Blockchains Basics

Contents

Bitcoin & Blockchain

Bitcoin was created by *Satoshi Nakamoto*.

Blockchain **enables peer to peer transfer** of digital assets **without any intermediaries**.

The blockchain was originally **created to support Bitcoin**.

The blockchain is poised to innovate and transform a wide range of applications.

Two major contributions of cryptocurrency Bitcoin are a **continuously working digital currency system**, and a **model for autonomous decentralized application technology** called the blockchain.

With no central authority, how did Bitcoin realize trust and security? → By implementing software programs for validation, verification, consensus in a novel infrastructure called the blockchain. Computations elements were added in 2012.

How do we establish trust among the peers in such a decentralized system? → By having a process in place to validate, verify, and confirm transactions.

- **Record the transaction in a distributed ledger of blocks**
- **Create a tamper-proof record of blocks, chain of blocks,**
- **Implement a consensus protocol for agreement on the block to be added to the chain**.

So, validation, verification, consensus, and immutable recording lead to the trust and security of the blockchain.

Summarizing, blockchain technology supports methods for a decentralized peer-to-peer system, a collective trust model, and a distributed immutable ledger of records of transactions.

Blockchain structure

Many transactions → block.

Many blocks → chain through a digital data link.

Validation and consensus process are carried out by special peer nodes called miners.

The set of all UTXOs (Unspent Transaction Outputs) in a bitcoin network collectively defined the state of the Bitcoin Blockchain.

UTXO's are referenced as an input in a transaction and are also outputs generated by a transaction.

The structure of a given UTXO :

- a **unique identifier of the transaction** that created this UTXO
- an index or the **position of the UTXO in the transaction output list**

- a **value** or the amount it is good for
- an **optional script**, the condition under which the output can be spent

Does the UTXO's reference input exist in the network state? This is the only one of the many validation criteria.

Basic operations

Operations in the decentralized network are the **responsibility of the peer participants and their respective computational nodes** (laptop, desktop, server racks…).

These operations include :

- validation transactions
- gathering the transactions for a block
- broadcasting the ballot transactions in the block
- consensus on the next block creation
- chaining the blocks to form an immutable record

There are two majors roles for the participants :

- Participants that initiate transfer of value by creating a transaction
- additional participants called miners, who pick on added work or computation to verify transactions, broadcast transaction, compete to claim the right to create a block, work on reaching consensus by validating the block, broadcasting the newly created block, and confirming transactions

The miners are incentivized with bitcoins for the efforts in managing the blockchain.

All the valid transactions are added to a pool of transactions. Miners select a set of transaction from this pool to create a block. This creates a challenge. If every miner adds the block to the chain, there will be many branches to the chain, resulting in inconsistent state. Recall, the blockchain is a single consistent linked chain of flux.

We need a system to overcome this challenge → Miners compete to solving a puzzle to determine who earn the right to create the next block. In the case of bitcoin blockchain, this parcel is a computation of parcel and the central processing unit or CPU intensive.

Once a miner solves the puzzle, the announcement is broadcast to the network and the block is also broadcast to the network. Then, other participants verify the new block. Participants reach a consensus to add a new block to the chain. This new block is added to their local copy of the blockchain. Thus, a new set of transactions are recorded and confirmed. The algorithm for consensus is called proof-of -work protocol, since it involves work a computational power to solve the puzzle and to claim the right to form the next block.

Transaction zero, index zero of the confirmed block is created by the miner of the block. It has a special UTXO and does not have any input UTXO. It is called the coinbase transaction that generates a minor's

fees for the block creation. Currently, the minor's fees is 12.5 BTC for a bitcoin. This is how new coin is maintained in bitcoin.

Beyond Bitcoin

Ethereum Blockchain extended the scripting feature into a full-blown code execution framework called **smart contract**. A smart contract provides the very powerful capability of code execution for embedding business logic on the blockchain.

With the addition of code execution, comes the serious consideration about public access to the blockchain hence the classification of :

- public
- private
- permissioned blockchains based on access limits.

Smart contracts

The centerpiece and thrust of this **Ethereum** blockchain is a **smart contract**.

What is a smart contract?

A smart contract is a piece of code deployed in the blockchain node. Execution of a smart contract is initiated by a message embedded in the transaction. For example, a transaction could require a conditional transfer, it may require some evaluation, it may need more than one signature for transfer of assets, or it may involve waiting for a specific time or date.

Structurally, a smart contract resembles a class definition in an object-oriented design.

Specific programming languages have been designed for coding smart contracts. **Solidity** is one such language.

Where does the code in the smart contract get executed? Where is it located in a node?

Every node in Ethereum network should be able to execute the code irrespective of that underlying type of hardware or operating system. Enter **Ethereum Virtual Machine, EVM**. A smart contract written a high-level programming language is translated into EVM byte code, and then, deployed on the EVM.

Every node will host the same smart contract codes on the EVM.

Smart contracts add a layer of logic and computation to the trust infrastructure supported by the blockchain.

Ethereum Structure

Instead of UTXOs in Bitcoin Blockchain, Ethereum formally introduce the concept of an **account** as a part of the protocol. **The account is the originator and the target of a transaction**. A transaction directly updates the account balances.

It allows for transmit of value and messages and data between the accounts that may result in the state transitions. These transfers are implemented using transactions. There are two types of accounts, **Externally Owned Accounts** and **Contract Accounts**. Externally Owned Accounts (EOA) are controlled by private keys. Contract Accounts (CA) are controlled by the code and can be activated only by an EOA.

EOA is needed to participate in the Ethereum Network.

It interacts with the blockchain using transactions. A Contract Account ➔ smart contract. Every account has a coin balance. The participant node can send transaction for Ether transfer or it can send transaction to invoke a smart contract code or both.

Both types of **transaction require fees**. Fees are paid in Wei. *10^18 Wei = 1 ETH*

A transaction in Ethereum includes :

- the recipient of the message
- digital signature of the sender authorizing the transfer
- amount of Wei to transfer
- an optional data field or payload that contains a message to a contract
- STARTGAS which is a value representing the maximum number of computational steps the transaction is allowed
- Gas price a value representing the fee sender is willing to pay for the computations

Ethereum block structure has :

- a header
- transaction
- and runner-up block headers.

Ethereum operations

For a simple Ether transfer, the amount to transfer and the target address are specified along with the fees or gas points. The amount and the fees are transferred to their respective accounts.

An **Ethereum node** is a computational system representing a business entity or an individual participant.

An **Ethereum full node** hosts the software needed for :

- transaction initiation
- Validation

- Mining
- block creation
- smart contract execution
- the Ethereum Virtual Machine, EVM.

When the target address in a transaction is a smart contract, the execution code corresponding to the smart contract is activated and executed on the EVM. The input needed for this execution is extracted from the payload field of the transaction.

Current state of the smart contract is the values of the variables defined in it.

All the transactions generated are validated. Transaction validation involves checking the timestamp and the nonce combination to be valid and the availability of sufficient fees for execution.

Miner nodes in the network receive, verify, gather, and execute transactions. The in-work smart contract code is executed by all miners. Validated transactions are broadcast and gathered for block creation. **The consensus protocol used is a memory-based** rather than a CPU-based **proof of work**.

Incentive model

Mining is the process used to **secure the network** by :

- validating the computations
- collecting them to form a block
- verifying them
- broadcasting it

Every action in Ethereum requires crypto fuel, or **gas**. Gas points are used to specify the fees inside of Ether, for ease of computation using standard values.

Ether, as a cryptocurrency, varies in value with market swings, but **gas points do not vary**.

Ethereum **has specified gas points for each type of operation**.

If the fee specified and the gas point in the transaction are not sufficient, it is rejected. The gas points needed for execution must be in the account balance for the execution to happen. If there is any amount left over after the execution of a transaction, it is returned to the originating account.

Gas limit is the **amount** of gas points available for a block to spend.

Gas spent is the **actual amount** of gas spent at the completion of the block creation.

The proof of work puzzle **winner**, miner that creates a new block, is incentivized with the **base fees of three Ethers**, and the **transaction fees in Ethereum blockchain**. The winning miner also gets the fees, gas points for execution of a smart contract transactions.

These miners will solve the puzzle but didn't win the block are called Ommers. Ommer miners also get a small percentage of the total gas points as a consolation and for network security.

Public-Key Cryptography

Two techniques are predominantly used for securing the chain and for efficient validation and verification :

- **hashing**
- **asymmetric key encryption**

Public-key cryptography algorithm answers to these questions :

- how do you identify the peer participants?
- How do you authorize and authenticate the transactions?
- How do you detect forged or faulty transactions?

Example : Instead of sending just a simple message, a participant in Buffalo will send a transaction data encrypted by Buffalo's private key, and then encrypted by Kathmandu's public key. Kathmandu will first decrypt the data using its own private key, then use Buffalo's public key to decrypt assigned transaction data. **This ensures that only Kathmandu can decrypt and receive the data and that only Buffalo could have sent the data**.

Though RSA is very commonly used in many applications, block chains need a more efficient and stronger algorithm. Efficiency is a critical requirement since public key pair is frequently used in many different operations in block chain protocol. **Elliptic Curve Cryptography**, ECC family of algorithms is used in the bitcoin as well as an Ethereum block chain **for generating the key pair.**

Hashing

A hash function or hashing transforms and maps an arbitrary length of input data value to a unique fixed length value.

The following are two basic requirements of a hash function :

- the algorithm chosen for the hash function should be a one-way function and it should be collision free or exhibit extremely low probability of collision.
- to make sure that the hash value uniquely represents the original items hashed.

In the simple hash approach, all the data items are linearly arranged and hashed.

In a tree-structured approach, the data is at the leaf nodes of the tree, leaves are pairwise hash to arrive at the same hash value as a simple hash.

When we have a **fixed number of items** to be hashed, such as the items in a block header, and we are verifying the composite block integrity and not the individual item integrity, we use **simple hash**.

When the **number of items differ from block to block**, for example, number of transactions, number of states, number of receipts, **we use the tree structure** for computing the hash. Tree structure helps the

efficiency of repeated operations, such as transaction modification and the state changes from one block to the next.

Transaction Integrity

To manage the integrity of a transaction we need :

- secure & unique account address
- authorization of the transaction by the sender through digital signing
- verification that the content of that transaction is not modified

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

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

A transaction for transferring assets should be :

- **authorized**
- **non-repudiable**
- **unmodifiable**

Data is hashed and encrypted. This is the **digital signature**. The receiver gets the original data, and the secure hash digitally signed. Receiver can recompute the hash of the original data received and compare it with the received hash to verify the integrity of the document.

Consider the transaction to be that data :

- ➔ *Step 1* : find the hash of the data fields of the transaction.
- ➔ *Step 2* : encrypt that hash using the private key of the participant originating the transaction. Thus, digitally signing the transaction to authorize and making the transaction non-repudiable.
- ➔ *Step 3* : this hash just added to the transaction. It can be verified by others decrypting it using the public key of the sender of the transaction and recomputing the hash of the transaction. Then, compare the computed hash, and the hash received at the digital signature. If that is a match, accept the transaction. Otherwise, reject it.

Note that for the complete transaction verification, the timestamp, nons, account balances, and sufficiency of fees are also verified.

Securing Blockchain

Integrity of the block is managed by assuring that the block header contents are not tampered with, the transactions are not tempered with, state transitions are efficiently computed, hashed, and verified.

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

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

Advantage over flat versus tree representation : if any transaction is to be verified, only one path to the tree must be checked. You don't have to go through the entire set of transactions.

Smart contract execution in Ethereum results in state transitions. Every state change requires state root hash re-computation. Instead of computing hash for the entire set of states, only the affected path in the Merkle tree needs to be re-computed.

Block hash in Ethereum is computed by first computing the state root hash, transaction root hash and then receipt root hash, shown at the bottom of the block header. These roots and all the other items in the header are hash together with the variable nodes to solve the proof of work puzzle.

If any participant node tampers with the block, its hash value changes resulting in the mismatch of the hash values and rendering the local chain of the node in an invalid state. Any future blocks initiated by the node would be rejected by other miners due to hash mismatch. **This enforces the immutability of the chain**.

Decentralized system

In a decentralized system, there is nobody checking your credentials and certifying that you are trustworthy. Then, how do you do it? You do it by using algorithms and techniques discussed in the last section. Let's examine how these will help address the trust issues in a blockchain.

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

This is accomplished by :

- securing the chain using specific protocols
- validating the transaction and blocks for tamper proofing
- verifying the availability of resources for transactions
- executing and confirming the transactions

The Trust Trail is defined by these operations :

- validate transaction (1)
- verify gas and resources (2)
- gather transactions (3)
- execute transaction to get a new state (4)
- form the block (5)
- work towards consensus (6)
- finalize the block by the bidder (7)
- everyone adds the block to their chain and confirm the transactions (8)

Step 1 and 2 → The syntax, the transaction signature, time stamp, nonce, gas limit, and sender account balance are **validated before execution**. The fuel, or gas points, and other resources available for smart contract execution, are also **validated**. Transaction signatures and hash are also **verified**.

Step 3 → Merkle tree hash of the validated transactions is computed. This is in Ethereum. This is the transaction root of the block header. All miners execute the transaction for either transfer, as well as for execution of smart contracts. The state resulting from transaction execution are used in computing the Merkle tree hash of the states, the state root of the block header. The receipt root of the block header is also computed.

Consensus Protocol

A secure chain is a single main chain with a consistent state. Every valid block added to this chain, adds to the trust level of the chain. Each of the candidate blocks is by a competing miner.

The method to choose the next block is called Proof of Work. The point of view of the miner : If hash value is less than 2 par 128 for bitcoin, and less than function of difficulty for Ethereum, the puzzle has been solved. If it has not been solved, repeat the process after changing the nonce value. If the puzzle has been solved, broadcast the winning block that will be verified by other miners. Non-winning miner nodes add the new block to the local copy of the chain and move on to working on the next block.

Robustness

Robustness is the ability to satisfactorily manage exceptional situations.

Two exceptions :

- What if more than one miner solves the consensus puzzle where it close in time to each other?
- What if more than one transaction references as input the same digital asset? (double spending)

1. For Bitcoin : Two miners have solved the consensus puzzle very close to each other. Bitcoin protocol allows this chain split or two chains for the next cycle. One led by each of the competing blocks. The probability that the next block will happen at the same time in both these chains is extremely low. So, the winner of the next cycle for block creation consolidates one of the chains and that chain becomes the accepted chain. In this case, the newest block is added to the main chain. Now this **chain is the longest and the valid main chain**. The transaction in the other blocks are returned to the unconfirmed pool. Summarizing with a very low probability, **the main chain may split but if it does, the bitcoin protocol has methods to consolidate it to a single chain within a cycle**.
   For Ethereum: Ethereum handles more than one person we know by allowing Omar or Runner-Up blocks and allocating a small incentive for these Runner-Up blocks. This incentive model helps in keeping the chains secure. **New blocks are added only to the mainchain and not to the Runner-Up chains**. That are Runner-up blocks are maintained for six more blocks after they were added.

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

Forks

Soft fork and hard fork in the blockchain word are like the release of software patches, and new versions of operating systems, respectively. Forks are mechanisms that add to the robustness of the blockchain framework. Well-managed forks help build credibility in the blockchain by providing approaches to manage unexpected faults and planned improvements.