is used, which usually combines the current message block and the result from the previous round.

This process continues for multiple rounds until the entire message is hashed. A visual representation of this process is provided in the illustration below.



Due to the interconnected nature of hashing, where the output of one operation affects the input of the next, even a minor change (a single bit difference) in the original message can drastically alter the final hash value.

This phenomenon is known as the avalanche effect. Additionally, it's crucial to distinguish between a hash function and a hashing algorithm. The hash function itself takes two fixed-length binary blocks of data and generates a hash code.

A hashing algorithm, on the other hand, establishes how the message is divided into blocks and how the outcomes of multiple hash operations are combined.

Popular Hash Functions

Hash functions play an important role in computing, providing versatile capabilities like: Quick retrieval of data, Secure protection of information (cryptography), Ensuring data remains unaltered (integrity verification). Some commonly used hash functions are −

Message Digest (MD)

For a number of years, MD5 was the most popular and often used hash function.

- The hash functions MD2, MD4, MD5, and MD6 are members of the MD family. It was adopted as the RFC 1321, Internet Standard. It is a 128-bit hash function.
- In the software industry, MD5 digests are frequently used to ensure the integrity of transferred files. To enable users to compare the checksum of the downloaded file with the pre-computed MD5 checksum, file servers frequently provide this feature.

- In 2004, collisions were found in MD5. It was claimed that an analytical attack using a computer cluster was successful in under one hour. Since MD5 was compromised by this collision attack, using it is no longer recommended.

Secure Hash Function (SHA)

The four SHA algorithms which make up the SHA family are SHA-0, SHA-1, SHA-2, and SHA-3. Despite coming from the same family, the structure of it differs.

- The National Institute of Standards and Technology (NIST) released the first iteration of the 160-bit hash algorithm, known as SHA-0, in 1993. It did not gain much popularity and had few drawbacks. SHA-1 was created later in 1995 to address perceived flaws in SHA-0.
- SHA-1 is the most widely used of the existing SHA hash functions. It is used in most of the applications and protocols including Secure Socket Layer (SSL) security.
- In 2005, a technique was discovered for SHA-1 collision detection that can be used in a realistic time frame. So it is doubtful on SHA-1's long-term usability.
- SHA-224, SHA-256, SHA-384, and SHA-512 are the other four SHA variants in the SHA-2 family, which vary based on the number of bits in their hash value. The SHA-2 hash function has not yet been the target of any effective attacks
- Though SHA-2 is a strong hash function. Though significantly different, its basic design still follows the design of SHA-1. NIST thus demanded the creation of new competitive hash function designs.
- The Keccak algorithm was selected by the NIST in October 2012 to replace the SHA-3 standard. Keccak has several advantages, including effective operation and strong attack resistance.

CityHash

CityHash is another non-cryptographic hash function that is designed for fast hashing of large amounts of data. It is optimized for modern processors and offers good performance on both 32-bit and 64-bit architectures.

BLAKE2

BLAKE2 is a fast and secure hash function that improves upon SHA-3. It is widely used in applications like cryptocurrency mining that need fast hashing. There are two types of BLAKE2 –

- **BLAKE2b** − Best for 64-bit computers, it produces hash values up to 512 bits long.
- **BLAKE2s** − Best for smaller computers (8-32 bits), it produces hash values up to 256 bits long.

CRC (Cyclic Redundancy Check)

CRC (Cyclic Redundancy Check) is a technique used to detect errors in data transfer. It involves adding a special value called a checksum to the end of a message. This checksum is calculated based on the message's content and is included during transmission.

When the data is received, the recipient recalculates the checksum using the same method. If the new checksum matches the original one, it's likely that the message was transmitted without errors. While CRC is effective for error detection, it's not a security measure. It is primarily used to ensure the integrity of data during transmission, not to protect it from unauthorized access or modification.

MurmurHash

MurmurHash is a speedy and effective hash function that is not meant for security. It is great for things like hash tables but not for tasks that need protection against collisions (situations where different inputs produce the same hash).

Standard Length

Hashing involves converting a data set of any size into a shorter, fixed-length output using a mathematical formula.

Table I: Different Hash Functions

In table I, the message "CFI" is converted into hash values using three algorithms: MD5, SHA-1, and SHA-256. Each algorithm produces a unique output hash with a fixed length. MD5 generates a hash with 32 hexadecimal characters, SHA-1 with 40 characters, and SHA-256 with 64 characters.

Input Message	Hash Function	Output (Hash Value)
CFI	MD5 (128-bit, 16-byte) 32 characters	3A10 0B15 B943 0B17 11F2 E38F 0593 9A9A
CFI	SHA-1 (160-bit, 20-byte) 40 characters	569D C9F0 7B48 7F58 9241 AD4C 5C28 7DA0 A448 8D08
CFI	SHA-256 (256-bit, 32-byte) 64 characters	F3ED 0867 48FF 3641 3091 0BB6 6293 7080 2958 B5A2 52AF F364 1FC5 07FD E80D 9929

Table II: Using the Same Hash Function (SHA-1) with different Inputs

Besides the data (input) used, a hash function consistently generates a hash value with a fixed number of characters. As shown in Table II, different messages inputted into the same hash function (SHA-1 in this case) consistently produce output values of 40 hexadecimal characters in length.

Input Message	Hash Function	Output (Hash Value)
CFI	SHA-1	569D C9F0 7B48 7F58 9241 AD4C 5C28 7DA0 A448 8D08
Corporate FI	SHA-1	82C0 5EDC 608F AA08 8EE0 BDD8 8E22 3B38 CA38 82CC
CF Input	SHA-1	2013 85FC EEE4 F73D 07F0 4F2A A4CB BOE9 12BF BBB8
CFI 1	SHA-1	C501 23CE 8BB2 A42D 5BB4 4DA7 3FC2 3B3D 62F5 14A5

Applications of Hash Functions

Based on its cryptographic characteristics, the hash function has two direct uses.

Password Storage

Hash functions provide protection to password storage. Instead of storing passwords in clear, mostly all login processes store the hash values of passwords in the file.

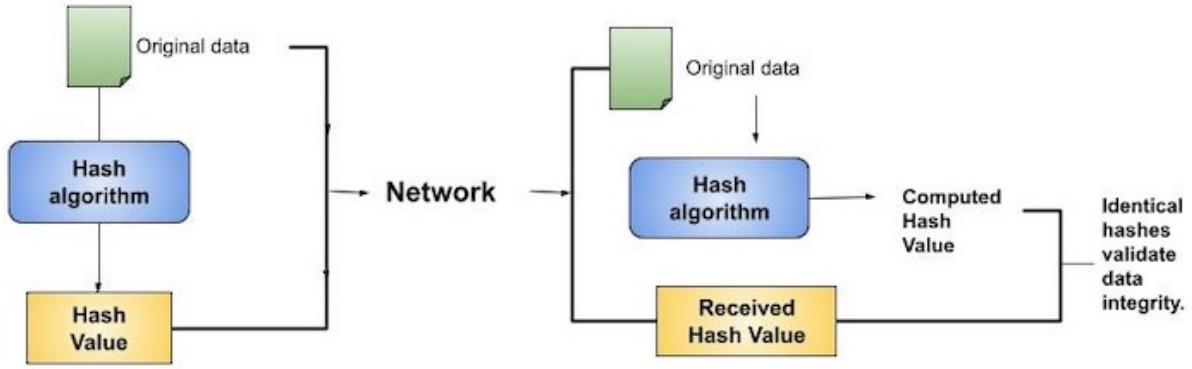
The Password file is a table of pairs in the format (user id, h(P)).

Even if an attacker has access to the password, all they can see is the hashes of the passwords. Because the hash function contains the pre-image resistance feature, he cannot use it to log in or get the password from it.

Data Integrity Check

Data integrity checks, commonly using hash functions, provide assurances about the accuracy of data files by creating checksums. This method allows users to detect any alterations made to the original file.

However, it does not guarantee the authenticity of the file. An attacker could potentially modify the entire file and generate a new hash, sending it to the receiver. This integrity check is only effective if the user trusts the file's original source.



Hashing vs Encryption

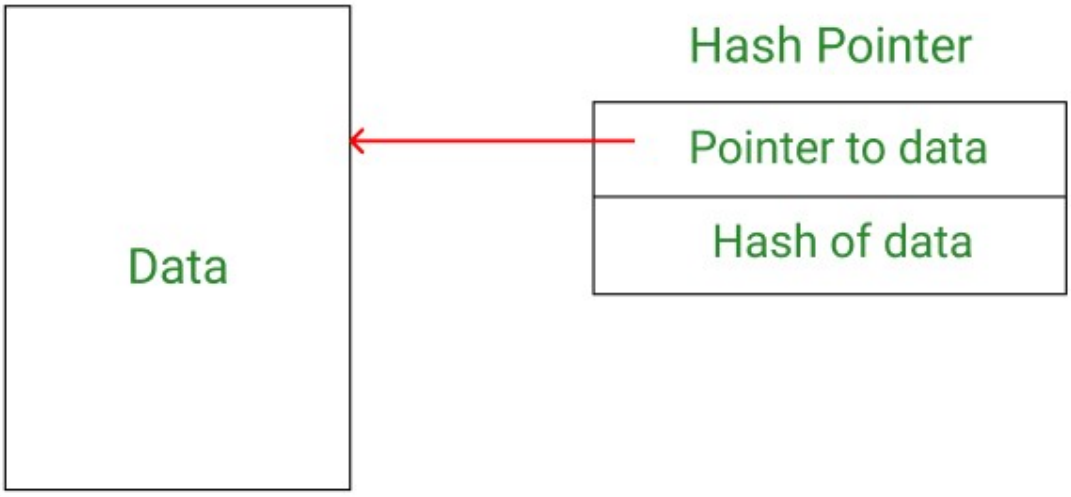
Encryption transforms data into a disguised form, requiring a cipher (key) to decipher and read it. Encryption and decryption are reversible processes enabled by the cipher. Encryption is used with the goal of later deciphering the data.

Hashing transforms data of any size into a fixed-length output. Unlike encryption, hashing is typically a one-way function. The high computational effort needed to reverse a hash makes it difficult to retrieve the original data from the hashed output.

Data is protected during transmission by encryption, which stops unwanted access. By comparing the data to a distinct fingerprint (hash) created from the original data, hashing ensures the integrity of the data. Encryption keeps data confidential, while hashing ensures authenticity by detecting any modifications.

HASH POINTER AND MERKLE TREE

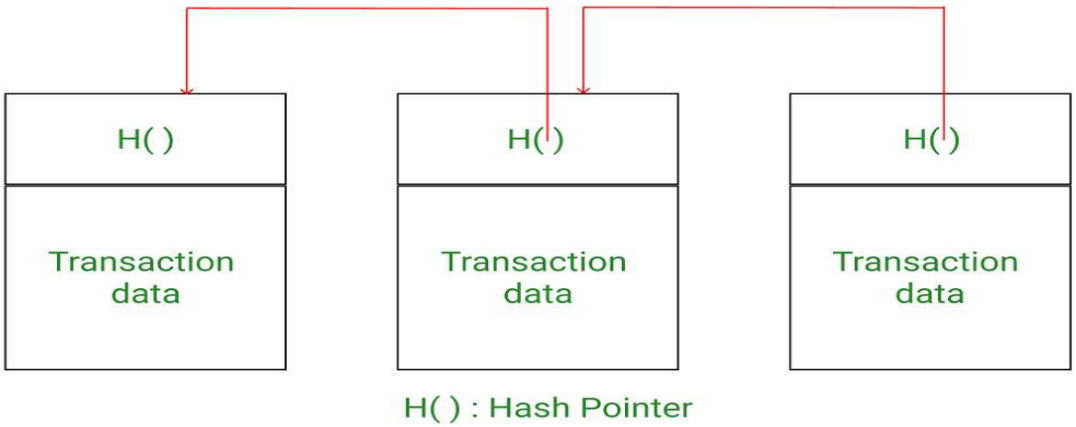
A regular pointer stores the memory address of data. With this pointer, the data can be accessed easily. On the other hand, a hash pointer is a pointer to where data is stored and with the pointer, the cryptographic hash of the data is also stored. So a hash pointer points to the data and also allows us to verify the data. A hash pointer can be used to build all kinds of data structures such as **blockchain and Merkle tree.**



Blockchain Structure

The blockchain is a proficient combination of two hash-based data structures-

1. **Linked list:** This is the structure of the blockchain itself, which is a linked list of hash pointers. A regular linked list consists of nodes. Each node has 2 parts- data and pointer. The pointer points to the next node. In the blockchain, simply replace the regular pointer with a hash pointer.
2. **Merkle tree:** A Merkle tree is a binary tree formed by hash pointers, and named after its creator, Ralph Merkle.



Blockchain as linked list with hash pointers

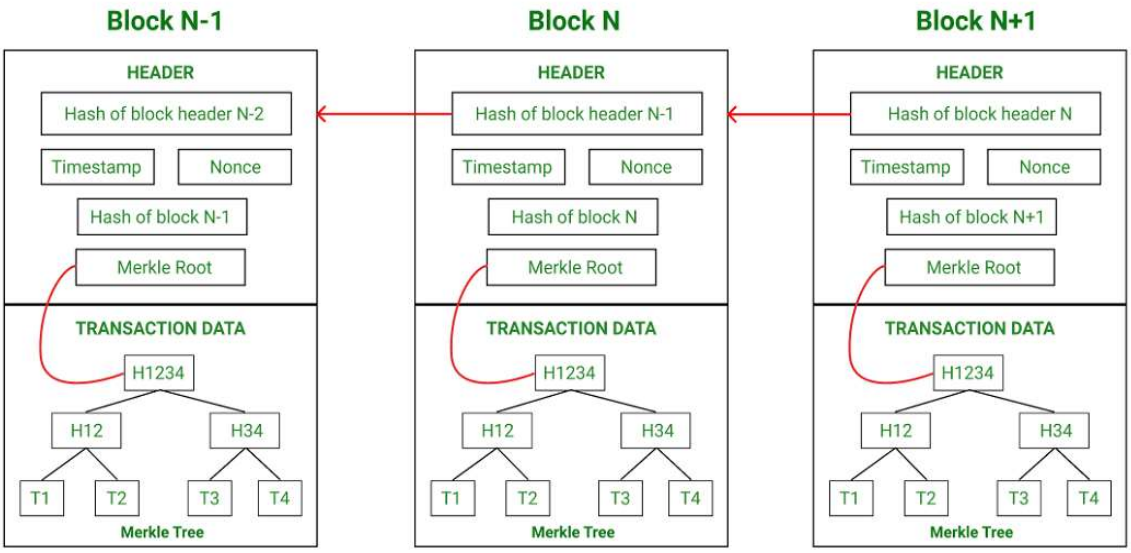
Block Structure

1. Block header: The header data contains metadata of the block, i.e information about the block itself. The contents of the block header include-

- Hash of the previous block header.
- Hash of the current block.
- Timestamp.
- Cryptographic nonce.
- Merkle root.

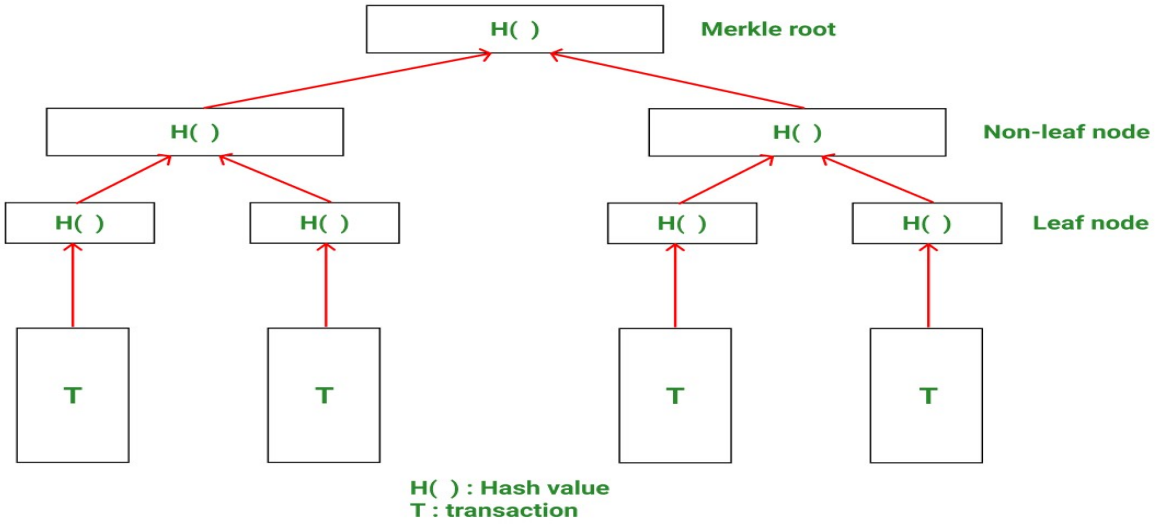
2. Merkle tree: A Merkle tree is a binary tree formed by hash pointers, and named after its creator, Ralph Merkle.

- As mentioned earlier, each block is supposed to hold a certain number of transactions. Now the question arises, how to store these transactions within a block? One approach can be to form a hash pointer-based linked list of transactions and store this complete linked list in a block. However, when we put this approach into perspective, it does not seem practical to store a huge list of hundreds of transactions. What if there is a need to find whether a particular transaction belongs to a block? Then we will have to traverse the blocks one by one and within each block traverse the linked list of transactions.
- This is a huge overhead and can reduce the efficiency of the blockchain. Now, this is where the Merkle tree comes into the picture. Merkle tree is a per-block tree of all the transactions that are included in the block. It allows us to have a hash/digest of all transactions and provides proof of membership in a time-efficient manner.
- So to recap, the blockchain is a hash-based linked list of blocks, where each block consists of a header and transactions. The transactions are arranged in a tree-like fashion, known as the Merkle tree.



Each block comprises of block header + Merkle tree

Merkle Tree Structure



Structure of Merkle tree

1. A blockchain can potentially have thousands of blocks with thousands of transactions in each block. Therefore, memory space and computing power are two main challenges.

2. It would be optimal to use as little data as possible for verifying transactions, which can reduce CPU processing and provide better security, and this is exactly what Merkle trees offer.

3. In a Merkle tree, transactions are grouped into pairs. The hash is computed for each pair and this is stored in the parent node. Now the parent nodes are grouped into pairs and their hash is stored one level up in the tree. This continues till the root of the tree. The different types of nodes in a Merkle tree are:

- **Root node:** The root of the Merkle tree is known as the Merkle root and this Merkle root is stored in the header of the block.
- **Leaf node:** The leaf nodes contain the hash values of transaction data. Each transaction in the block has its data hashed and then this hash value (also known as transaction ID) is stored in leaf nodes.

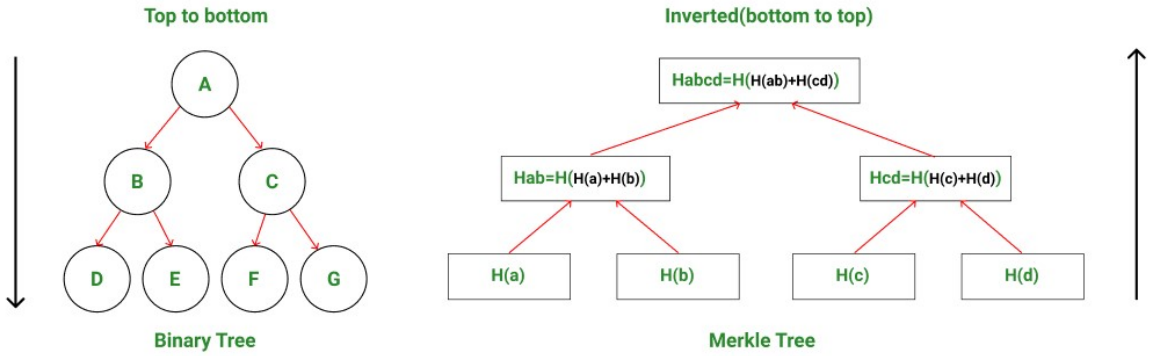
- **Non-leaf node:** The non-leaf nodes contain the hash value of their respective children. These are also called intermediate nodes because they contain the intermediate hash values and the hash process continues till the root of the tree.

4. Bitcoin uses the SHA-256 hash function to hash transaction data continuously till the Merkle root is obtained.

5. Further, a Merkle tree is **binary in nature**. This means that the **number of leaf nodes needs to be even** for the Merkle tree to be constructed properly. In case there is an odd number of leaf nodes, the tree duplicates the last hash and makes the number of leaf nodes even.

How Do Merkle Trees Work?

- A Merkle tree is constructed from the leaf nodes level all the way up to the Merkle root level by grouping nodes in pairs and calculating the hash of each pair of nodes in that particular level. This hash value is propagated to the next level. This is a **bottom-to-up** type of construction where the hash values are flowing from down to up direction.
- Hence, by comparing the Merkle tree structure to a regular binary tree data structure, one can observe that Merkle trees are actually **inverted down**.



Binary tree direction vs Merkle tree direction

Example: Consider a block having 4 transactions- T1, T2, T3, T4. These four transactions have to be stored in the Merkle tree and this is done by the following steps-

Step 1: The hash of each transaction is computed.

H1 = Hash(T1).

Step 2: The hashes computed are stored in leaf nodes of the Merkle tree.

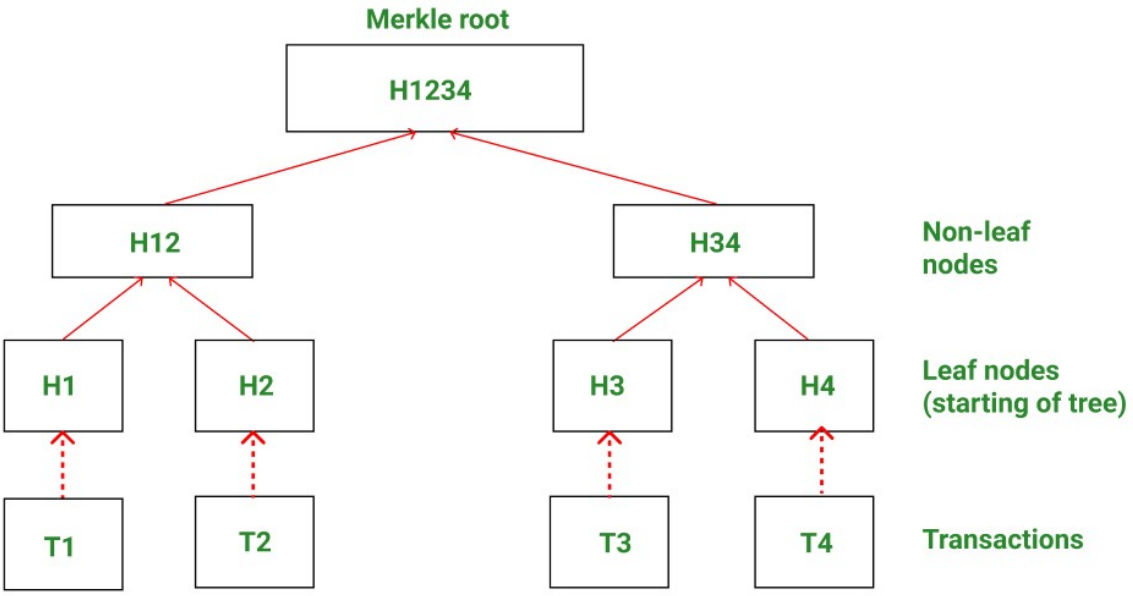
Step 3: Now non-leaf nodes will be formed. In order to form these nodes, leaf nodes will be paired together from left to right, and the hash of these pairs will be calculated. Firstly hash of H1 and H2 will be computed to form H12. Similarly, H34 is computed. Values H12 and H34 are parent nodes of H1, H2, and H3, H4 respectively. These are non-leaf nodes.

H12 = Hash(H1 + H2)

H34 = Hash(H3 + H4)

Step 4: Finally H1234 is computed by pairing H12 and H34. H1234 is the only hash remaining. This means we have reached the root node and therefore H1234 is the Merkle root.

H1234 = Hash(H12 + H34)



Merkle tree works by hashing child nodes again and again till only one hash remains.

Key Points:

- In order to check whether the transaction has tampered with the tree, there is only a need to remember the root of the tree.
- One can access the transactions by traversing through the hash pointers and if any content has been changed in the transaction, this will reflect on the hash stored in the parent node, which in turn would affect the hash in the upper-level node and so on until the root is reached.
- Hence the root of the Merkle tree has also changed. So Merkle root which is stored in the block header makes transactions tamper-proof and validates the integrity of data.
- With the help of the Merkle root, the Merkle tree helps in eliminating duplicate or false transactions in a block.
- It generates a digital fingerprint of all transactions in a block and the Merkle root in the header is further protected by the hash of the block header stored in the next block.

Why Merkle Trees are Important For Blockchain?

- In a centralized network, data can be accessed from one single copy. This means that nodes do not have to take the responsibility of storing their own copies of data and data can be retrieved quickly.
- However, the situation is not so simple in a distributed system.
- Let us consider a scenario where blockchain does not have Merkle trees. In this case, every node in the network will have to keep a record of every single transaction that has occurred because there is no central copy of the information.
- This means that a huge amount of information will have to be stored on every node and every node will have its own copy of the ledger. If a node wants to validate a past transaction, requests will have to be sent to all nodes, requesting their copy of the ledger. Then the user will have to compare its own copy with the copies obtained from several nodes.
- Any mismatch could compromise the security of the blockchain. Further on, such verification requests will require huge amounts of data to be sent over the

network, and the computer performing this verification will need a lot of processing power for comparing different versions of ledgers.

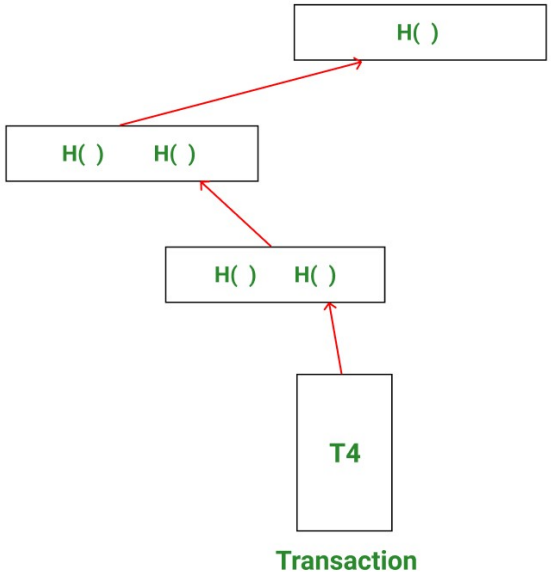
- Without the Merkle tree, the **data itself has to be transferred all over the network** for verification.
- Merkle trees allow comparison and verification of transactions with **viable computational power and bandwidth**. Only a small amount of information needs to be sent, hence compensating for the huge volumes of ledger data that had to be exchanged previously.

Merkle trees use a one-way hash function extensively and this hashing separates the proof of data from data itself

Proof of Membership

A very interesting feature of the Merkle tree is that it provides **proof of membership**.

Example: A miner wants to prove that a particular transaction belongs to a Merkle tree Now the miner needs to present this transaction and all the nodes which lie on the path between the transaction and the root. The rest of the tree can be ignored because the hashes stored in the intermediate nodes are enough to verify the hashes all the way up to the root.



Proof of membership: verifying the presence of transactions in blocks using the Merkle tree.

If there are n nodes in the tree then only log(n) nodes need to be examined. Hence even if there are a large number of nodes in the Merkle tree, proof of membership can be computed in a relatively short time.

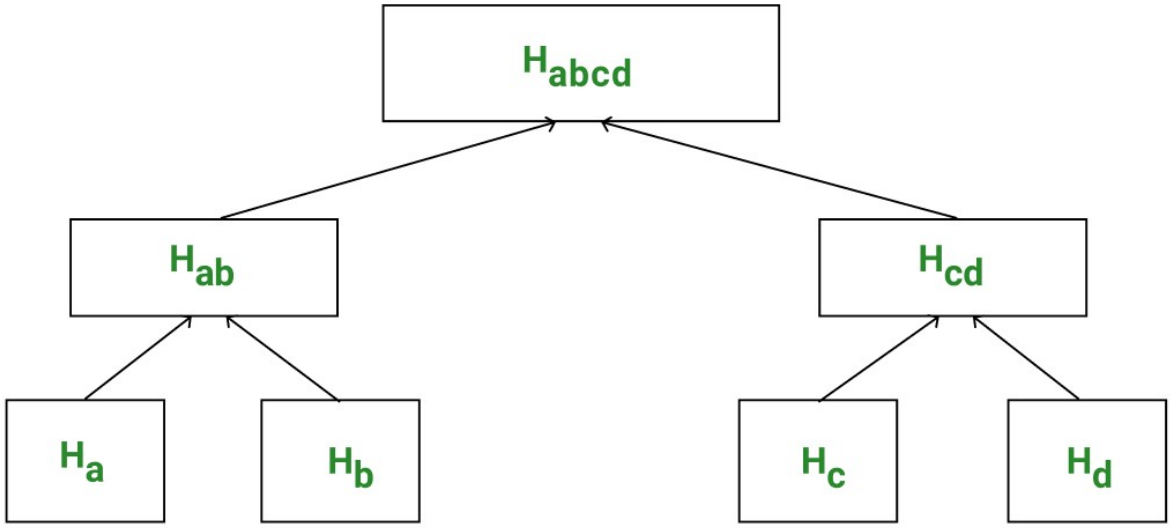
Merkle Proofs

A Merkle proof is used to decide:

1. If data belongs to a particular Merkle tree.
2. To prove data belongs to a set without the need to store the whole set.
3. To prove a certain data is included in a larger data set without revealing the larger data set or its subsets.

Merkle proofs are established by hashing a hash's corresponding hash together and climbing up the tree until you obtain the root hash which is or can be publicly known.

Consider the Merkle tree given below:



Let us say we need to prove that transaction 'a' is part of this Merkle tree. Everyone in the network will be aware of the hash function used by all Merkle trees.

1. H(a) = Ha as per the diagram.
2. The hash of Ha and Hb will be Hab, which will be stored in an upper-level node.
3. Finally hash of Hab and Hcd will give Habcd. This is the Merkle root obtained by us.
4. By comparing the obtained Merkle root and the Merkle root already available within the block header, we can verify the presence of transaction 'a' in this block.

From the above example, it is clear that in order to verify the presence of 'a', 'a' does not have to be revealed nor do 'b', 'c', 'd' have to be revealed, only their hashes are sufficient. Therefore Merkle proof provides an efficient and simple method of verifying inclusivity, and is synonymous with "proof of inclusion".

A **sorted Merkle tree** is a tree where all the data blocks are ordered using an ordering function. This ordering can be alphabetical, lexicographical, numerical, etc.

Proof of Non-Membership:

- It is also possible to test non-membership in logarithmic time and space using a sorted Merkle tree. That is, it is possible to show that a given transaction does not belong in the Merkle tree.
- This can be done by displaying a path to the transaction that is immediately before the transaction in question, as well as a path to the item that is immediately following it.
- If these two elements in the tree are sequential, this proves that the item in issue is not included or else it would have to go between the two things shown if it was included, but there is no room between them because they are sequential.

Coinbase Transaction:

A coinbase transaction is a unique Bitcoin transaction that is included in the Merkle tree of every block in the blockchain. It is responsible for creating new coins and also consists of a coinbase parameter that can be used by miners to insert arbitrary data into the blockchain.

Simple Payment Verification(SPV)

- SPV makes it extremely easy for a client to verify whether a particular transaction exists in a block and is valid **without having to download the entire blockchain.** The users will only require a copy of the block headers of the longest chain.

- This copy of headers is stored in the SPV wallet and this wallet uses the SPV client to link a transaction to a Merkle branch in a block. SPV client requests **proof of inclusion(Merkle proof),** in the form of a Merkle branch. The fact that the transaction can be linked to a Merkle branch is proof that the transaction exists.
- Now by assessing the blocks which are being mined on top of the transaction's block, the client can also conclude that majority of the nodes have built more blocks on top of this chain by using consensus mechanisms like Proof of Work, and hence this is the longest, valid blockchain.

Advantages of Merkle Tree

1. **Efficient verification:** Merkle trees offer efficient verification of integrity and validity of data and significantly reduce the amount of memory required for verification. The proof of verification does not require a huge amount of data to be transmitted across the blockchain network. Enable trustless transfer of cryptocurrency in the peer-to-peer, distributed system by the quick verification of transactions.
2. **No delay:** There is no delay in the transfer of data across the network. Merkle trees are extensively used in computations that maintain the functioning of cryptocurrencies.
3. **Less disk space:** Merkle trees occupy less disk space when compared to other data structures.
4. **Unaltered transfer of data:** Merkle root helps in making sure that the blocks sent across the network are whole and unaltered.
5. **Tampering Detection:** Merkle tree gives an amazing advantage to miners to check whether any transactions have been tampered with.
    - Since the transactions are stored in a Merkle tree which stores the hash of each node in the upper parent node, any changes in the details of the transaction such as the amount to be debited or the address to whom the payment must be made, then the change will propagate to the hashes in upper levels and finally to the Merkle root.
    - The miner can compare the Merkle root in the header with the Merkle root stored in the data part of a block and can easily detect this tampering.
6. **Time Complexity:** Merkle tree is the best solution if a comparison is done between the time complexity of searching a transaction in a block as a Merkle tree and another block that has transactions arranged in a linked list, then-
    - **Merkle Tree search:** O(logn), where n is the number of transactions in a block.
    - **Linked List search:** O(n), where n is the number of transactions in a block.