**PEER-TO-PEER BLOCKCHAIN PROGRAMMING WITH PYTHON**

**Introduction**

Before we look at the various aspect of peer to peer blockchain programming, let us understand the meaning. According Toshendra (2021) Peer to peer network, commonly known as P2P is a decentralized network communications model that consists of a group of devices (nodes) that collectively store and share files where each node acts as an individual peer. In this network, P2P communication is done without any central administration or server, which means all nodes have equal power and perform the same tasks.

P2P architecture is suitable for various use cases and can be categorized into structured, unstructured, and hybrid peer-to-peer networks. The unstructured peer-to-peer networks are formed by nodes randomly from connection to each other, but they are inefficient than structured ones. In structured peer-to-peer systems, the nodes are organized, and every node can efficiently search the network for the desired data. Hybrid models are actually a combination of P2P and client-server models, and when compared to the structured and unstructured P2P systems, these networks tend to present improved overall performance.

Today, P2P networks are the foundation of most of the cryptocurrencies, thus, making up a significant portion of the blockchain industry. Every transaction in a blockchain works in a distributed Peer-to-Peer (P2P) network that has no central authority to control data. It allows everyone (having access) to join the blockchain, and every computer added to the network is a node.

So, when a user creates a new block, it goes to each user on the network, and each node must verify this new block to ensure no one has altered it. When the verification is complete, each node starts adding the new block directly to their blockchain.

All the nodes present in the network make a consensus, confirming the validity of blocks and rejecting the tampered ones.

Our discussion will be centered on the following four **Layers of Blockchain**

**Application / Services Layer**

* + Side Chain
  + Smart contract
  + Cryptocurrency
  + Digital Copy

**Blockchain Consensus Layer**

* + Proof of Stake (PoS)
  + Delegated Proof of Stake (DPoS)
  + Proof of Work (PoW)

**Data Layer**

* + Data Storage
  + Merkle tree
  + Block

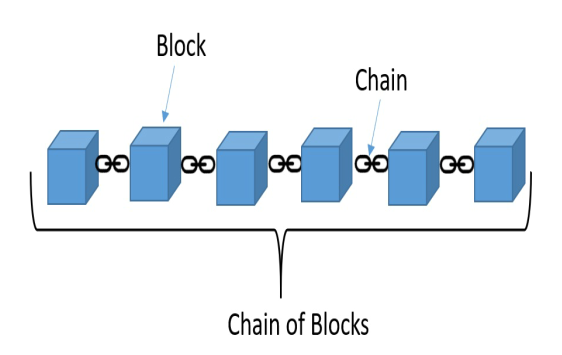
**Network Layer**

* + Communication Mechanism
  + Peer-to-Peer (P2P) Network

**Blockchain on P2P Network**

P2P is a technology that is based on a very simple principle, and that is the concept of decentralization. The peer-to-peer architecture of blockchain allows it to run on the peer-to-peer network which enables all cryptocurrencies to be transferred worldwide, without the need of any middle-man or intermediaries or central server. With the distributed peer-to-peer network, anyone who wishes to participate in the process of verifying and validating blocks can set up a Bitcoin node.

In simple term blockchain is just a collection of digital information stored on a public database. The data is referred to as the block while the database is the chain of blocks. By this understanding we can define **Blockchain as a chain of linked blocks, where each block stores a transaction made with specific parameters**. Blockchain is a decentralized ledger tracking of one or more digital assets on a peer-to-peer network. The beauty of blockchain technology is that it is open and distributed ledger system that can document transactions between parties. These transactions are efficiently and verifiably recorded permanently. Consequently, blockchain technologies are coded in open-source protocols through public and private chains. As such, the distributed ledger transactions are recorded and shared by several participants in the blockchain system, without full control of a single entity. The superiority of blockchain technology is that since it is decentralized peer-to-peer network where all the computers are connected in some way, and where each maintains a complete copy of the ledger and compares it to other devices to ensure the data is accurate. This is unlike a bank, where transactions are stored privately and are managed only by the bank.

Figure I: Blockchain

blockchain, is the solution, where transactions are maintained in a distributed shared ledger and replicated across a global P2P network; security and privacy are ensured with cryptographic technologies, and transaction integrity is achieved through a consensus mechanism.

As new transactions are made, they are broadcasted to all network nodes, and over time all transactions that have occurred are sequenced together in the public ledger and made available in all replicated network nodes, as shown in the following diagram below

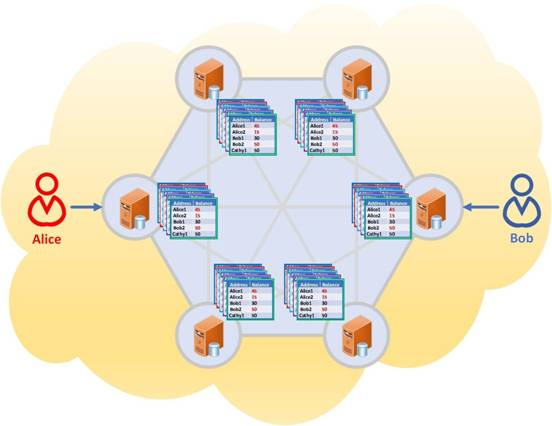


Figure II: Transactions on distributed blockchain

**Versions of Blockchain**

**1. BlockChain 1.0 (Cryptocurrency)**BlockChain Version 1.0 was introduced in 2005 by Hall Finley, who implements DLT (Distributed Ledger Technology) represents its first application based on Crypto currency. This allows Financial Transaction based on BlockChain technology or DTL which is executed with the help of BitCoin. This type of Version is permissionless as any participant will perform valid transaction of Bitcoin. This type is mainly used in Currency and Payments.

**2. BlockChain 2.0 ( Smart Contracts)**The new Version of BlockChain come because there is a problem in version 1.0 which was Mining of BitCoin was Wasteful and there was also lack of Scalability of Network in it. So problem is improved in Version 2.0. In this version, the BlockChain is not just limited to Cryptocurrencies but it will extend up to Smart Contracts.Thus, Small Contracts are Small Computer’s which live in the Chains of Blocks. These Small Computer’s are  free computer programs that executed automatically, and check the condition defined earlier like facilitation, verification or enforcement and reduce transactions cost efficiency. In BlockChain 2.0,  BitCoin is replaced with Ethereum. Thus, BlockChain 2.0 was successfully processing high number of Transactions on Public network rapidly.

**3. BlockChain 3.0 (DApps)**After Version 2.0, new version was introduced which includes DApps which is known as Decentralized Apps. A DApp is like a conventional app, it can have frontend written in any language that makes calls to its backend, and its backend code is running on decentralized Peer-To-Peer Network. It makes use of decentralized storage and communication which can be Ethereum Swarm etc.

There are many decentralized Applications like BitMessage, BitTorrent, Tor, Popcorn, etc.

**The bottlenecks with the preceding versions building up to Blockchain 3.0**

Blockchain 3.0 targets to address the scalability, sustainability, cost, interoperability, and security related dilemmas that we had with blockchain 1.0 and 2.0, promising better solutions with a refined framework. Before we give you concrete details about blockchain 3.0, we will first have to make you understand what dragged blockchain experts to think about blockchain version upgradation? To know why, we will have to go back to the time when Satoshi Nakamoto wrote a whitepaper on bitcoin back in the year 2009. This was when blockchain 1.0 came into existence. The excitement around bitcoins grew tremendously then. The concept that any middlemen do not back this digital currency (read, government), unlike fiat currencies, grabbed everyone’s attention. More to it, everyone was thankful for the underlying technology, blockchain, that facilitated a secure, efficient, and transparent environment for carrying transactions, cutting down the middlemen and double-spending. But a common question, where do this extraordinary cryptocurrency come from, arose? For bitcoins, miners have to solve a highly complex math problem, so difficult that it is impossible to solve the problem by us. Hence, miners had to use high-powered computers, and the likelihood of the problem getting solved was 1 in 7 trillion. Beyond the wastage of resources, bitcoins possessed scalability issues, having transaction processing speed capacity between 3.3 and 7 transactions per second. Soon, the limitations were taken into consideration, which was why professionals launched blockchain 2.0. In this version, we saw Ethereum bragging its transaction processing speed, which is roughly 15 transactions per second (double of bitcoins TPS).

Blockchain 2.0 saw the birth of smart contracts, computer programs that execute autonomously when a predefined condition meets, staying on the blockchain network. Though the second version was far better and quicker than the first version, it tremendously failed really to meet the pace of the centralized payment mediums like Visa and PayPal (how is explained later). Both the versions were an unfeasible, unsecured (remember 51% attack?), and expensive solution. And, such a serious situation opened opportunity doors for blockchain 3.0. Let's see what Blockchain 3.0 is. fintech players. And for this to happen, the scalability issues had to be first considered and solved. Blockchain 3.0 is an upgraded version of blockchain 2.0, built to improve the technology’s capabilities and solve the existing problems, while facilitating speedier, cost-effective, and efficient transactions. One of the things that make blockchain 3.0 notable and viable is DAG (Directed Acyclic Graph). Let’s first understand the logic behind the data structure, DAG. As the name implies, the information on a DAG-based network flows acyclically. So, this means that the information cannot be sent back to the sender. The information will flow in only one direction. It makes sure that nodes are not connected to any previous ones. Such a structure eliminates the block times, which is 10 minutes for bitcoins and 20 seconds for Ethereum, thereby allowing transactions to get processed almost in real-time. DAG is being used by IoT chain (ITC) and it processes 10,000 transactions per second, which is far more than Visa.

**Blockchain 3.0: The Newcomers Welcomed**

As Blockchain 1.0 and Blockchain 2.0 tremendously failed in its endeavors to gain mass confidence, we have a new era of the technology waiting for us to embrace it - BLOCKCHAIN 3.0.Cardano - Spearheaded by Charles Hoskinson (one of the co-founders of Ethereum), Cardano is an advanced blockchain platform, that includes smart contracts, transaction systems, and Dapps, developed out of scientific philosophy and high-level research. Pioneering an altogether new approach for digital currencies, Cardano is leading the charge of Blockchain 3.0. Let’s see how Cardano is better than Ethereum.

* Cardano uses Haskell programming language, whereas Ethereum uses Solidity. Haskell is a universally accepted programming language with non-strict semantics, whereas Solidity is a contact-oriented language explicitly built for writing smart contracts. Haskell enables developers to write codes precisely, enabling them to establish efficient and secure protocols. Solidity, on the other hand, possess serious security issues.
* Cardano uses a proof of stake consensus mechanism, whereas Ethereum uses proof of work. Instead of miners wasting their resources for computing the math problem, the POS system relies on factors such as the wealth of the creator and complexity levels in the network at that time. POS consensus mechanism not only saves energy resources but also reduces the risks of 51% attack (because, an attacker trying to purchase 51% of coins will automatically see the coin’s cost appreciation).

**Types of Blockchains**

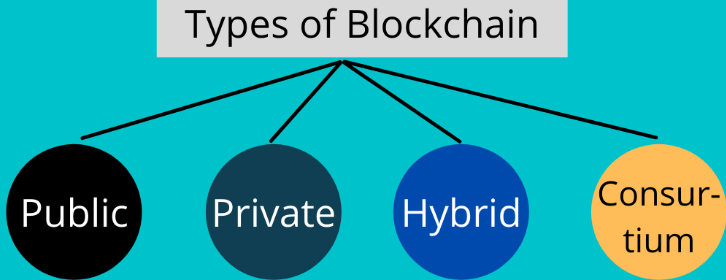


Figure III: Types of blockchain

Blockchain is of different types, and users leverage this technology for many use cases depending upon its type. So, different types of blockchains are:

**Public Blockchains**

Blockchains facilitate a decentralized, open network of multiple computers that anyone can access to request or verify a transaction for accuracy. It allows users to create new blocks, access all the blocks in the blockchain, and validate the data.

As they are open and need excellent security, they use concepts like proof of stake or proof of work. The block miners who validate transactions are rewarded financially. Public blockchains are mainly used for [mining](https://geekflare.com/finance/cryptocurrency-mining-for-beginners) and exchanging cryptocurrencies.

**Examples**: Bitcoin, Litecoin, and Ethereum blockchains.

**Private Blockchains**

Private blockchains are centralized and governed by a person or organization that decides who can access the blockchain, be added as a node, and verify records. Unlike public blockchains, private blockchains aren’t open and offer access restrictions. If anyone wants to join a private blockchain, they must obtain permission from the administrator.

**Example**: B2B virtual currency exchanges like Hyperledger.

**Consortium Blockchains**

A group of companies or organizations govern these permissioned blockchains instead of one individual. They are more decentralized than a private blockchain to enjoy more security. It allows limited access and the present nodes determine the consensus process.

Furthermore, it constitutes a validator node to initiate, receive, and verify transactions while member nodes have permission to initiate or accept transactions. Here, users can transfer digital assets from one blockchain to another with improved efficiency and scalability.

**Example**: Consortium blockchains are used in payments and banking, such as Quorum and Corda.

**Hybrid Blockchain**

Hybrid blockchains combine the attributes of private and public blockchains. They can be centralized or decentralized and allow organizations to set up a permission-based private blockchain along with a public blockchain. Thus, the organizations can control the data access in the blockchain and what data to access publicly.

**Example**: It’s used in real estate and retail industries, such as IBM Food Trust.

In this article we will dwell more on public blockchain, owing to the fact that they are more popular than the private block, notwithstanding I would like to use e-naira as a case study for private blockchain before we continue with public blockchain.

**E-Naira on private blockchain**

Before we give a detailed description of how e-naira works on the private blockchain, I want to emphasize that e-naira is not a cryptocurrency but the digital equivalent of hard printed naira. Transactions with crypto on public blockchain is immutable and unchangeable due to consensus algorithms that validates transactions, those on the private blockchain like e-naira can be manipulated.

A private blockchain is accessible and controlled by a single entity, organization, or enterprise. The Hyperledger Fabric, chosen to run the e-naira by the CBN, is an example of a private blockchain. Hyperledger Fabric is **a modular blockchain framework** that acts as a foundation for developing blockchain-based products, solutions, and applications using plug-and-play components that are aimed for use within private enterprises. They’re also known as permissioned blockchains. A trusted intermediary can choose to grant full or partial access to another entity if it deems fit.

Bitcoin made blockchain technology famous, so like the cryptocurrency, the tech is associated with decentralization and openness. But private blockchains are the opposite. Although it’s the same tech, there’s no form of decentralization; a single entity control everything. Blockchain is a database or a record book, if you like, that stores data. It is a distributed ledger that stores transactions that have been validated on a blockchain network.

Validated transactions are aggregated and stored in a block; this block is added to previous blocks creating a chain, hence the word blockchain. This means the CBN will have total control. A trusted intermediary can either build the entire blockchain software from scratch or license it from a third party like the Hyperledger Fabric, as the apex bank did.

The private blockchain network is set up using nodes that run this software. A node is simply a system that validates transactions in a blockchain network.  The CBN will run the validator nodes itself or use a subsidiary — most likely BItt Inc. It will also create an app with which users — Nigerians — can interface with these nodes. Once you carry out a transaction on the app, it is relayed to the CBN who validates them.

Also, there are two major ways to set up a private blockchain: distributed ledger model, and shared ledger model.

From the presentation by the CBN containing the role of strategic stakeholders in the e-Naira launch, it can be inferred that the regulator will most likely use the shared ledger model.

**Backdrop with e-naira**

Some of the reasons people run a private blockchain is to remove the need for transparency, to be able to scale, increase transaction speed with reduced energy consumption, and reduce the need for numerous gadgets in terms of nodes manipulation of information is possible on a private blockchain.

**Properties of a blockchain**

A blockchain is essentially a decentralized distributed database or a ledger and transactions are immutable as follows:

**Decentralization:** In simple terms, it means that the application or service continues to be available and usable even if a server or a group of servers on a network crashes or is not available. The service or application is deployed on a network in a way that no server has absolute control over data and execution, rather each server has a current copy of data and execution logic.

**Distributed:** This means that any server or node on a network is connected to every other node on the network. Rather than having one-to-one or one-to-many connectivity between servers, servers have many-to-many connections with other servers.

**Database:** This refers to the location for storing durable data that can be accessed at any point in time. A database allows storage and retrieval of data as functionality and also provides management functionalities to manage data efficiently, such as export, import, backup, and restoration.

**Ledger:** This is an accounting term. Think of it as specialized storage and retrieval of data. Think of ledgers that are available to banks. For example, when a transaction is executed with a bank—say, Tom deposits 100 naira in his account, the bank enters this information in a ledger as a credit. At some point in the future Tom withdraws 25 naira. The bank does not modify the existing entry and stored data from 100 to 75. Instead it adds another entry in the same ledger as a debit of 25 naira. It means a ledger is a specialized database that does not allow modification of existing data. It allows you to create and append a new transaction to modify the current balance in the ledger. The blockchain is a database that has the same characteristics of a ledger. It allows newer transactions to be stored in an append-only pattern without any scope to modify past transactions. It is important here to understand that existing data can be modified by using a new transaction, but past transactions cannot be modified. A balance of 100 naira can be modified at any time by executing a new debit or credit transaction, but previous transactions cannot be modified.

**Immutability:** Blockchain is an immutable (unchangeable) and shared digital ledger that stores records or transactions in several places on a network of computers. Here, each verified transaction is added in a space called a block that links with other subsequent blocks with the help of cryptography, forming a chain.

**What Are the Building Blocks of Blockchain?**

**Block**

As explained above, blockchain refers to a chain of different blocks that contain data or records. And the data in each block is based on the kind of [blockchain](https://geekflare.com/learn-blockchain/). For example, a banking blockchain will have blocks containing information such as account number, account holder’s name, branch name, etc.

The first block in a blockchain is known as the Genesis block, and all the blocks contain valid records encoded and hashed. Each block has a cryptographic hash of its own and that of the previous block in the same blockchain, linking them and forming a chain. This iterative process validates the previous blocks’ integrity with digital signatures. **Types of information stored in a block**

The blocks store key information like the date and time of a transaction. It also stores the amount of money transferred at that instant. Now, this is added with a digital signature before storing it in a block. This digital signature serves as your username, claiming that the data in the block belong to you.

Each block is given a unique **cryptographic code referred to as a Hash**. This allows you and other users to differentiate blocks from each other

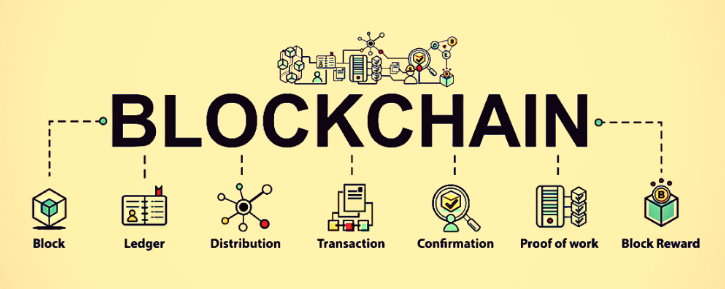


Figure IV: Components of a blockchain

**Merkle Tree**

A Merkle tree, also known as a Merkle Patricia tree, is a type of data structure that comprises multiple hashes or timestamps stored in the blockchain. As you can probably guess from its name, it’s a method of storing data in the blockchain by using hashes. What’s great about this method is that it allows for the problem of having duplicate entries in the chain to be eliminated and that all transactions are verified at all times. This way, you don’t have to trust your wallet provider to be keeping up-to-date transaction histories. Instead, you can verify all of your transactions manually at any time. No one can alter the blockchain. They could create a false record that’s easily detected due to being in a different place in the chain. We will discuss more about this in hash function.

**Blockchain Consensus algorithm**

A consensus algorithm inside a blockchain is the set of rules of the house about how the blockchain is governed, how users formulate and agree on the rules, and how transactions happen. Governance is an important aspect for a blockchain because it determines how decentralized or centralized the network is.

For instance, blockchain consensus algorithms allow any user to propose changes to the network and all others to vote on these proposals. In the delegated versions, users select delegates who make rules and govern the network on behalf of other users.

Some versions allow users to contribute to governance based on the number of resources (compute or amount of cryptocurrencies) they contribute. In Bitcoin, for instance, miners vote on changes based on the number of computer resources or computing power they contribute to support the network and approve transactions.

In proof of Work algorithm, miners compete to create a block and the one who creates a block successfully is rewarded with the cryptocurrency after creation. Miners vote to allow or reject proposals of upgrades forwarded by other users.

**Blockchain Nodes**

A blockchain exists out of blocks of data. These blocks of data are stored on **nodes.** Nodes can be any kind of device (mostly computers, laptops or even bigger servers). Nodes form the infrastructure of a blockchain. All nodes on a blockchain are connected to each other and they constantly exchange the latest blockchain data with each other so all nodes stay up to date. They store, spread and preserve the blockchain data, so theoretically a blockchain exists on nodes. A full node is basically a device (like a computer) that contains a full copy of the transaction history of the blockchain.

**What do nodes do?**

When a miner attempts to add a new block of transactions to the blockchain, it broadcasts the block to all the nodes on the network. Based on the block’s legitimacy (validity of signature and transactions), nodes can accept or reject the block. When a node accepts a new block of transactions, it saves and stores it on top of the rest of the blocks it already has stored. In short, here is what nodes do:

* Nodes check if a block of transactions is valid and accept or reject it.
* Nodes save and store blocks of transactions (storing blockchain transaction history).
* Nodes broadcast and spread this transaction history to other nodes that may need to synchronize with the blockchain (need to be updated on transaction history).

**Who can run a node?**

Some blockchains have thousands of nodes simultaneously online. Anyone can run a node by simply downloading the transaction history of a blockchain. Many crypto and blockchain enthusiasts are running nodes voluntarily. They do this to contribute to a blockchains community, its development, security and integrity, but also simply because it’s their hobby and makes them feel part of the project. Running a node is considered fairly simple for someone slightly tech savvy, and does not require a lot of resources. Some blockchains however, now contain such a large amount of transaction data that it actually requires a lot of memory on a device to run a full node. Many crypto users who just want to *use* a blockchain therefore use **wallet applications**. These applications allow them to broadcast transactions from their wallet without being required to download the entire blockchain history on their own device.

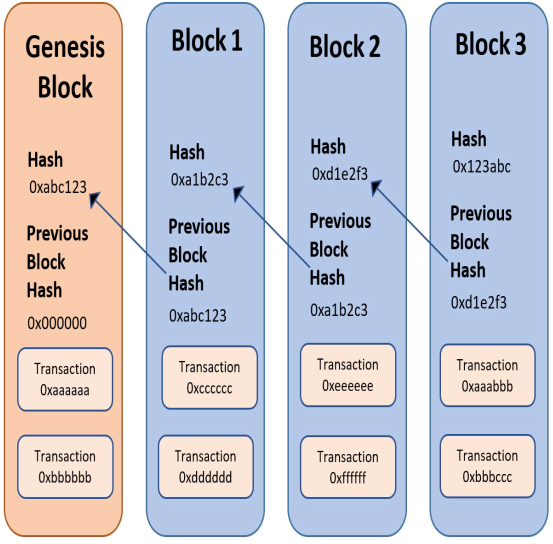
**What are master-nodes?**

And finally, some blockchains also feature **master-nodes**. Master-nodes are usually heavier equipped than normal nodes. Next to validating, saving and broadcasting transactions, master-nodes sometimes also facilitate other events on the blockchain dependent on their nature, such as governing voting events, providing execution of protocol operations and enforcing the laws of the according blockchain. Master-nodes are generally **always**online (24/7), and facilitate much more memory than normal nodes. You could say a master-node is like hosting a very large server on the network. Because hosting a master-node usually requires much more resources (electricity, up-time, maintenance, storage space, memory), hosting one generally provides payment in the form of interest.

**How are blocks related to each other?**

In blockchain and Ethereum every block is related to another block. There is a parent-child relationship between two blocks. There can be only one child to a parent and a child can have a single parent. This helps in forming a chain in blockchain.

blocks are shown—**Block 1**, **Block 2**, and **Block 3**. **Block 1** is the parent of **Block 2** and **Block 2** is the parent of **Block 3**. The relationship is established by storing the parent block's hash in a child's block header. **Block 2** stores the hash of **Block 1** in its header and **Block 3** stored the hash of **Block 2** in its header. So, the question arises—who is the parent of the first block? Ethereum has a concept of the **genesis block** also known as **first block**. This block is created automatically when the chain is first initiated. You can say that a chain is initiated with the first block known as the **Genesis Block** and the formation of this block is driven through the genesis.json file. Let's take a look at the following diagram:



**Mining**

The act of producing units of a cryptocurrency (such as bitcoins) through some kind of effort. The effort is required so that people can't just create infinite amounts of the digital currency, which would devalue it. In bitcoin, mining requires computing power. Here is a detailed description of how mining works. Bitcoin mining is the process of making computer hardware do mathematical calculations for the Bitcoin network to confirm transactions and increase security. As a reward for their services, Bitcoin miners can collect transaction fees for the transactions they confirm, along with newly created bitcoins. Mining is a specialized and competitive market where the rewards are divided up according to how much calculation is done. Not all Bitcoin users do Bitcoin mining, and it is not an easy way to make money.

**Smart contracts**

Smart contracts are simply programs stored on a blockchain that run when predetermined conditions are met. They typically are used to automate the execution of an agreement so that all participants can be immediately certain of the outcome, without any intermediary’s involvement or time loss. They can also automate a workflow, triggering the next action when conditions are met. A smart contract is also **a self-executing contract with the terms of the agreement between buyer and seller being directly written into lines of code**. The code and the agreements contained therein exist across a distributed, decentralized blockchain network.

**Sidechain**

**Sidechain is a separate blockchain that acts as an extension to the parent blockchain, often referred to as the mainchain.** To create a sidechain, it is necessary to introduce an off-chain process that will propagate the data between both blockchains. This will allow for the transfer of the assets or synchronization of any kind of data between the blockchains.

**Proof of Work**

 This algorithm is based on the idea of solving a complex mathematical puzzle to give a solution block. It requires a lot of computational power and the miner who solves the puzzle to mine a block and gets rewarded by Bitcoins. For moving forward with this step miners are required to solve the complex mathematical problem. Computers with high computational powers are used for solving complex problems. This entire process of solving the problem is known as mining.

Mining is the key to unlock the entry of a new block in the database system. Miners are the people who perform mining. The mining process is extremely excruciating for the miners, but the ones who perform it well get rewards in bitcoin.

**Validation**

A group of miners collectively share their rewards that they have obtained via solving the mathematical problems. As the network size increases it becomes even more difficult to add blocks. The equation keeps on expanding in size and further validation keeps getting complex too. Every new block is then validated multiple times before making them a part of the chain.

The monopoly soon starts taking in and a lot of blocks are refrained from entering. Power of mining also becomes concentrated in the hands of a few.

**Staking the Cryptocurrency in the Proof of Stake**

This algorithm validates a block, with the block creator being selected based on the number of coins saved on the wallet. They then get rewarded for finding the block. In other words, the algorithm computer code awards the highest chance of mining most transactions and therefore the block, to the person with the highest amount of coins in the validator’s pool.

In the next rounds of validation, the chances for the previously selected validator keep reducing until other validators also get the chance to verify a block. *Only the ones with cryptocurrency can make the changes.* Staking the[cryptocurrency](https://www.analyticssteps.com/blogs/what-cryptocurrency) is the last step in the transaction process. In this step, the users who are willing to make changes to the blockchain network must hold any specific cryptocurrency.

Proof of Stake eradicates the need for proof of work. Mining takes a back seat when this step is put in action. Users can make changes without solving the complex tasks. It also helps in saving energy.

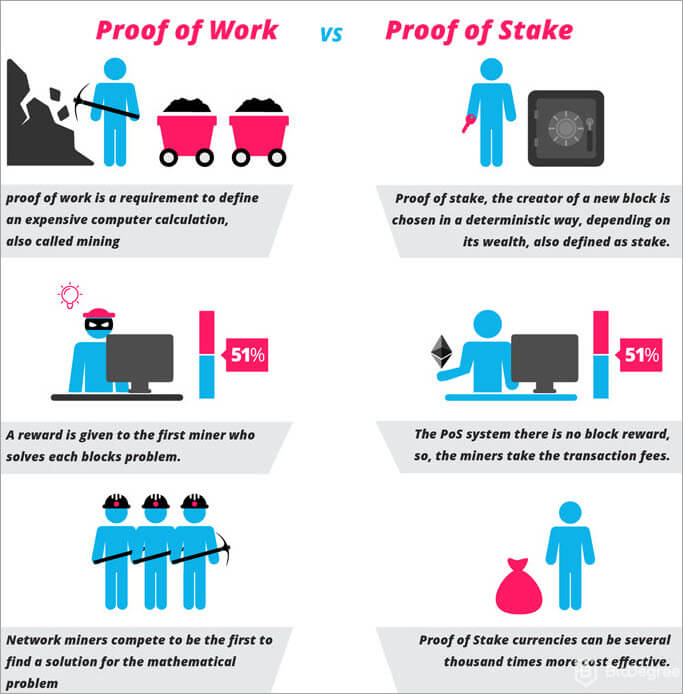
**Delegated Proof-of-Stake (DPoS)**

In Delegated Proof-of-Stake (DPoS), stakers select delegates and outsource validation of the block to them. Stakeholders will cast a vote to select delegates.

Other algorithms include Delegated Proof-of-Stake (DPoS), stakers select delegates, and outsource validation of the block to them. Stakeholders will cast a vote to select delegates; Byzantine Fault Tolerance (BFT) which select validators of blocks based on their reputation by leveraging their real identities.

The trustworthy validators are pre-approved and selected by participants to act as moderators. Others are Practical Byzantine Fault Tolerance (pBFT); Federated Byzantine Agreement (FBA); and Delegated Byzantine Fault Tolerance (dBFT).

Some blockchains use hybrid algorithms to leverage the benefits of more than one algorithm.



**Figure V: Proof of Work versus Proof of stake**

**How Does a Blockchain Transaction Work?**

This is how a typical transaction occurs in a blockchain:

**Step 1: Transaction request**

First, an individual request a transaction involving real estate, banking, cryptocurrency, records, contracts, etc.

**Step 2: Distribution**

The transaction requested gets broadcasted in the peer-to-peer network through nodes located across the globe.

**Step 3: Validation**

The nodes in the network validate the transaction using algorithms and solving complex equations. If they find the transaction legitimate, the records are entered inside blocks.

**Step 4: Adding blocks to the blockchain**

After the transaction completes, the newly created block is chained with the previous block with cryptography and encryption. It has a hash code and contains the hash code of the previous block. Once this block has filled its allocated space, the next block starts filling and attaching to the previous block; hence a long chain of transactions is formed. This is immutable and transparent for everyone in the blockchain.

The steps are further explained below

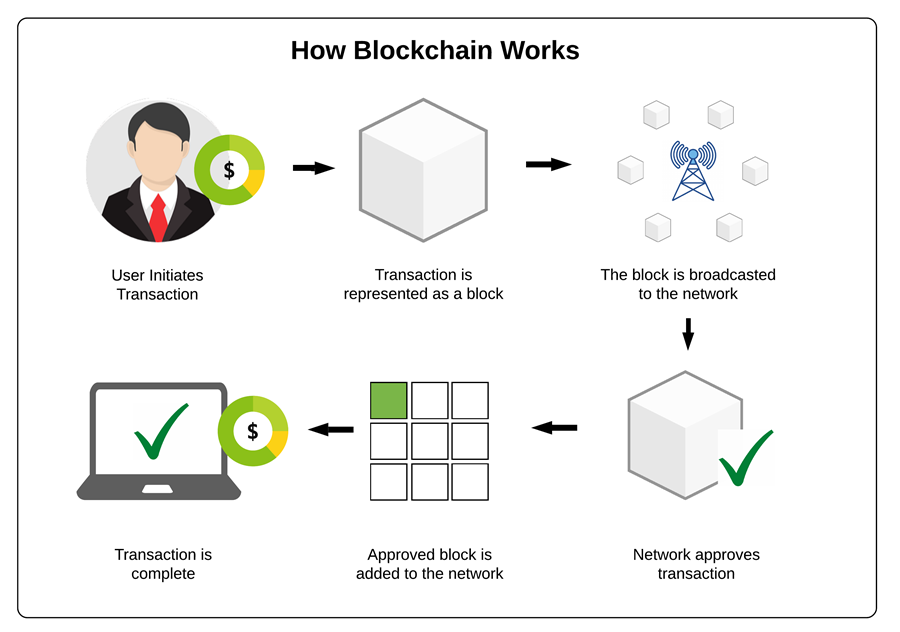


Figure VI:Phases of blockchain transaction

**Factors that contribute to the popularity of Blockchain**

The following factors contribute immensely to the popularity of blockchain:

**Trust**: Blockchain helps in creating applications that are decentralized and collectively owned by multiple people. Nobody within this group has the power to change or delete previous transactions. Even if someone tries to do so, it will not be accepted by other stakeholders.

**Autonomy**: There is no single owner for blockchain-based applications. No one controls the blockchain, but everyone participates in its activities. This helps in creating solutions that cannot be manipulated or induce corruption.

**Intermediaries**: Blockchain-based applications can help remove the intermediaries from existing processes. Generally, there is a central body, such as vehicle registration, license issuing, and so on, that acts as registrar for registering vehicles as well as issuing driver licenses. Without blockchain-based systems, there is no central body and if a license is issued or vehicle is registered after a blockchain mining process, that will remain a fact for an epoch time-period without the need of any central authority vouching for it.

**Fundamental Blockchain programming Terminologies**

**Hash Function**

A hash function transforms input string of any size into a fixed-length string output (called as the hash) such as 32-bit or 64-bit or 128-bit or 256-bit depending on what hash function is used. Hash Functions can be defined as atype of encryption that doesn’t make use of keys. It uses a cipher to generate a hash value of a fixed length from the plaintext. It is nearly impossible for the contents of plain text to be recovered from the ciphertext.

**Hashing** is the process of transforming any input data into fixed length random character data, and it is not possible to regenerate or identify the original data from the resultant string data. Hashes are also known as fingerprint of input data. It is next to impossible to derive input data based on its hash value. Hashing ensures that even a slight change in input data will completely change the output data, and no one can ascertain the change in the original data. Another important property of hashing is that no matter the size of input string data, the length of its output is always fixed. For example, using the SHA256 hashing algorithm and function with any length of input will always generate 256-bit output data. This can especially become useful when large amounts of data can be stored as 256-bit output data. Ethereum uses the hashing technique quite extensively. It hashes every transaction, hashes the hash of two transactions at a time, and ultimately generates a single root transaction hash for every transaction within a block. In nutshell the hash is the cryptographic by-product of the hash algorithm which is a one-way function, meaning it can’t be possible to reverse output into inputs again. The algorithm also produces one unique output. These properties are very essential for instance in cryptocurrency Bitcoin where it is used in its consensus mechanism.



Figure VII: Hash Function

**The Concept of Cryptography**

**Cryptography** is the science of converting plain simple text into secret, hidden, meaningful text, and vice-versa. It also helps in transmitting and storing data that cannot be easily deciphered using owned keys.

There are the following two types of cryptography in computing:

Symmetric

Asymmetric

**Symmetric encryption and decryption**

**Symmetric cryptography** refers to the process of using a single key for both encryption and decryption. It means the same key should be available to multiple people if they want to exchange messages using this form of cryptography.

**Asymmetric encryption and decryption**

**Asymmetric cryptography** refers to the process of using two keys for encryption and decryption. Any key can be used for encryption and decryption. Message encryption with a public key can be decrypted using a private key and messages encrypted by a private key can be decrypted using a public key. Let's understand this with the help of an example. Tom uses Alice's public key to encrypt messages and sends it to Alice. Alice can use her private key to decrypt the message and extract contents out of it. Messages encrypted with Alice's public key can only be decrypted by Alice as only she holds her private key and no one else. This is the general use case of asymmetric keys. There is another use which we will see while discussing digital signatures.

The key pair generated by this algorithm consists of a private key and a unique public key that is generated using the same algorithm. It is also called Public-Key Cryptography.

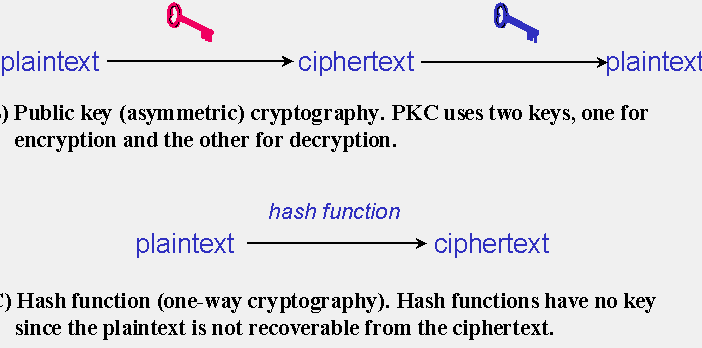


Figure VIII:Cryptography in blockchain

**Application of Asymmetric cryptography to Digital Signature**

**Digital signatures**

Earlier, we discussed cryptography using asymmetric keys. One of the important cases for using asymmetric keys is in the creation and verification of a digital signature. Digital signatures are very similar to a signature done by an individual on a piece of paper. Similar to a paper signature, a digital signature helps in identifying an individual. It also helps in ensuring that messages are not tampered with in transit. Let's understand digital signatures with the help of an example.

Alice wants to send a message to Tom. How can Tom identify and ensure that the message has come from Alice only and that the message has not been changed or tampered with in transit? Instead of sending a raw message/transaction, Alice creates a hash of the entire payload and encrypts the hash with her private key. She appends the resultant digital signature to the hash and transmits it to Tom. When the transaction reaches Tom, he extracts the digital signature and decrypts it using Alice's public key to find the original hash. He also extracts the original hash from the rest of the message and compares both the hashes. If the hashes match, it means that it actually originated from Alice and that it has not been tampered with.

Digital signatures are used to sign transaction data by the owner of the asset or cryptocurrency, such as Ether.

**Programming Languages for Blockchain Development**

**1. Solidity**

Influenced by JavaScript, Powershell, and C++, Solidity is the first blockchain programming language that one must learn. Especially when you want to [develop dApps](https://appinventiv.com/blog/how-to-make-money-on-your-dapp/) or are looking to get into the ICO development game.

The Solidity blockchain programming language was developed by [Vitalik Buterin](https://www.linkedin.com/in/vitalik-buterin-267a7450/?originalSubdomain=ca" \t "_blank), the mastermind behind Ethereum. It serves [blockchain development firms](https://appinventiv.com/blockchain-application-development/) with a myriad of benefits, such as:-

* Developer-friendliness,
* Accessibility to JavaScript infrastructures, debuggers, and other tools,
* Statically typed programming,
* Possibility of inheritance properties in smart contracts,
* Precise accuracy

**2. Java**

Java, the official language of [Android mobile app development](https://appinventiv.com/android-application-development/) and a preferred option for [backend development](https://appinventiv.com/blog/quick-guide-mobile-app-backend-development/), is also considered one of the top blockchain programming languages used for Blockchain development.

The language is derived from C-syntax and is widely chosen for building sophisticated smart contracts and dApps because of its following properties: -

* Robust support for OOP (Object-Oriented Programming) methodology,
* Ease of memory cleaning
* Availability of ample libraries
* As Java is static, so it makes the maintenance of code a piece of cake for anyone.
* Code portability and flexibility is possible by using the Java Virtual Machine.
* Availability of a large number of Java libraries offers a pluggable architecture to the developers.
* Java is a recommended programming language for Ethereum blockchain which involves cryptocurrencies.

Some of the best examples of Blockchain solutions developed using Java are NEM, IOTA, NEO, and Hyperledger Fabric.

**3. Python**

Blockchain programming in Python has not only ruled the world of app development, [IoT app development](https://appinventiv.com/iot-app-development/), and network servers’ development, but is also proving to be an asset in the Blockchain-as-a-service arena.

The language, created in 1991, is widely used for dApps and [Smart Contracts development](https://appinventiv.com/blog/smart-contract-guide/) because of the exceptional features it offers. Some of those features of blockchain coding in Python include:

* Blockchain programming in Python language is easy to learn
* Access to dynamic architecture
* Perfect for both base and scripting approaches,
* Open-source support
* Blockchain coding in python is efficient for prototyping

Steem, Hyperledger Fabric, and NEO are a few popular Python–based Blockchain projects that are prevailing in the industry.

**4. JavaScript**

JavaScript is the most known artificial language nowadays. Despite the issues that triggered the creation of add-ons like CoffeeScript, Flow, and matter, JS continues to be the foremost asked for in several areas.

It is widely glorious for adding interactive parts to net applications and browsers. Most front-end developers like JS. However, the Node.js runtime provides considerably additional options than simply front-end development

JavaScript is the widely liked language, in step with a Stackoverflow survey. A similar survey showed that sixty-six of respondents prefer to work with JS. 17.8% of respondents need to master JS.

Considered for a wide range of app and game development needs, JavaScript is also one of the best Blockchain programming languages to keep an eye on.

The language, in the form of frameworks like Node.js framework, offers innumerable benefits to developers benefits, such as:

* Blockchain programming with javascript enables easier and earlier entry to market
* Blockchain programming with javascript enhances scalability
* Availability of multiple Blockchain [JavaScript frameworks](https://appinventiv.com/blog/javascript-framework-guide/)
* No hassle of integration of respective resources, and more

**5. C++**

C++ is very popular in the technology world and this is true for [Blockchain technology](https://mobcoder.com/blog/blockchain-technologies/) also. C++ is popular for blockchain because of its versatile inclinations like move semantics, primitive control over memory, advanced multi-threading, and other object-oriented features like function overloading, runtime polymorphism, etc.

These varied OOPS peculiarities of C++ enable Blockchain developers to effortlessly mold data and functions together in a single module, just like Blockchain molds the blocks with cryptographic chains. C++ is also popular in Bitcoin (Blockchain technology), a cryptocurrency. The first-ever implementation of Blockchain was originally written in C++.

**6. Simplicity**

Created by Russell O’ Connor, Simplicity may be a high-level Blockchain cryptography language that hit the market in November 2017.

The Simplicity programming language is predicated on Hedera helix and works with a Haskell-like syntax that makes cryptography easier and effective. Besides, it’s extremely mathematical in nature and makes the code line human-readable. attributable to that, it’s extremely used for developing good Contracts and blockchain solutions that job with each Bitcoin and Ethereum Virtual Machine (EVM).

7. Golang

Go has been designed and developed at Google by Henry M. Robert Griesemer, Rob Pike, and Ken Thompson. The language is very versatile and straightforward to be told, that makes it compatible with Blockchain, Big Data, Machine Learning, etc. Offering high quantifiability and optimum speed performance. Go may be a statically typewritten programing language and prove its value within the following ways:

* It is syntactically the same as C, which suggests veteran Blockchain developers will simply comprehend the code.
* It comes with memory safety which suggests the app is safe from software bugs once accessing the memory.
* Its readability, additionally as usability, is analogous to Python and Java, that ends up in high quantifiability and higher speed performance.

**Coding a P2P blockchain in Python**

**Creating the Block class**

Import [hashlib](https://docs.python.org/3/library/hashlib.html" \t "_blank), a module that lets us create one-way encrypted messages. Cryptography techniques like hashing make Blockchain create secure transactions.

A hash function is an algorithm that takes some data (usually an encoded string) and returns a unique identifier, often named “digest” or “signature.” This last part is vital; with a hash function, a slight difference in the input produces a radically different identifier as an output. We’ll see this in action later on.

For now, just import the built-in module hashlib:

import hashlib

This module includes most of the hashing algorithms you’ll need. Just keep in mind we’ll be using the **hashlib.sha256()** function.

class CoinBlock:

def \_\_init\_\_(self, previous\_block\_hash, transaction\_list):

self.previous\_block\_hash = previous\_block\_hash

self.transaction\_list = transaction\_list

self.block\_data = f"{' - '.join(transaction\_list)} - {previous\_block\_hash}"

self.block\_hash = hashlib.sha256(self.block\_data.encode()).hexdigest()

**GeekCoinBlock Class Explanation**

First, we create a class named **CoinBlock**, a wrapper for objects that will have certain characteristics (attributes) and behaviors (methods).

Then we define the **\_\_init\_\_** method (also named constructor), which gets invoked each time a GeekCoinBlock object gets created.

This method has three parameters:

* **self** (the instance of each object)
* **previous\_block\_hash** (a reference to the previous block)
* **transaction\_list** (a list of transactions made in the current block).

We store the previous hash and transaction list and create an instance variable **block\_data** as a string. This doesn’t happen with real cryptocurrencies, in which we store that kind of data as another hash, but for simplicity purposes, we’ll store every block of data as a string.

Finally, we create the **block\_hash**, which other blocks will use to continue the chain. Here’s where hashlib comes in handy; instead of creating a custom hash function, we can use the pre-built **sha256** to make immutable blocks.

This function receives encoded strings (or bytes) as parameters. That’s why we’re using the **block\_data.encode()** method. After that, we call **hexdigest()**to return the encoded data into hexadecimal format.

**Using our Block Class**

Let’s use our Block class to create a chain of blocks (Blockchain).

In the same file, create a couple of transactions made up of simple strings stored in variables, for example:

class CoinBlock:

...

t1 = "Nancy sends 8 GC to May"

t2 = "Mandy sends 6.3 GC to James"

t3 = "Jay sends 5.2 GC to Alisson"

t4 = "Alice sends 2.1 GC to Noah"

Now, build the first block of our Blockchain by using the CoinBlock class and print its attributes. Take into account that the **previous\_hash** parameter of the genesis block (first block that precedes other blocks) will always be some arbitrary string or hash, in this case, “firstblock.”

block1 = GeekCoinBlock('firstblock', [t1, t2])

print(f"Block 1 data: {block1.block\_data}")

print(f"Block 1 hash: {block1.block\_hash}")

Then, we do the same with the second block, but passing the first block hash as the **previous\_hash**argument.

block2 = CoinBlock(block1.block\_hash, [t3, t4])

print(f"Block 2 data: {block2.block\_data}")

print(f"Block 2 hash: {block2.block\_hash}")

Note that if we run the program at this point and try to change the value of coins that Noah sent to 8 it will generate a different hash function

**The complete Python code Explained above**

import hashlib

class OlaCoinBlock:

def \_\_init\_\_(self, previous\_block\_hash, transaction\_list):

self.previous\_block\_hash = previous\_block\_hash

self.transaction\_list = transaction\_list

self.block\_data = f"{' - '.join(transaction\_list)} - {previous\_block\_hash}"

self.block\_hash = hashlib.sha256(self.block\_data.encode()).hexdigest()

t1 = "George sends 9.1 GC to Joe"

t2 = "Joe sends 2.5 GC to Adam"

t3 = "Adam sends 1.2 GC to Bob"

t4 = "Bob sends 0.5 GC to Charlie"

t5 = "Charlie sends 0.2 GC to David"

t6 = "David sends 0.1 GC to Eric"

block1 = OlaCoinBlock('firstblock', [t1, t2])

print(f"Block 1 data: {block1.block\_data}")

print(f"Block 1 hash: {block1.block\_hash}")

block2 = OlaCoinBlock(block1.block\_hash, [t3, t4])

print(f"Block 2 data: {block2.block\_data}")

print(f"Block 2 hash: {block2.block\_hash}")

block3 = OlaCoinBlock(block2.block\_hash, [t5, t6])

print(f"Block 3 data: {block3.block\_data}")

print(f"Block 3 hash: {block3.block\_hash}")

**Conclusion**

Blockchain introduces a distributed ledger that can be shared across networked devices. Individuals on the network can share files and values such as cryptocurrencies securely, on a peer-to-peer basis without the need for middlemen. This means reduced interruptions and there is no single point of failure, there is high reliability in the network. Due to cryptography, all assets are secured with high security.

The most important aspects of blockchain are its security, ensured by cryptography; scalability where the network should accommodate millions of users without compromising on security and reliability; and decentralization, which means control and governance must be achieved by all individuals on the network and not a select few.

The rules by which individuals agree on transactions and the creation of the chain is called consensus algorithm or mechanisms. The basis of these mechanisms is Proof of Work where individuals agree on what and when transactions go through or get processed, based on the amount of computer processing power they contribute. Blockchain technology has kept growing.

New consensus algorithms are more than 10 and keep on being innovated to ensure networks are scalable, more secure, and more decentralized.

**References**

[1] Toshendra, K. S. (2021) Blockchain and Role Of P2P Network: Retrieved from <https://www.blockchain-council.org/blockchain/blockchain-role-of-p2p-network/>

[2] V. Buterin,(2021) “On Public and Private Blockchains. Ethereum Blog”, Crypto Renaissance Salon. Internet: https://blog.ethereum.org/2021/08/07/on-public-and-private- blockchains/

[2] Crosby, Nachiappan, Pattanayak, Verma & Kalyanaraman, “Blockchain Technology: Beyond Bitcoin”, Applied Innovation Review. Issue No. 2, June 2016, Pantas and Ting

Sutardja Center for Entrepeneurship & Technology, Berkeley Engineering. Available: https://j2-capital.com/wp-content/uploads/2017/11/AIR-2016-Blockchain.pdf

[3] Sultan, K. Ruhi, U. Lakhani, R. (2018) “Conceptualizing Blockchains: Characteristics & Applications”, 11th IADIS International Conference Information Systems.

Available: https://arxiv.org/ftp/arxiv/papers/1806/1806.03693.pdf

[4] Coletti, P. (2019) “Bitcoin's baby: Blockchain's 'tamper-proof' revolution”, BBC News, Internet: https://www.bbc.com/news/technology-32781244, May, 2015 [Mar. 15, 2019].

[5] Alharby, Maher & van Moorsel, Aad. (2017). Blockchain Based Smart Contracts : A Systematic Mapping Study, pp. 125-140. Available:

https://www.researchgate.net/publication/319603816\_Blockchain\_Based\_Smart\_Contra cts\_A\_Systematic\_Mapping\_Study

[6] Christidis, K. and Devetsikiotis, M. (2016) "Blockchains and Smart Contracts for the Internet of Things," in IEEE Access, vol. 4, pp. 2292-2303.

Available:http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7467408&isnumb er=7419931

[7] Alharby, M. and van Moorsel, A., (2017). Blockchain-based smart contracts: A systematic mapping study. arXiv preprint arXiv:1710.06372. Available: <https://arxiv.org/ftp/arxiv/papers/1710/1710.06372.pdf>

[8] Nakamoto, S. (2008) Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf