POLITECNICO DI MILANO

Dipartimento di Elettronica, Informazione e Bioingegneria

Designing a blockchain platform for IoT forensics evidence preservation

Supervisor: Prof. Alessandro Enrico

Candidate:

Aida Rezaeepoor

Matricola: 942106

Academic year 2022-2023

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor at Politecnico di Milano, Prof. Alessandro Enrico Cesare Redondi, for guiding me throughout this work. He helped me to progress during these months. Obviously, His patience, enthusiasm and collaboration made this research possible.

I would like to thank my family because of their financial and emotional support to motivate me in every step of this process to make progress unconditionally. Additional thanks also go to my friends as they encouraged me continuously during this project with their positive characteristics and attitudes.

I also would like to thanks all our respected other professors in the Telecommunication engineering department and all the others members of the departments who assisted me through this thesis directly or indirectly and shared with me new ideas to improve.

In the end, I strongly believe that Politecnico di milano improved my skills to be a successful person in whatever I decide to foster. I believe that constant research in this area will revolutionize the world of digital evolution.

Abstract

The Internet of Things is growing steadily, with millions of devices that make everyday life easier. Such a tough relationship between humans and their smart devices, turning them into digital witnesses of our daily lives through their sensors. This opens up a new field of digital investigation called IoT forensics. Feature-Sniffer is an add-on for OpenWrt-based access points and allows to easily perform online traffic feature extraction, which avoids to store large PCAP files. There is a file which presents Feature-Sniffer with an accurate description of details, and we show its possible uses with practical examples for device identification and activity classification from encrypted traffic produced by IoT cameras.

Today, many Internet of Things (IoT) devices have become an integral part of our lives and are revolutionizing the world. Technological developments in the Internet of Things (IoT) have spawned multiple forms of cyber-attacks that exploit the heterogeneity of IoT networks. Forensics collects, stores, and analyzes data of IoT devices. Digital evidence capture differs from physical evidence due to its fragile electronic nature. Therefore, vulnerabilities such as tempering should be prevented to ensure IoT evidence integrity [1].

This tamper-proof functionality can be provided using a hash code for the data. Traditional databases have failed to provide adequate security measures and solutions to vulnerabilities. IoT applications that operate on the basis of centralized systems must rely on central instances. Such systems fail to provide data immutability, data traceability, and transparency. Blockchain technology is a fully decentralized system that eliminates trusted third parties (central authorities). A feature of this technology is that once data is entered, it cannot be changed or removed from the system. Unlike centralized systems, blockchain offers data traceability and transparency. State-of-the-art blockchain technology has shown promising performance in building applications in various areas of life, from cryptocurrencies to smart contracts, consensus, and decentralized applications. The characteristics of blockchain technology can attract new users to integrate the IoT data entered into blockchain technology. This work describes the design and implementation of a blockchain-based decentralized IoT

framework that can address the challenges associated with developing such frameworks while taking advantage of blockchain's inherent security features. This decentralized IoT framework uses blockchain in combination with other peer-to-peer mechanisms and aims to provide: Access control, Secure IoT data transmission, A peer-to-peer data sharing business model, Secure end-to-end IoT communications without relying on a central intermediary for authentication or data processing.

A smart contract is a digital program that can be read by any participant and dynamically executed in response to events on the blockchain. Additionally, smart contracts allow registered accounts to interact with evidence. This paper contains proofs containing numerical features of csv files extracted from traffic generated by some of IoT devices. One of the main parts of the contributions is to suggest a possible structure for the desired evidences to be inserted in the blockchain with Ethereum platform. In this use case, we learned which consensus mechanism can be more suited to be implemented in the blockchain. With all possible applications of blockchain-based frameworks within IoT, this work takes a step towards the goal of a trustworthy and decentralized IoT blockchain.

KEYWORDS:

IOT, blockchain, SHA256, Consensus algorithm, Python

Contents

Parole chiave: IOT, blockchain, SHA256, Consensus algorithm, Python

Abbreviations

IoT: Internet of Things

P2P: peer to peer

DB: Data Base

SHA: Secure Hash Algorithm

NSA: National Security Agency

CSPs: Cloud service providers

SC: Smart Contract

POW: Prove Of Work

PoS: Proof of stake

Sec: Second

List of figures

List of tables

Chapter 1

1. Introduction

1.1. Context and motivation

In current Internet of Things (IoT) architectures, users implicitly trust third-party or hidden services to process data collected from IoT devices and issue security certificates. The Internet of Things has moved from a visionary concept to a reality, revolutionizing our lives. Thanks to innovations in low-power communication technologies, sensor systems, energy-efficient microcontrollers, and battery devices, the number of intelligent, connected devices surrounding us is growing every day. Among the many application scenarios of IoT, smart buildings and smart homes are the most ones which are researched. Buildings and homes rich in energy resources (electricity grids) and connectivity (mainly WiFi) make perfect playgrounds for all kinds of the IoT prototypes, even considering that's where people spend most of their time. This strict coexistence of humans and intelligent devices has two effects. B. Focus on sabotage of home/building security systems or identity theft from IoT devices. On the one hand, such devices witness our daily life through sensors. For these reasons, the term "IoT forensics" has been coined in recent years to mean activities related to collecting sensitive information from IoT devices, whether compromised or not.[2]

IoT forensics is different and more sophisticated than traditional digital forensics. First, IoT devices typically have limited or no persistent memory to analyze during investigations. The main value available for forensic analysis is therefore the network traffic they generate and exchange with other entities (such as cloud services). Therefore, it is imperative to set up a system that can easily capture, store, and analyze such network traffic as close as possible to its source.

For example, IoT traffic from multiple devices connected to an access point using NAT technology (as it is often the case) may be of difficult analysis when observed from the external, since all devices appear with the same source MAC and IP addresses. This makes

it difficult to perform per-device, feature-based traffic analysis, which is the de-facto approach in case of encrypted traffic such as the one produced by current IoT devices.[3]
At the same time, storing the complete network packets produced by IoT devices (e.g., in the form of PCAP files) may be not viable in the long run, due to the large storage space required. Over the past decade, there has been a shift from local desktop applications to remote web services that store data remotely. In fact, many security threats have emerged aimed at exploiting and compromising these key trust points [4] . These attacks were seen not only in the IoT through botnets [5], [6] , but also the Internet at large through adversarial attacks on centralized servers [7]. The data collected by IoT devices contains confidential and private information, therefore there are some privacy implications in the event of a security leak within centralized services. Blockchain technology is suitable for data exchange and is used as a platform for IoT devices. Hash keys created on a blockchain can be used to store large amounts of data, stored in the form of Merkle trees. Hash of Evidence not only requires less storage space to hold data, but also preserves the integrity, tamper-proof and security of evidence data. When inserting data into the blockchain, there are structural advantages such as the following. Blockchain brings immutability as data cannot be changed or deleted. In addition, it brings traceability as every kind of action is recorded for ever [8] , [9] . Blockchain also creates transparency as all recorded data is visible to network participants. Therefore, considering these advantages, blockchain is deployed as a secure and tamper-proof platform. Transactions in blockchain are performed to prove security with minimized overhead. The recorded data on the blockchain can neither be deleted nor modified. A homomorphic computation is performed to protect the data by encryption. Blockchain technology was developed to manage and store data controlled by IoT devices. Based on transaction selection, new blocks are created by miners. In our proposed work, the miner is the owner of her IoT device. Only registered users can access blocks of transactions aggregated on the blockchain. To identify a block, it is important to have a unique and trusted signature as proof of work to complete the authentication challenge. Therefore, the public key and signature of individual devices play an important role in solving security problems.
To participate in a cryptocurrency network, each node must have its own copy of the blockchain, which is synchronized with other participants [10] . It is clear that all

cryptocurrencies must provide some way to protect the blockchain from attacks. For example, an attacker can reverse a spending transaction by spending some money and then sending their own version of the blockchain that doesn't contain the transaction. Because the network is distributed, users do not know which version of the ledger is active. Proof-of-Work algorithms provide network security in the form of block mining. The main point of PoW is that each node that wants to participate in mining must solve a computationally difficult problem to ensure the validity of the new mined blocks. Each new block gives the miner a certain amount of coins. This protocol is fair in the sense that miners using 1/p of total computing power can create blocks with probability p. An attacker must solve the same task as any other participant in a PoW-protected network.

1.2. Research objective

Forensics is a criminal investigation to record criminal activity. I would like to know if there are any hackers trying to manipulate the data in the blockchain. Therefore, one of the goals of this work is to build a blockchain platform that is secure against various types of attacks.

In the last years, with the increase of interest in IoT and in Digital Forensics, a high number of works focused on the study of IoT devices behavior as possible evidence for forensics investigations. In particular the literature on IoT Forensics can be roughly divided into three categories: IoT devices identification, real-life event detection, IoT forensics framework.

In this work, we have the data from IoT devices which includes the actions from humans who live near IoT devices. By studying this traffic, we can understand which action the person is doing. We want to store this data into a safe data base in order to be secure enough. Then whenever we need this data (for example we want to know the actions of a user in a specified timestamp), by retrieving the information, we can access to each data of a user. We aim to incorporate with Ethereum blockchain among several forensic architectures, where security is controlled in IoT devices. Our study focuses on proof-of-work (PoW) consensus protocols, which is the dominant choice in existing digital currencies. Security is a major topic of data protection aimed at ensuring security from the moment of data entry. Indeed, designing a specific architecture for storing IoT data can defeat many types of attackers. This paper contribution is based on several key issues. From a security perspective, our goal is to address some challenges of blockchain, such as privacy and storage capacity. Security with blockchain is the best whereas using a third party for the purpose of authentication. The goal of this paper was to design a novel decentralized framework based on the blockchain data structures that can be used to inject real-time traffic information from IoT devices collected in the form of transactions at the blockchain.

At first, we extracted some data as evidence from IoT devices in several rows in excel file. Each row has its own specific timestamp and the specific IoT device which is identified by its mac address. Evidences are represented by numerical features extracted from the traffic which is produced by each of the IoT devices. In this thesis, we aim to look forward to find a possible structure to insert and store our evidence in the Ethereum blockchain platform.

1.3. Research structure

The following parts represent the structure of this work.
Chapter 1 (introduction) briefly refers to the IoT forensics, data storage, retrieving data from IoT devices and new look towards new security issues involved in sensitive data.
Chapter 2 starts off understanding the concept of blockchain technology, digital forensics and existing blockchain platforms. Furthermore, the consensus mechanisms will be explained as it will be used as validation item to let block be added to blockchain.
Chapter 3, has taken theory into implementation, and have demonstrated a realworld implementation of blockchain for the data frame (.csv file) with Python programming language with considering traceability, immutability, transparency and security along with performance metrics we observed in our implementation.
In Chapter 4, we will see the results and comparisons.
Finally**,** In Chapter 5 we conclude with the complete implementation, results, and future work discussed about.

Chapter 2

2. Background and related works

This chapter provides a detailed overview of the current developments and trends in blockchain for IoT evidences with definition of consensuses and smart contracts.

2.1. Blockchain history

Blockchain was first introduced by Satoshi Nakamoto when he created Bitcoin [11] . As a specific type of database, blockchain differs from a typical database in the way it stores information. Bitcoin was the first invention built on blockchain technology and was the first digital cryptocurrency to solve the double-spending problem without the need for a trusted intermediary or central server.

2.2. Block chain definition

In general, blockchain is a distributed database. This means that the central authority cannot have full control over the database or change its data without traceability. Additionally, the database is distributed. In other words, each node on the blockchain network holds a complete copy of the database. Blockchain databases are called "immutable" because due to the decentralization of databases, there is no authority to change or delete data. Therefore, once data is added to the database, it cannot be removed or changed later. The only functions allowed are data updates and additions, and these rules are enforced by the consensus protocol. Blockchain-based systems are the product of cryptography, public key infrastructure and economic modelling, applied upon peer-to-peer networking and distributed consensus to achieve distributed database synchronization [12] , [13] . The blockchain is a distributed data structure, and is dubbed a distributed "ledger" in its utility of recording transactions occurring within a network [14] . In cryptocurrencies, applications have a record of the entire blockchain. Additionally, distributed ledgers (ledgers for implementing blockchain) could be used in networks where any form of data exchange takes place. In a blockchain-based peer-to-peer network, all participating peers maintain an identical copy of the ledger. A canonical shared state of the blockchain is maintained through decentralized consensus among the peers.

Blockchain-based systems advantages

Some of the advantages of blockchain-based systems are as described [15] :

- **Decentralization:** In the blockchain ecological environment, data are distributed and stored and linked by a cooperative mechanism. Such data is lightweight and secure.
- **No tampering:** Blockchain adopts some different encryption methods to track and manage the whole data in the life cycle. In this case, data can reach the specific consensus by each other and can be not dependent of each other.
- **Openness, transparency and traceability:** According to various business needs, blockchain fully records each data operation in the chain and maps the operation record to the information before and after the data operation. At the same time, blockchain provides a public interface for data queries, and requesters perform relevant data interaction work according to the established consensus which was reached before.

2.3. Hash function

It is a function that maps data x to a fingerprint y = h(x) of the data [16] . Fingerprints are usually fixed length and shorter than the original data. A fingerprint represents the data that creates it, just as a human fingerprint represents the person who created it. Hash function is a one-way function as shown in the figure 1. A one-way hash function is a function that is easy to compute but difficult to reverse. For example, given a one-way hash function and a fingerprint, it's virtually impossible to identify the hashed data using the fingerprint. Although unrealistic, it's commonly understood that the best strategy we know is to brute-force search through all possible inputs when looking for a particular output. The Bitcoin blockchain uses the one-way hash function called SHA256 (Secure Hash Algorithm) [17] ,

[18].

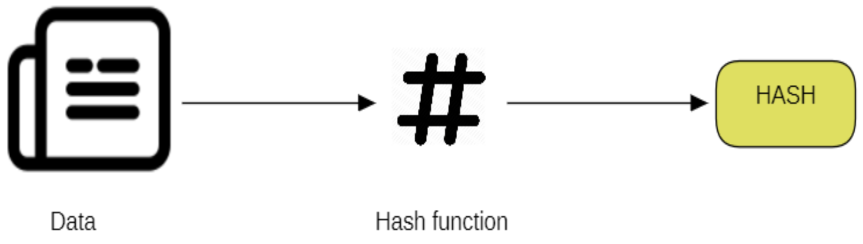


Figure 1 Hash Function

The purpose of a hash function is to generate a short and fixed-length hash value (also called a message digest) from an input message of arbitrary length. A hash digest must be a unique representation of the message, also known as a message fingerprint [19] , [20]. The hash function can be represented as the following formula:

$$H = h(m)$$

Where *h()* is the hash function, *m* is the message and *H* is the hash value.

As the hash functions are always one-way functions, which the hash value doesn't retrieve the original message, no inverse function ($h(H)^{-1}$) for this operation can be found. One-way also means that, unlike encryption primitives, functions do not use private keys to generate hashes (AES uses secure keys to encrypt and decrypt messages). A hash value is a unique, short, fixed-length representation of a message. Modern hash functions exceed this length by 128 bits. Most blockchain technologies use hash functions that perform message digests that are 256 bits long. The message digest length is an important factor in choosing a hash function. The more bits used in the message digest, the more resistant it is to collision attacks. A collision attack is finding a message that matches a hash value. To determine successfully the message which corresponds to the hash value of n bits, the attacker has to hash $2^{n/2}$

messages [21]. Cryptographic hash functions are more sensitive if the messages they hash are different. The difference in bits results in completely different hash values. An example of hash (SHA256) production is shown below in the situation in whuch just one bit has changed in the input text. Just the letter “c” and “d” have been changed in the ASCII code.

h("abc") = ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad

h("abd") = a52d159f262b2c6ddb724a61840befc36eb30c88877a4030b65cbe86298449c9

Storing hashes on the blockchain

One way to receive the benefits of blockchain without paying a fortune for transactions is to store hashes of evidence on the blockchain due to the fact that the original size of the evidence is too large. In this case the hash can be used to check if the data has changed. All we store on the blockchain is the hash of the evidence. Compared to our data, hashes are very small, so the transaction costs are relatively low. The raw data can be optionally saved in any way we want.

2.4. Merkle tree

A Merkle tree refers to a binary tree structure that can summarize content and audit content in large datasets efficiently and safely, as shown in Figure 2. A Merkle tree holds a complete copy of all transactions that have ever occurred on the blockchain [22].

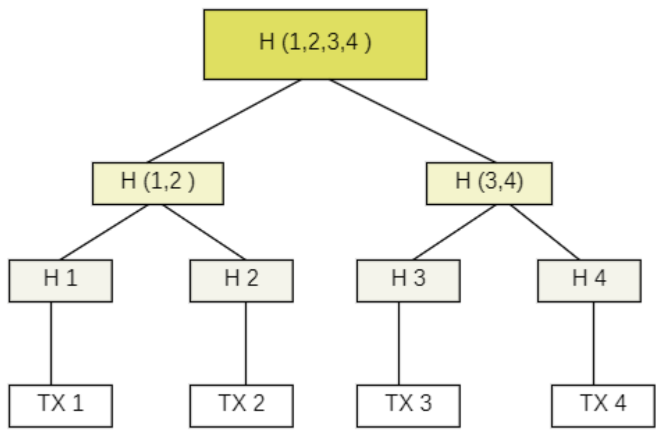


Figure 2 Merkle tree structure

A Merkle tree summarizes all the transactions in a block by generating a digital fingerprint of the entire collection of transactions, allowing users to see if a transaction is included in a block. When one transaction changes or changes, so does the root of the Merkle tree. A field in the block's header contains the root of the Merkle tree that is generated when the block is created. A Merkle tree is created by repeatedly hashing pairs of nodes until only one hash remains. This hash is called the root hash or the root of the Merkle tree. As shown in Figure 3, each leaf node contains a hash of transactions of data and each non-leaf node contains a hash of previous hashes.

HT1 = Hash (Transactional_Evidence#1)

HT2= Hash (Transacrional_Evidence#2)

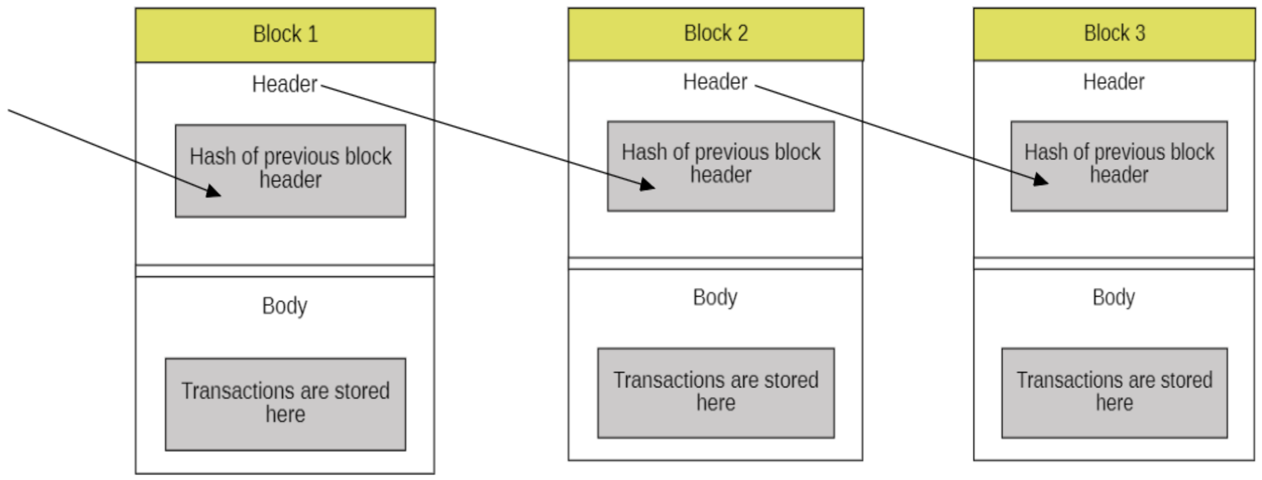
HT1,2 = Hash (HT1|HT2)

H Root = Hash (H12|……)

Figure 3 Illustrates Merkle tree of nodes

2.5. Blockchain structure

A blockchain is a decentralised ledger of all transactions on a peer-to-peer network, containing previous block hashes, versions, timestamps, nonces, bits, and Merkle roots. Each block body contains a list of transactions and a transaction counter, and when a block is accepted by the network, it is sent to the network by the consensus algorithm and included


Block 1
Header
Hash of previous block header
Body
Transactions are stored here
Block 2
Header
Hash of previous block header
Body
Transactions are stored here
Block 3
Header
Hash of previous block header
Body
Transactions are stored here

in the blockchain, so the blockchain is a block containing transaction details. appear in the network as shown in Figure 4.

Figure 4 Physical structure of blocks in blockchain

Transaction information may relate to token transfers or any type of data exchange that takes place over a network. Each block is divided into her two parts, the header, and the body. Transactions are stored in the body of the block. The header of each block contains the identifier of the previous block, so the blocks are connected in a chain like a linked list, as shown in Figure 5.The first block in the chain is called the "genesis" block [23]. The identifier of each block is obtained by taking its cryptographic hash, which is why having each block linked to the previous block helps the blockchain achieve immutability of its contents.

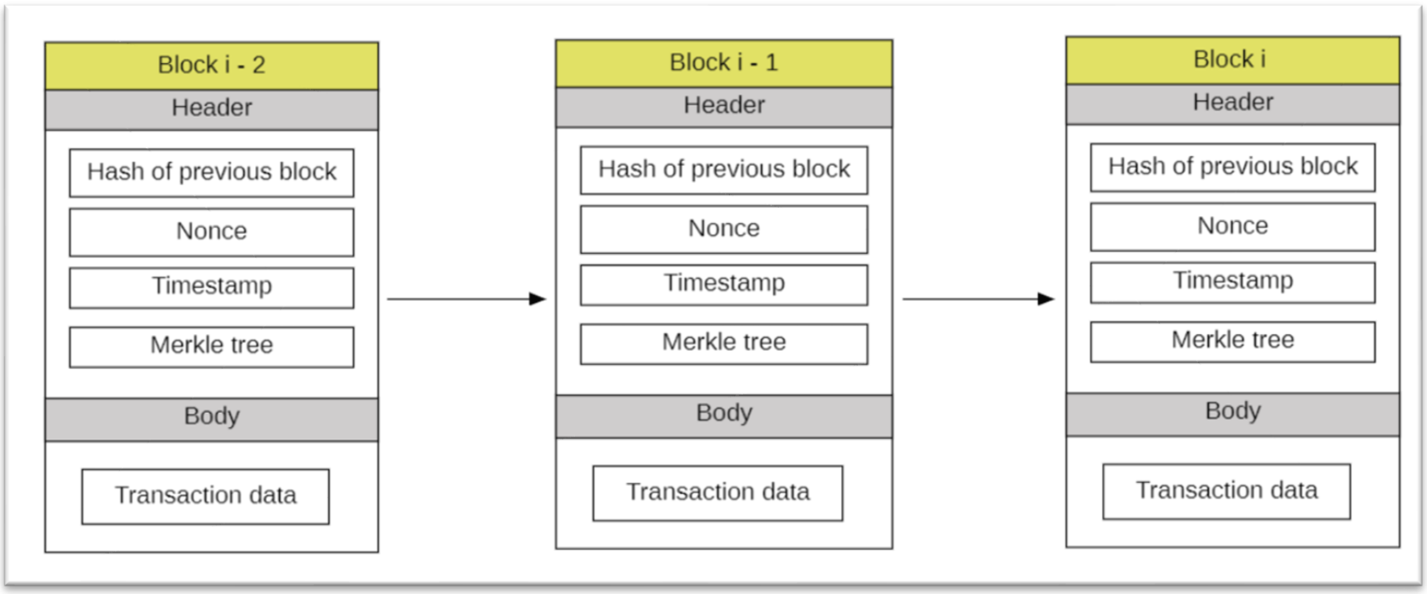


Figure 5 Graphical representation of the block chain

If a hacker were to alter the contents of a past block, its identifier would no longer be valid, and a domino effect would render the parent block hashes in the subsequent blocks invalid as well. Therefore, to successfully alter the contents of a single block, the attacker would have to alter the headers in all successive blocks, and have this alteration take place in the majority of the nodes in the network, so as to have the peers reach consensus on this altered blockchain. Other than the block's own identifier, and the identifier of the previous block, the header contains a timestamp of when the block was published, and the Merkle tree root [24] for all the transactions stored within the body of the block as illustrated in Figure 6. The Merkle tree root significantly reduces the effort required to verify transactions within a block.

The block chain is a linearly growing data structure, with higher transaction activity inflating the sizes of newer blocks. As part of all consensus algorithms, peers verify transactions recorded in a newly published block. The transactions within a block all have a transaction ID, whereby each transaction ID is the cryptographic hash of the corresponding transaction's information stored in the block. The transaction IDs are hashed together in pairs and a hash tree is built within the block, as shown in Figure 5. The root of this tree is stored in the block header. To verify a transaction, a local copy of all the transactions is not required, and verification can be carried out by simply using the Merkle tree branch containing the transaction in question. A tampered transaction would produce altered hashes within its branch and would be detected without much computational effort. In the event of multiple nodes in the blockchain network producing valid blocks at the same time, the block chain can fork, and maintaining a single canonical version of the blockchain becomes an issue. Mainstream block chain networks [14] [25] resolve this issue by only considering the longest fork as canon, and all orphaned forks are discarded. Other fields included in the block header contain information specific to the consensus algorithm used within the blockchain network. The maximum number of transactions that a block can contain depends on the block size and the size of each transaction. The block size can be even 5 MB and contains usually more than 500 transactions.

As shown in Figure 6, the timestamp is to verify the time in which the block is created [26] . A nonce is a counter used for proof of work (POW), and bits are used to record the hash value of the current target. Version is used to update the version number of the blockchain system software. A Merkle tree records a hash value for each transaction. Each transaction is stored in the block body, after which a sha256 operation (for hashing) is performed to encrypt and secure the transaction. A transaction counter records the number of transactions in the block body. To ensure data integrity, the hash value will be completely different if the transaction or block is modified. If the hash value changes, the Merkle root will also change. Therefore, it guarantees the operational security of the blockchain. Blockchain transparency is proven as it can automatically detect all changes.

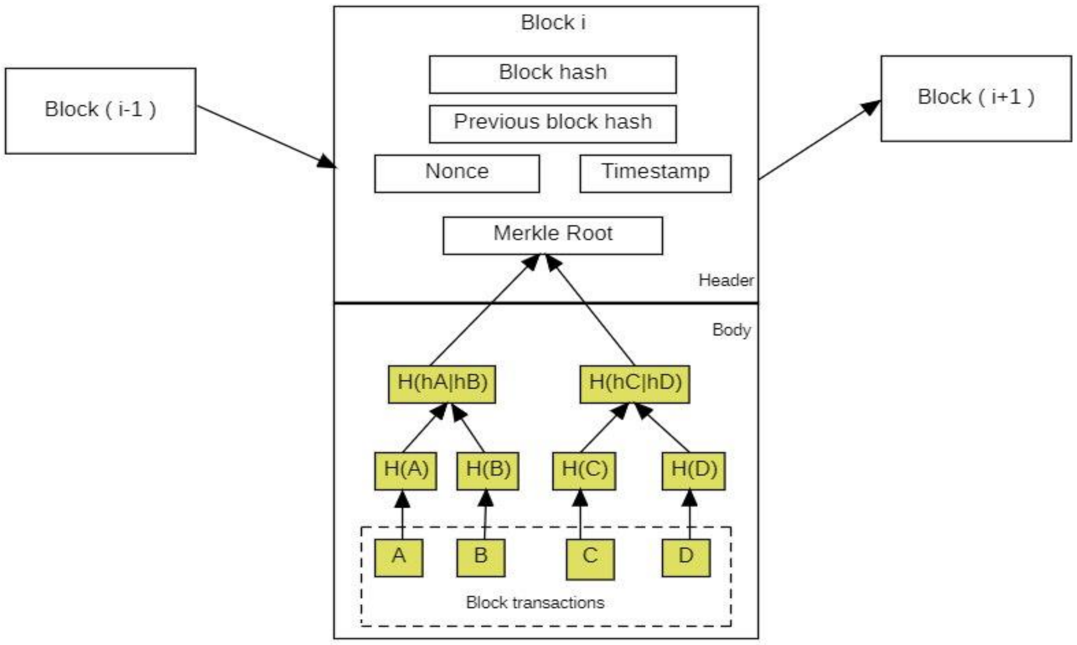


Figure 6 block header with Merkle tree of transactions

2.6. Cryptography

Encryption is defined as a method of protecting sensitive data from unauthorized access. There are various cryptographic techniques that are expressed as part of blockchain security protocols. In a blockchain network, cryptography protects transactions made between two nodes. There are three main themes that are important to blockchain: distributed ledgers, peer-to-peer networks, and cryptographic security. Distributed ledger systems and peer-to-peer networks (P2P) operate securely with robust security technology. Blockchain uses two types of security approaches. One is encryption and the other is hashing. Encryption is used to encrypt messages in P2P networks. Hash functions are intended to ensure the security of block information and link a large number of blocks within a blockchain. Encryption (encoding information) and decryption (decoding encrypted information) are the two main concepts of cryptography.

Cryptography is therefore a security method for protecting gateways or exchanging information between two nodes to protect against third-party intrusion.

2.7. SHA256

SHA stands for Secure Hash Algorithm, a cryptographic hash algorithm used to determine the integrity of certain data. SHA256 is the successor to SHA1. This is because it is now much more resilient to collision attacks, as long hashes that are hard to crack can be generated. Encrypted messages cannot be decrypted by this algorithm, making them very secure. It is also a 256-bit block cipher algorithm. SHA-256 converts a 64-character hexadecimal string to a character length of 256 bits. Hence the name SHA-256.

2.8. Private and public key cryptography

Each peer on the blockchain has a public/private keypair (k_p^r, k_s^r) which is used for addressing, and creating digital signatures on each transaction, for guaranteed non-repudiation. Since these keypairs are not associated to real-life identities, blockchains offer "pseudonymity" to its users [27]. Signed transactions are executed by functions written in deployed smart contracts to transfer cryptocurrency tokens or interact with application binary interfaces (ABIs). Transactions can also be made between two separate blockchains via sidechains. Sidechains [23] are blockchains synchronized with and running in parallel to an existing blockchain, referred to as the "main chain". Tokens can be transferred from the main chain to side chains and vice versa. Side chains use tokens in isolated use case scenarios. Therefore, sidechains augment the functionality of the mainchain and provide a testing ground for blockchain application development.
To further clarify, transactions on the blockchain are signed with a private key to prove ownership of the public address. A private key owner can sign data by applying a cryptographic algorithm to the private key. Everyone else can verify that the owner of the private key really is the owner by unsigning it with the public key. Encryption systems also work in reverse, encrypting data with a public key so that only the owner of the corresponding private key can decrypt the data. The public address works like the address of the user's bitcoin wallet. A compressed public key, called a Bitcoin address, can be shared with anyone and used by others to send funds to that wallet. The corresponding private key can later be used to unlock the assets sent to this bitcoin address [28] [14]. The private key used in Bitcoin is 256 random bits. A public address is then calculated based

on the private key using the Ecliptic Curve Digital Signature Algorithm. The public key is hashed and preceded by two zeros to form the Bitcoin address [28] , [14].

2.9. Peer to peer network

A peer-to-peer network, also known as peer-to-peer, is a decentralized network with a group of devices called nodes that collectively store and share files, with each node acting as a single peer. His P2P network on blockchain runs without central control. This means that all nodes have the same permissions to perform the same tasks [29] , as shown in Figure 7.

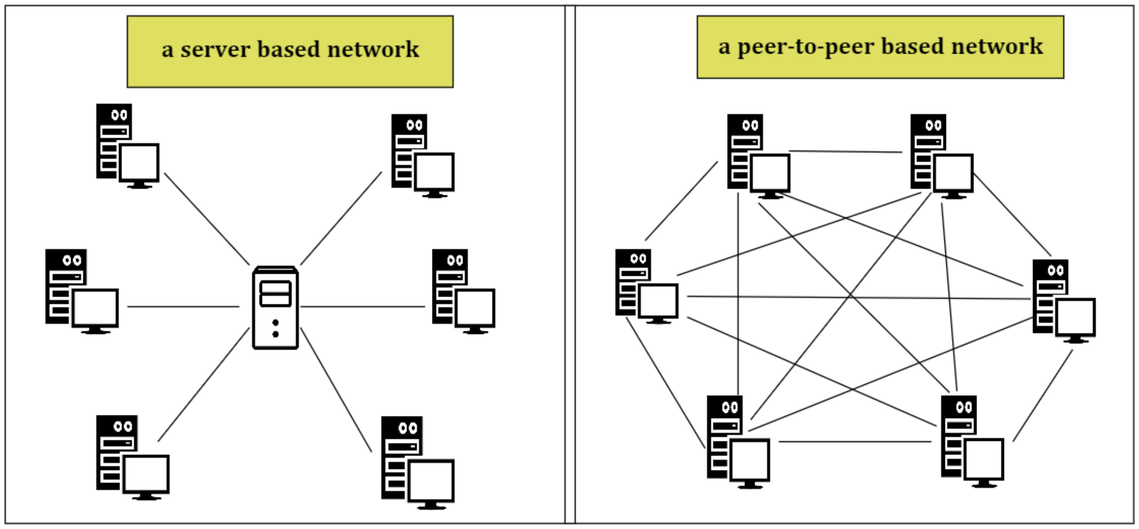


Figure 7 P2P structure

2.10. Smart contracts

The concept of the smart contract [30] was first introduced by Nick Szabo in 1994, which is defined as “a computerized transaction protocol that executes the terms of a contract”. Within the context of blockchain, the smart contract acts as a trusted distributed application that gains its trust from the blockchain and the underlying consensus among the peers. Since they reside on the blockchain, smart contracts have a unique address through which the end user can address a transaction to it. The main benefits of deploying Smart Contracts over a blockchain are that the blockchain guarantees that the contract terms cannot be modified. Blockchain makes it impossible to tamper or hack the contract terms. Thus, smart contracts deployed over a blockchain are expected to bring reduction in costs of verification, execution, arbitration and fraud prevention to overcome the moral hazard problem. Smart contracts can be utilized to perform a variety of functions within a blockchain network, such as:
Allowing ‘multi-signature’ transactions, whereby a transaction requires a specified amount of signatures to be issued [31].

Enabling automated transactions triggered by specific events. This facilitates request-response type transactions, for decentralized data access within a blockchain-based system. A smart contract can also be triggered when a message is sent to the smart contract’s address [32].
Providing utility to other smart contracts. For example, in Ethereum, inheritance can be written into smart contracts, where one contract can invoke functions written in another contract.
Allowing storage space for application-specific information, such as membership records, lists or boolean states.

2.11. Blockchain types

The blockchain approach is a decentralized platform within a peer-to-peer network for sharing data. Blockchains can be categorized as partially decentralized (permissioned blockchains) and fully decentralized as non-permissioned blockchains (permissionless blockchains)[33] . Moreover, as there are different principles such as access control mechanisms and authentication, the blockchain can be divided into 3 main blockchains called a public blockchain, a private blockchain, or a consortium blockchain [34] (see Table 1).

Features	Public Blockchain	Private Blockchain	Consortium Blockchain
Management	Non centralized	Centralized	Partially centralized
Access permission	Reading is public	Public / Restricted	Public / Restricted
Consensus determination	All miners	One organization	Selected set of nodes
Consensus process	Permission-based	Permission-based	Permission-free

Table 1 Comparison of blockchain modes

Public blockchain

Bitcoin and Ethereum are permissionless blockchains. Any entity can participate in the network without permission granting, and entities are anonymous on the network. Each entity does not trust other entities, but views them as enemies. A public blockchain is a permissionless, inherently decentralized, open-source network in which anyone can participate and perform mining and transactional operations, regardless of entity or context [34] , [35] . Each blockchain node has maximum privileges to write, read, audit, or analyze blockchain records, such as cryptocurrencies. A public blockchain-based peer-to-peer (P2P) network allows users to collect transaction records and initiate mining processes to get desired outputs. Miner nodes gather information about transactions in blocks, verify their legitimacy, then reach consensus and start adding results and blocks to the current blockchain [36] .

A consensus mechanism is used to ensure that blocks are identical across the blockchain and that no node has so many blocks to collide. Members are not identified on the public blockchain. They are allowed to build blocks before mining, and each node makes the public blockchain vulnerable to Sybil attacks. Proof-of-Work (PoW) consensus is a powerful way to address such issues on public blockchains. If a competitor wants to dominate the blockchain in this process, 51% of the blockchain network's mining power is needed. Cryptographic keys whose address is a hash of the user's public key are used to secure transactions on the blockchain. Nodes can participate in

transactions and transfer additional node assets simply by signing a hash of their ability to retrieve information and including the new owner's public key throughout the transaction. Similarly, the current owner must verify the signature to verify the chain of ownership [37].
However, neither the PoW protocol nor the public blockchain approach are suitable for financial and banking applications due to the huge amount of data required and the complexity of the computer systems involved. However, effective and uncomplicated methods for these applications are currently being developed [37]. The PoW consensus algorithm reduces the number of nodes participating in PoW and encourages multiple mining nodes to participate. As a result, we plan to reduce energy waste caused by excessive hashing power in mining competitions and to fairly distribute mining opportunities [15].

Private blockchain

Private blockchain technology is a permissioned centralized network that allows the private exchange and distribution of any amount of data within a community of entities or people. In addition, since private blockchain mining operations are performed by individuals or specific companies, the blockchain will not be used by new or unknown users unless a special order is issued by the governing body [34]. One of private blockchain’s most popular features is the Hyperledger [34]. To ensure confidentiality and stability, a deterministic shared consensus protocol has been proposed that works in planning, preparation, and interaction phases used in private blockchains. Writing within the private blockchain is restricted, and the network is only allowed to write or transact to management nodes. This makes private blockchains appear centralized. However, other characteristics of private blockchains, such as consensus and distributed ledgers, make this type of blockchain suitable for banks and financial institutions.

Consortium blockchain

The consortium blockchain mechanism is a kind of private and public blockchains which is hybrid. Individuals or a group of companies make the decision on block verification and consensus [34] , [38] . This coalition of organizations agrees on the network’s presence and mining nodes. The network block, where the extracted block is assumed to be a legitimate block, is minted by a multi-signature method if it is accepted and signed by the governing nodes [38]. Consortium blockchains allow individual or organizational participants to verify blocks, rather than requiring everyone to participate in the process or having a single entity to decide the verification process. Hyperledger Fabric is the exact example of a consortium blockchain framework [39]. The consortium's blockchain uses consensus algorithms such as Practical Byzantine Fault Tolerance (PBFT) and Byzantine Fault Tolerance (BFT) via the Tendermint algorithm to validate transactions and distribute applications on multiple machines in a secure and consistent manner.

2.12. Miner

A peer in a blockchain network that solves difficult cryptographic problems and confirms or verifies transactions by adding them or "blocking" them on the blockchain, thereby updating the ledger.

2.13. Mining

To define a new block and add it to the blockchain, there are different strategies for choosing nodes that create hash values based on consensus algorithms. These nodes in the network must solve complex "mathematical puzzles", and once the puzzle is solved, all other nodes in the network confirm that their computations are correct. The process of solving this puzzle and creating a new hash as a result is called "mining". Mining is usually based on advanced mathematical calculations and therefore requires a lot of computing power. It also requires the use of dedicated computer hardware. As a result, not all network nodes can act as "miners". To change the data stored in a block, a malicious actor would have to edit the entire blockchain after that given block. However, this is basically impossible as it requires a lot of computational and processing power.

Miners earn a mining reward

Mining consumes energy, and energy costs money. Therefore, we need an incentive for miners to mine new blocks. This incentive is called a “mining reward” and is usually paid in the cryptocurrency of the blockchain network. For example, miners are rewarded with the cryptocurrency "ETH" for mining blocks on the Ethereum-based blockchain. Without miners, there are no new blocks. As a result, the blockchain becomes useless.

2.14.Consensus mechanism

Consensus rules are one of the central points of blockchain [40] . These algorithms help reach a common consensus on which blocks to include in the blockchain. Once consensus is formed on the network validating a block, the block is added to the chain, becoming a permanent, immutable record accessible to all participants. Consensus rules also aim to reach mutual agreement in a secure manner that can filter out malicious participants. It is also possible that the system contains faulty participants. However, until the majority of participants are honest (the system continues to behave as originally desired), the entire system can remain in a trusted environment. There are many consensus rules today (about 30 [41]). One of the best known is the Proof of Work [41] (PoW) consensus, which applies to the Bitcoin [11] and Ethereum (main network) [23] blockchains. This consensus is based on cryptographic challenges that must be solved by the members participating in the consensus. This consensus rule provides a safe environment until her 51% of participants are honest. Note that PoW consensus is a hardware-resource intensive task that is computed among the participants in the consensus rule. The member who solves a particular her PoW faster than other participants, gets more cryptocurrencies (e.g. Ether on the Ethereum blockchain). Practical Byzantine Fault Tolerance [30] (PBFT) is one of the most widely used consensus rules between private and consortium blockchains (such as Hyperledger Fabric [40]). This consensus is based on successive votes that can create a safe environment until her 2/3 of the consensus participants are honest. Unlike PoW, the purpose of PBFT is not to set up a competition between participants to acquire cryptocurrencies, but to reach consensus more securely. This consensus rule consumes less hardware resources because there are no encryption issues with this consensus. Two important matches are Proof of Work and Proof of Stake.

Proof of Work

When a new block is added to the blockchain, consensus between nodes is required. For this reason, Proof of Work (PoW) algorithms require each node to solve a puzzle whose difficulty can be adjusted. Therefore, the first node to solve the puzzle gets the right to add a new block to the current chain. The effort a node puts in to solve a puzzle is called PoW and is paid to the node that computes the right to hash. This node is called a mining node or miner and the actions taken to solve the puzzle are called mining [42]. The solution to the puzzle is found in PoW, so normally when created using the SHA-256 hash function, the hash should start with a series of zero bits. The average amount of work required is exponential in the number of zero bits required and can be found by performing a simple hash operation. In PoW, the puzzle difficulty is adjusted each time a 2016 block is added, so the average rate of adding a new block to the chain is 1 block every 10 minutes. [42]. When a new block is created, the header information is combined and sent as an input parameter to a SHA-256 hash function [43] . If the output of this function is below a threshold T (depending on difficulty), the sought value is accepted. Otherwise, the node should continue calculating the secret value until the output of the SHA-256 function is accepted. The smaller the value of T, the higher the difficulty of the puzzle. [42].

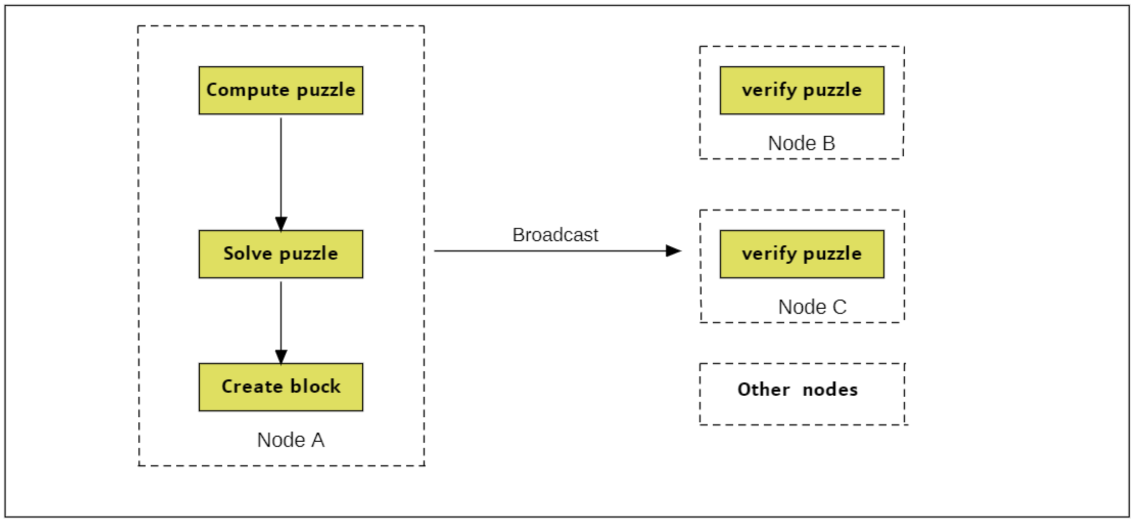


Figure 8 PoW consensus Mechanism [44]

2.14.1.1. The advantage of PoW

PoW has been widely successful mainly due to the following characteristics:

- Finding solutions to specific problems is difficult.
- If you find a solution to this problem, you can easily verify that it is the correct solution.

At Proof of Work, other nodes verify the validity of a block by checking if the hash of the block's data is less than a preset number. Due to the limited supply of computing power, miners are also encouraged not to cheat. Attacks on networks are expensive due to high hardware and energy costs and the potential loss of mining profits.

Prove of Stake

The Proof-of-Stake (PoS) algorithm aims to reduce the power consumption of the ever-growing PoW blockchain network [45] . As an alternative to computationally intensive puzzle solving, Proof of Stake aims to stake the economic share of peers in the network (as seen for example in Peercoin5). Here, the term “miner” is replaced with “validator” and one validator is chosen to publish a block on the blockchain, similar to proof-of-work algorithms.

The difference is in the selection of validators. In Proof of Stake, validators are chosen pseudo-randomly, with probabilities proportional to the validator's share in the network (as seen in NXT6 and Blackcoin7). Proof of Stake (PoS) is an alternative to PoW that allows miners to create blocks based on the amount of resources at risk instead of requiring computational work [46]. Based on such an approach, PoS can reduce the energy cost of the expensive PoW mining process and reliance on dedicated hardware. However, PoS-based blockchains, which do not require resource overhead, make the network more vulnerable to attacks.

2.15. Ledgers

A traditional blockchain network maintains a public ledger of all transactions as a copy of every node. A ledger works like a public database owned by no one. In public and private blockchains, ledgers are managed securely through cryptography. Ledgers contain blocks of transactions, and some ledgers can even store other types of data such as: B. The state of the program running on the blockchain.

Distributed Ledger Technology (DLT)

This is a family of technologies that also includes blockchains where the ledger is run by a group of peers rather than a single central authority.

2.16. Blockchain platforms for IoT

This section describes the most popular and popular blockchain platforms that support IoT application development and service integration.

Bitcoin Platform

Bitcoin is a popular blockchain platform that includes active cryptocurrencies that provide a decentralized system for securely executing transactions without intermediaries or third parties [47] . Bitcoin is involved in many IoT platforms to create micropayments and act as a wallet for transactions. However, while Bitcoin uses a limited scripting language to carry out these transactions, most IoT platforms use a common and reliable solution of smart contracts. Smart contracts can more securely manage and record all interactions within a transaction without the limitations of scripting languages.

Ethereum Platform

Originally, Ethereum was a public, permissionless, blockchain-based platform implementing a Proof of Work based consensus protocol. Ethereum is an open-source blockchain framework that leverages decentralized applications where anyone can own and control the platform [48] . Additionally, the platform is flexible and adaptable, including smart contracts that enable the integration of new IoT technologies and applications. A vibrant and popular platform with a broad community supporting the development of multi-language-based application such as Go, C++ and Python. The platform will be developed based on a consensus mechanism that enables the development and customization of IoT applications and reduces the latency of blockchain approaches. However, this platform does not provide the data confidentiality which is required for most IoT applications.

Hyperledger-fabric Platform

The Hyperledger-Fabric platform is a incredibly famous open-supply platform (constructed primarily based totally at the Golang and Java languages) permitting builders to construct blockchain programs the usage of a modular structure method [49] . This modular method permits the platform to be prolonged with a couple of components, which include club offerings and consensus algorithms, making it an awesome preference to help enterprise organization solutions, in conjunction with diverse different blockchain platforms. In addition, Hyperledger Fabric is a permission-primarily based totally community that offers a information confidentiality function to encrypt transactions in order that they cannot be changed via way of means of unauthorized persons. However, there are various obstacles and downsides associated with the platform`s cap potential to help IoT programs and development. For example, it's far much less or in part decentralized, extra prone because of consider issues, has handiest one validator node and has bad scalability of the consensus algorithms required to make a dependable system-primarily based totally settlement throughout a couple of gadgets of an organization’s dispensed community [50].

Stellar Platform

It features a public blockchain with its own consensus algorithm which is like Practical Byzantine Fault Tolerance (PBFT) [51] but uses elements from Social network modeling. The difference is that a node agrees to a transaction if its neighbors agree. Neighboring nodes are trusted more than other nodes. If a transaction is accepted by a threshold number of nodes in the network, there is a cascading effect due to homogeneity, and the transaction is confirmed by the entire network with high certainty. As such, the protocol requires far less computational power as it does not require solving cryptographic puzzles. Unlike Ethereum, there is no specific language for smart contracts. It is still possible to string together several transactions and write them atomically within the blockchain. Stellar also features special accounts called multi-signature which essentially lets several owners handle a single account. A minimum agreement must be reached between the owners to operate from these accounts. Transaction chaining and multi-signature accounts can be combined to create more complex contracts [52] .

Multichain Platform

Multichain is a private blockchain platform that provides application development and deployment as well as offers privacy and a control-based P2P network [53]. The multichain platform enhances and leverages the existing application program interfaces (APIs) of Bitcoin's core software by adding new features to support financial transactions. The platform provides both an API and a command line interface to support multi-chain configurations. Additionally, Multichain is a permissioned blockchain that offers options for application development. It can be an open blockchain or a closed blockchain, depending on your business needs. Additionally, it is an open source blockchain platform that supports C, C++, Python, and Java script. Multichain is a permissioned blockchain and provides a great solution for IoT to collect data when data erasure is a concern, but it cannot protect against the risk of data theft. In addition, communicating intelligent things with other resources within the allowed multi-chain is slow and costly [54].

Platform	**Blockchain**	**Popularity&active**	**Consensus algorithm**	**Pricing**	**Supported languages**	**Smart contracts**
Bitcoin	Public	High	PoW	Fees per transactions	Script and c++	No
Hyperledger-Fabric	Private, Permissioned	High	PBTF	Open Source Price	Python, Golang and Java	Yes
Multichain	Private, Permissioned	Medium	PBTF	Free, Open source Price	Python, C#	Yes
Quorum	Public, Permissioned	High	Raft, IBFT	Fees per transaction	Python, C#	Yes
Lisk	Public, Permissioned	Medium	DPoS	Fees per transaction	Java Script	Yes
Litecoin	Public	Low	Scrypt	Fees per transaction	C++	No
HDAC	Public, Permissioned	Low	EPoW	Fees per transaction	Web Assembly	Yes
IOTA	Public, Permissioned	Low	PoW, TANGLE	Pricing not clear as yet	Python, C, Javascript	No

Table 2 Blockchain platforms

Table 2 shows a comparison of the existing platforms of blockchain that are used to develop the IoT applications presented in this section.

Most platforms include smart contracts that allow application logic to extend beyond cryptocurrency transactions. The most widely used programming languages are Python, JavaScript, C++, and consensus algorithms PoW and PBT. Most platforms have public and syndicated permissions so you can use them to build global and syndicated applications. In fact, the consensus algorithm is the core feature that determines the performance of his blockchain-based IoT applications in terms of block rate, consistency, scalability, and security. Consensus algorithms based on PoW are considered the most secure in open networks. On the other hand, pow precludes the possibility of block mining on IoT devices due to its high computational requirements. PBFT-based private blockchain consensus mechanisms can provide IoT systems with high block production rates but limit the number of participating miners. Among the above blockchain projects, Ethereum is suitable for many IoT applications, including a large number of his IoT systems and heterogeneous networks. As a public blockchain, Ethereum has high scalability as it can support many heterogeneous devices. On the other hand, Hyperledger Fabric is suitable for his IoT network with large amounts of data. Fabric integrates blockchain through a customer service approach to achieve high transaction volumes up to tens of thousands of transactions per second. Hyperledger Fabric requires controllable network infrastructure and is not as publicly accessible as Ethereum.

2.17. Visual scenario on how the blockchain corporate with other parts

In figure 9, a visual scenario shows a conceptual scenario of an IoT blockchain platform. In this scenario, a number of his IoT devices, data stores, user devices, servers, and local bridges are connected by a peer-to-peer blockchain network. An IoT server is a service provider that can interact with local bridges and blockchain networks in order to provide various services to end-users, such as storing data in storage space via blockchain networks. A data storage network on the blockchain can store physical device profiles and environmental data collected by sensors. It can either be a hardware storage like a hard disk or a software storage such as a DB. User client can be any terminal devices, such as smart phones, laptops, and PCs, through which end users can read or write data to the blockchain network. For example, home users can view the status of various home appliances stored on the blockchain for a specific period of time. There are various communication protocols that developers can apply to their IoT products and systems. B. Bluetooth, ZigBee, WiFi, and 2G/3G/4G cellular. A local bridge uses these communication technologies to connect a cluster of IoT devices to a server and also acts as a service agent for those devices.

Nowadays, with the advancement of hardware technology, embedded devices such as Raspberry Pi can directly consume web services by invoking representational state transfer application programming interfaces (REST APIs). Therefore, two approaches are presented for Communicating with physical devices, which is either via the local bridges or via direct wireless communications. Unlike most existing projects that focus on connecting IoT devices to blockchain networks using bridges, the proposed work focuses on simple communications. IoT devices can be categorized into sensors and actuators. Sensors are used to collect environmental data such as temperature and send this data to our server for further use. Actuators, on the other hand, are used to perform specific actions (such as turning on lights) according to the commands they receive. user.

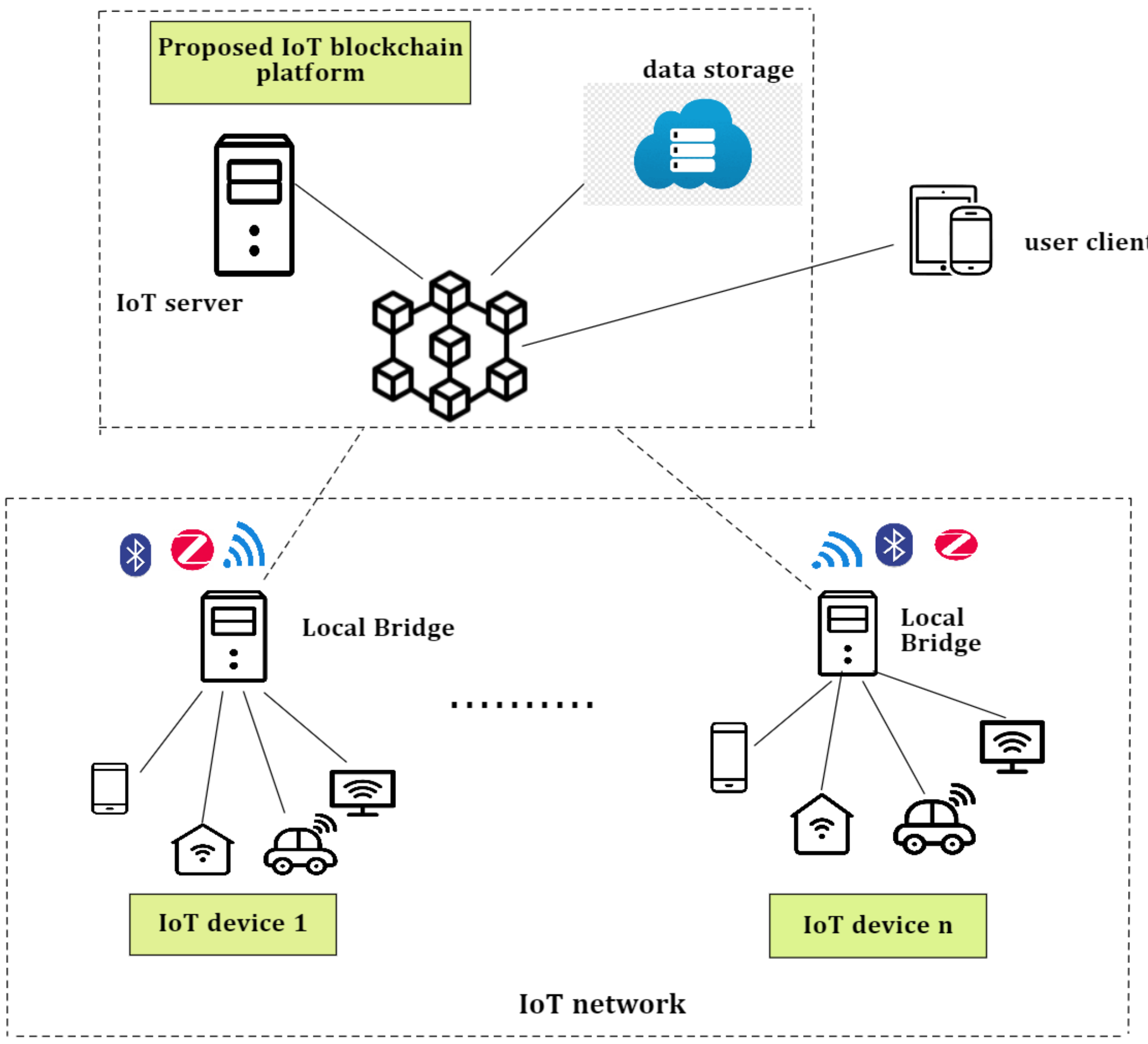


Figure 9 Visual scenario of correlation in IoT and blockchain

2.18. Functional aspect of blockchain

A blockchain platform includes some specific processes and architectures. First, blockchain has a specific transaction generation process to ensure transaction validity. Blockchain data units are called transactions, and a certain number of transactions is a block. Each transaction participant in a blockchain system is called a MINER. When a new transaction is generated on the blockchain, miners broadcast the transaction information across the network and each miner integrates the received transaction information into blocks. To ensure security, each miner should store a complete copy of the blockchain data. Each miner tries to find her Proof of Work (PoW) hard enough within a block, a process known as mining. When the miner finds evidence, it broadcasts it across the network. Other miners agree on block validity through a specific consensus mechanism as shown in Figure 10.

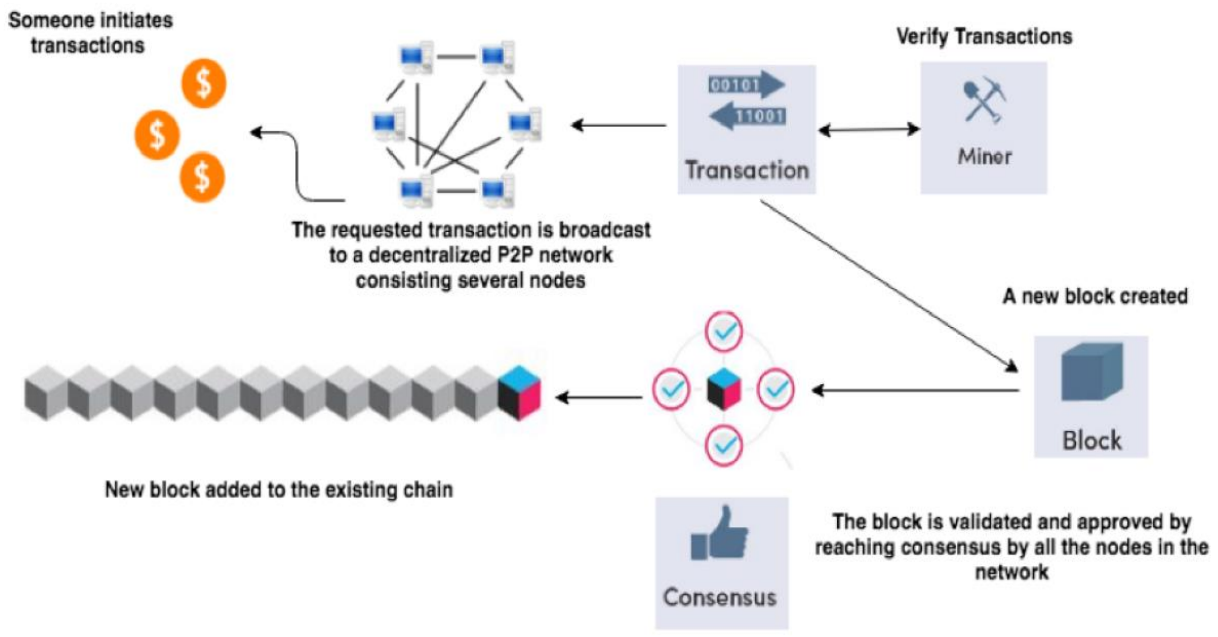

Someone initiates transactions
The requested transaction is broadcast to a decentralized P2P network consisting several nodes
00101
11001
Transaction
Verify Transactions
Miner
A new block created
Block
New block added to the existing chain
Consensus
The block is validated and approved by reaching consensus by all the nodes in the network

Figure 10 Functional diagram of a Blockchain network

Chapter 3

3. Implementation of evidence in blockchain

This section describes the work of implementing a public blockchain using the Ethereum platform with the Python programming language. Blockchain is widely used in the field of IoT security. However, when it comes to IoT forensics, this concept is still in the testing stage. When it comes to implementing blockchain in an IoT environment, it can play an important role in the security, management, control, and security of IoT devices. One example is digital evidence, where the use of blockchain technology is encouraged to ensure quality and prevent tampering. In this case, blockchain technology is used to store data controlled by IoT devices. In addition to the ability to make IoT device management more efficient, IoT devices can be removed from the control of a central authority that can operate or shut down the system. This makes it more difficult to attack networks because they don't revolve around people. Additionally, data received from IoT devices and stored on public blockchain networks is also less vulnerable to plaintext and cryptographic attacks due to hashing of the data on the blockchain.

3.1. Functionality of the modules

In this python program, we used some modules to run each section of our blockchain project. These modules are as following:

3.1.1. App.py

This module is the main module in our program.

3.1.2. AppConfig.py

It's the module to set the relation between app.py class with excel module and Blockchain module.

3.1.3. Blockchain.py

It's a module for blockchain.

3.1.4. Data_sent.py

This class maintains the strings of the whole program.

3.1.5. ExtractFromExcel.py

With this module, we extract the information from the excel file.

3.2. Analysis of the codes

In this part, we go forward step by step by considering the layers.

3.2.1. App.py

The first module called data-sent contains string data shown in Figure 11.

```
from data_sent import data_sent
from  AppConfig import AppConfig
```

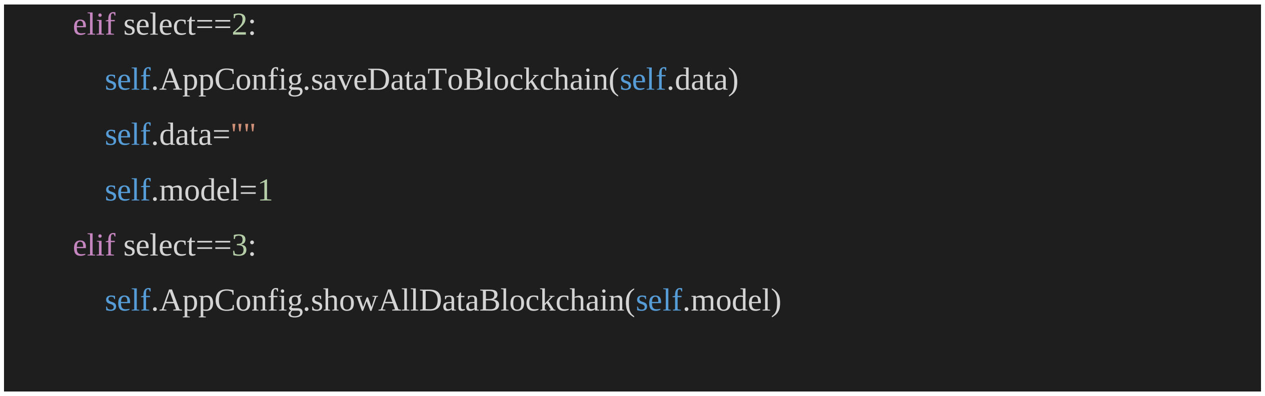
Figure 11 Import a dataset

In the following class called App, we added these modules and we built objects from “data_sent” and “AppConfig” to have access to its methods in the whole program. The “help” method, prints us the contents of the helping list which is in the “data_sent” class. “app” method is the main method for running the program at app class which creates the object of appConfig for running the whole program as shown in Figure 12.

```
class App():
    def __init__(self) -> None:
        self.data_sent=data_sent()
        self.AppConfig=AppConfig()
        self.data=""
        self.model=-1
def help(self):
        for index,item in enumerate(self.data_sent.p1):
            print(item)
def app(self):
        self.appConfig()
```
Figure 12 Class APP()

In the “appConfig” method, the user enters the number as the input to get the output with calling “appSelect” as shown in Figure 13.

```
def appConfig(self):
        print("welcome to system to manage data blockchain ---\n")
        while True:
            self.help()
            select=int(input("Enter : \n"))
            self.appSelect(select)
            if select==4:
                return False
 def  appSelect(self,select):

        if select==1:
            self.data=self.AppConfig.getDataFromExcel()
            self.model=-1
```



```
    elif select==2:
        self.AppConfig.saveDataToBlockchain(self.data)
        self.data=""
        self.model=1
    elif select==3:
        self.AppConfig.showAllDataBlockchain(self.model)
```

Figure 13 Functionality of “appConfig” and “appSelect” methods

There are 2 parameters in the whole program as model and data which are integer and string. We use “data” variable for storing the extracted data from excel as shown in Figure 13. Also we use “model” variable to control the structure and prevent the estimated errors. When the users enters 1 as input, it starts to extract the information from the row of the excel file and insert in the “data” variable. If the user inserts “2”, the row will be stored in data base. For storing data in data base, we use “saveDataToBlockchain” method. If the user inserts “3” as input, it shows the data stored in the blockchain and identify the validity of the data as illustrated in figure 13.

3.2.2. AppConfig.py

The module called “extraxtFromExcel” is responsible for extracting data from excel file. Also, the last module called “Blockchain” contains the blockchain information as illustrated in Figure 14. The “AppConfig” class is the complementary for the “app” class. In the constructor of “AppConfig” class, we built the objects to use in all the program as shown in Figure 14.

```
from data_sent import data_sent
from ExtractFormExcel import ExtractFormExcel
from Blockchain import Blockchain
class AppConfig():
    def __init__(self) -> None:
        self.data_sent=data_sent()
        self.ExtractFromExcel=ExtractFormExcel()
        self.Blockchain=Blockchain()
```

Figure 14 “AppConfig” class constructor

In the “getDataFromExcel” method, we extracted data from excel file to return this “data” in the method again. At first, we request the user to enter the row of the record in the excel file which he wants to save in the blockchain. Then, we place this parameter as an input for “configExcel” which is in the class called “ExtractFromExcel”. Now if that row exists, it shows us the data related as shown in Figure 15.

```
def getDataFromExcel(self):
        try:
            select=int(input(self.data_sent.p2))
            data=self.ExtractFromExcel.configExcel(select)
            print(self.data_sent.p5)
            print(self.data_sent.p3)
            print (data)
            print(self.data_sent.p5)
        except:
             print(self.data_sent.p4)
return data
```

Figure 15 Get data from the row in excel file

The duty of “saveDataToBlockchain” method is to store data in blockchain. It has an input as the data extracted from excel file (In the case the data parameter is not empty). This data will be stored in the blockchain with the “mine-block” method. If there is an error, we will receive the error message as shown in Figure 16.

```
def saveDataToBlockchain(self,data):
    try:
        if len(data) !=0:
            print(self.data_sent.p5)
            print(self.data_sent.p6)
            self.Blockchain.mine_block(data)
            print(self.data_sent.p5)
        else:
            print(self.data_sent.p7)
    except:
        print(self.data_sent.p7)
```

Figure16 Save data to blockchain method

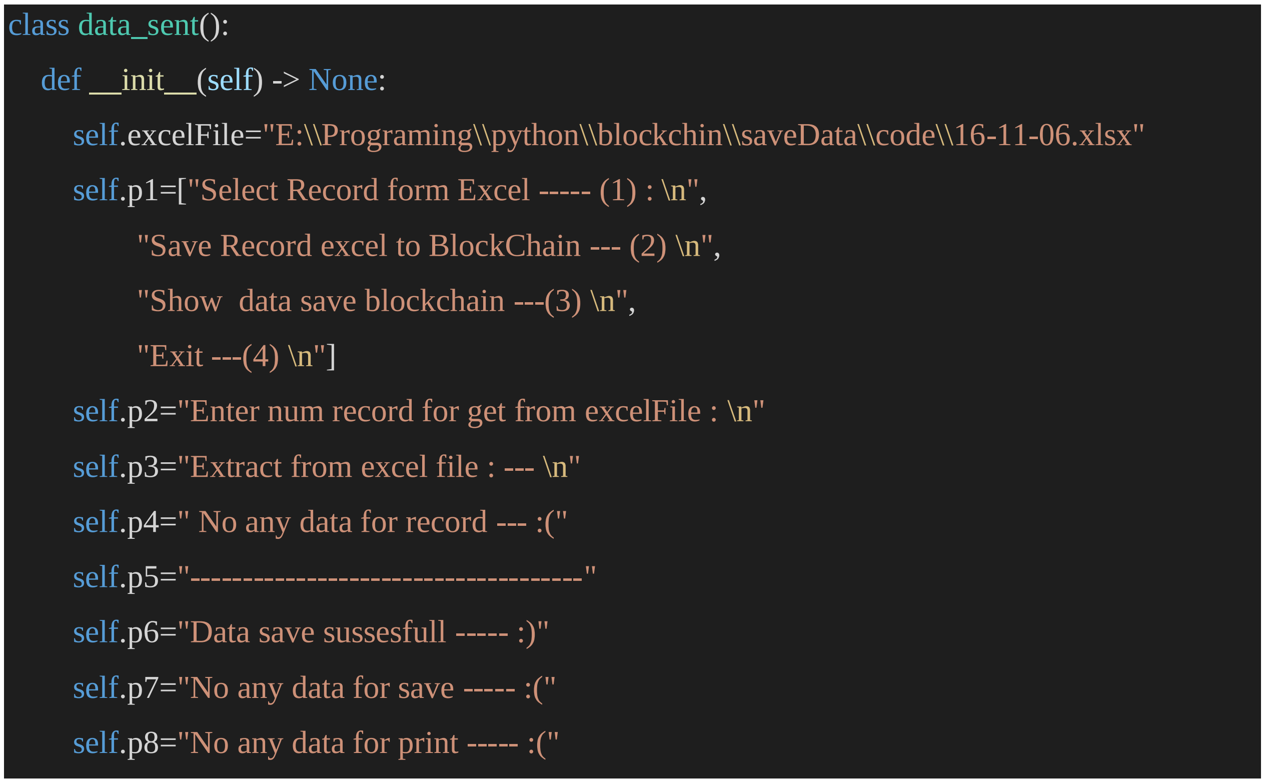
In the “showAllDataBlockchain” method, the display of data will be done by “display_chain” method and the validation activity will be done by “valid” method. If there is an error, it shows us the error message in output in Figure 17.

```
def showAllDataBlockchain(self,data):
    if data ==1:
        print(self.data_sent.p5)
        self.Blockchain.display_chain()
        self.Blockchain.valid()
        print(self.data_sent.p5)
    else:
            print(self.data_sent.p8)
```

Figure 17 Display all data in blockchain

3.2.3. data_sent.py

This module is the prepared messages which we use during our program for better coherence. There is some information in the constructor of this class as string and list to use in the whole program as shown in Figure 18.



```
class data_sent():
    def __init__(self) -> None:
        self.excelFile="E:\\Programing\\python\\blockchin\\saveData\\code\\16-11-06.xlsx"
        self.p1=["Select Record form Excel ----- (1) : \n",
                "Save Record excel to BlockChain --- (2) \n",
                "Show  data save blockchain ---(3) \n",
                "Exit ---(4) \n"]
        self.p2="Enter num record for get from excelFile : \n"
        self.p3="Extract from excel file : --- \n"
        self.p4=" No any data for record --- :("
        self.p5="-------------------------------------"
        self.p6="Data save sussesfull ----- :)"
        self.p7="No any data for save ----- :("
        self.p8="No any data for print ----- :("
```

Figure 18 Constructor of “data_sent” class

3.2.4. ExtractFromExcel.py

This module, extracts the data from excel file. Xlrd is a module to read data from excel file (Fig 9). In the constructor of “ExtractFromExcel”, we use data-sent object and xlrd module for reading data from excel file as illustrated in Figure 19.

```
import xlrd
from data_sent import data_sent
class ExtractFormExcel():
    def __init__(self) -> None:
        self.data_sent=data_sent()
        self.wb = xlrd.open_workbook(self.data_sent.excelFile)
        self.sheet = self.wb.sheet_by_index(0)
```

Figure 19 Constructor of "ExtractFromExcel" class

We receive the row information and show the output as a list of the row. The "configExcel" method is for reading the row in Figure 20.

```
def readRow(self,select,index):
        information=[]
        for i in range(select):
            information.append(self.sheet.cell_value(index, i))
        return information
 def configExcel(self,index):
        for i in range(1,index+1,1):
            data=self.readRow(1,i)
            return data[0]
```

Figure 20 Reading each row of excel file

3.2.5. Blockchain.py

At first, we used the standard libraries of python:

Datetime is for time information. Hashlib, is used in constructing a blockchain. Json is for standardization of data to show and store data as shown in Figure 21.

```
import datetime
import hashlib
import json
```

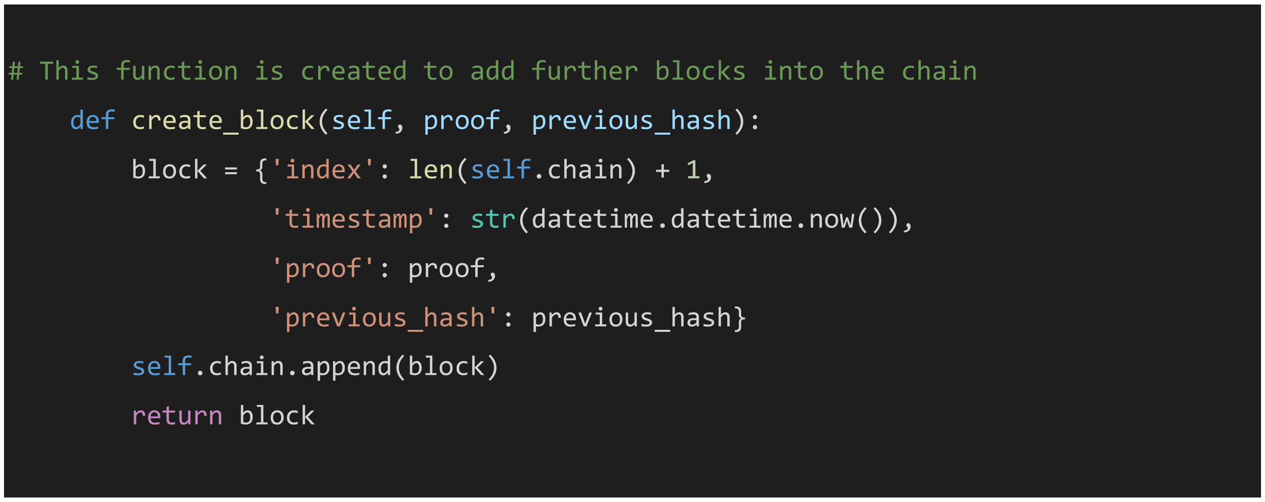
Figure 21 Standard libraries for blockchain

In the constructor of “Blockchain” class, we created a chain as list. We use the Proof of Work algorithm in this blockchain. As the input of “create_block”, because we are in the gensis level of blockchain (as the first block), we have the assumption of having predefined variables called proof=1 and previous_hash=0 (as we are in the first block, so no block was presented before this block in Figure 22.

```
class Blockchain:
     # This function is created to create the first block and set its hash to "0"

    def __init__(self) -> None:
        self.chain = []
        self.create_block(proof=1, previous_hash='0')
```

Figure 22 Constructor of blockchain

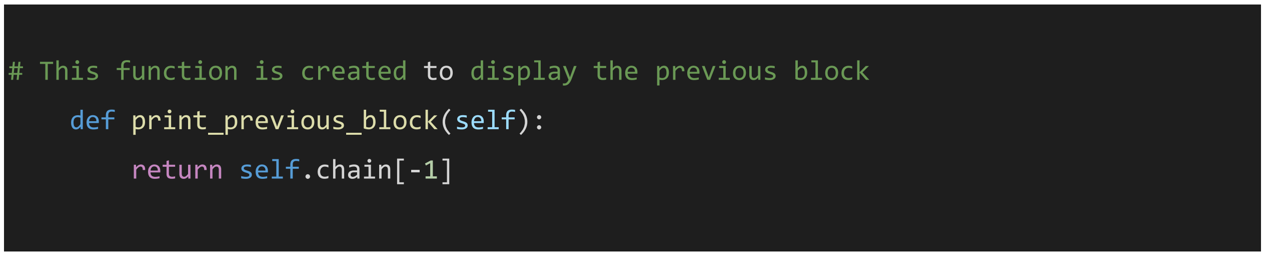
Here we want to create the new other blocks (The first block was created previously). In this method, we have to input variables called “proof” and “previous_hash”. In the body of this method, we have the block dictionary which consists of index as the length of the list chain, timestamp as the time of the creation of the block, the proof amount and the previous_hash of previous block. After that this dictionary is created, we added this dictionary to the chain list. Then, this dictionary amount (as a new block of information) is as our output as shown in Figure 23.



```
# This function is created to add further blocks into the chain
    def create_block(self, proof, previous_hash):
        block = {'index': len(self.chain) + 1,
                 'timestamp': str(datetime.datetime.now()),
                 'proof': proof,
                 'previous_hash': previous_hash}
        self.chain.append(block)
        return block
```

Figure 23 Creating the new block to add to the chain

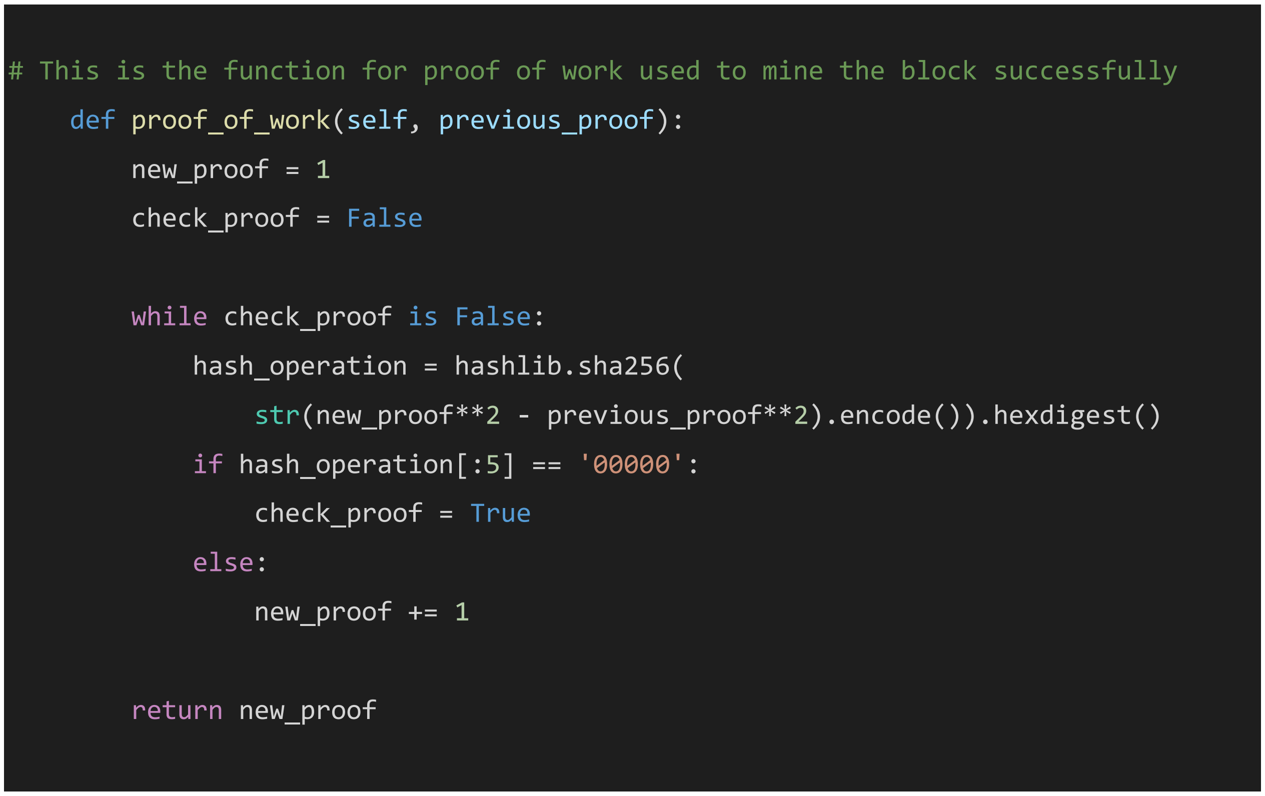
This method is to show the previous block (there is -1 as the output to return the previous block in Figure 24.



```
# This function is created to display the previous block
    def print_previous_block(self):
        return self.chain[-1]
```

Figure 24 Returning the previous block info

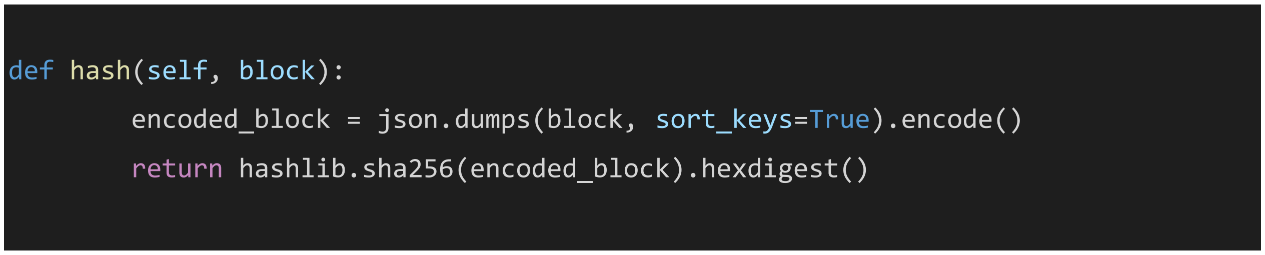
Now, there is the method to implement the Proof of Work algorithm . Our purpose is to Calculate new_proof until the check_proof is equal to "False". In this algorithm, we used one argument called previous_proof which indicates the previous proof. There are 2 variables called "new_proof" with the first assumption of 1 and "check_proof" as a logical variable with the first assumption of False in Figure 25. We continue the loop until we exit from loop. Hexdigest() Returns the encoded data in hexadecimal format. The Python hashlib module is an interface for hashing messages easily. This contains numerous methods which will handle hashing any raw message in an encrypted format. The core purpose of this module is to use a hash function on a string, and encrypt it so that it is very difficult to decrypt it.



```
# This is the function for proof of work used to mine the block successfully
    def proof_of_work(self, previous_proof):
        new_proof = 1
        check_proof = False

        while check_proof is False:
            hash_operation = hashlib.sha256(
                str(new_proof**2 - previous_proof**2).encode()).hexdigest()
            if hash_operation[:5] == '00000':
                check_proof = True
            else:
                new_proof += 1

        return new_proof
```

Figure 25 Proof of Work algorithm

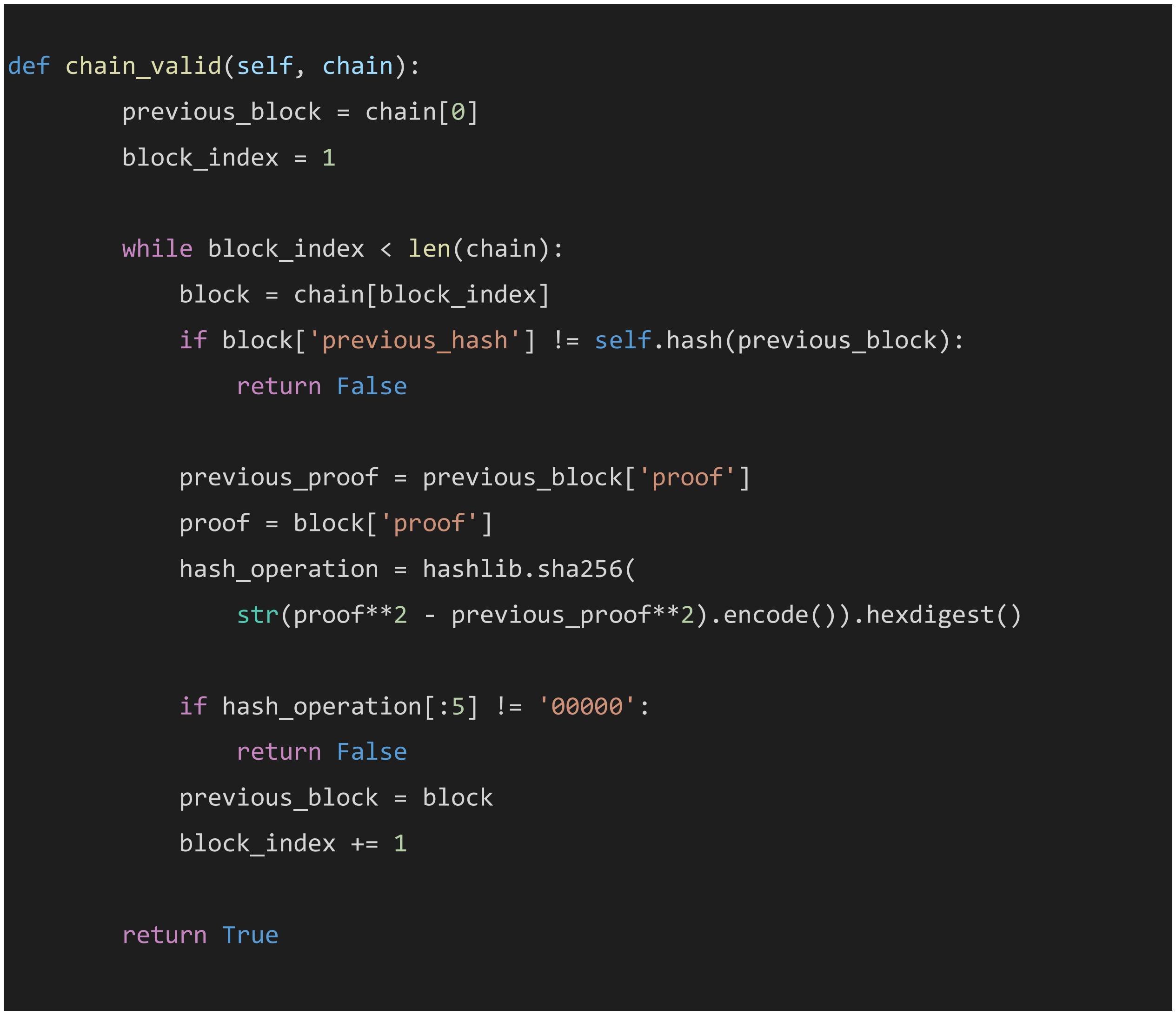
This method aims to create the hash of block. In this method, it encodes a block (as input) using JSON structure to create the hash as the output as shown in Figure 26.



```
def hash(self, block):
        encoded_block = json.dumps(block, sort_keys=True).encode()
        return hashlib.sha256(encoded_block).hexdigest()
```

Figure 26 Creating the hash of the identified block

This method works on validity of the identified chain in Figure 27. The "previous_block" receives the value of index 0 of the chain as the input. In the while loop, we consider the validation (If it is valid or not). In the while loop, if the "block_index" is more than the length of the chain, the chain is valid, else the chain is not valid for the blockchain. In the body of the while loop, we try to calculate the previous block hash and then compare it with the hash of previous block. If the “previous_block” hash is different from the block['previous_hash'], the block is not validated as shown in Figure 27.



```
def chain_valid(self, chain):
        previous_block = chain[0]
        block_index = 1

        while block_index < len(chain):
            block = chain[block_index]
            if block['previous_hash'] != self.hash(previous_block):
                return False

            previous_proof = previous_block['proof']
            proof = block['proof']
            hash_operation = hashlib.sha256(
                str(proof**2 - previous_proof**2).encode()).hexdigest()

            if hash_operation[:5] != '00000':
                return False
            previous_block = block
            block_index += 1

        return True
```

Figure 27 Validation of block and chain

If the while loop doesn’t return False, we will check the validity of the block with “hash_operation” variable. For calculating “hash_operation” variable using standard libraries, first, we calculate “previous_proof “ and “proof” parameter, then with the defined structure, we calculate the “hash_operation” to check the validity of chain. If the if condition is not right, then the while loop will continue until the validation or not validation of chain is defined as shown in Figure 27.

This method is to store the data in the created block. The argument called “data” are the input data to be stored in the block (Fig 20). At first, it creates the “previous_block” parameter (as a block type) using the “print_previous_block” method. Then the “proof” variable, takes the proof of work as input. After calculating the “previous_hash”, it is the time to create the block by the method called “create_block” with the output template for displaying block information as shown in the Figure 28.

```
def mine_block(self,data):
        previous_block = self.print_previous_block()
        previous_proof = previous_block['proof']
        proof = self.proof_of_work(previous_proof)
        previous_hash = self.hash(previous_block)
        block = self.create_block(proof, previous_hash)

        response = {'message': data,
                    'index': block['index'],
                    'timestamp': block['timestamp'],
                    'proof': block['proof'],
                    'previous_hash': block['previous_hash']}
        print(response)
```

Figure 28 Store data in the block using standard output

As shown in Figure 29, this method is responsible for displaying the chains and the length of the chains.

```
def display_chain(self):
        response = {'chain': self.chain,
                    'length': len(self.chain)}
        print(response)
```

Figure 29 Showing list of chains and lengths of chains as output

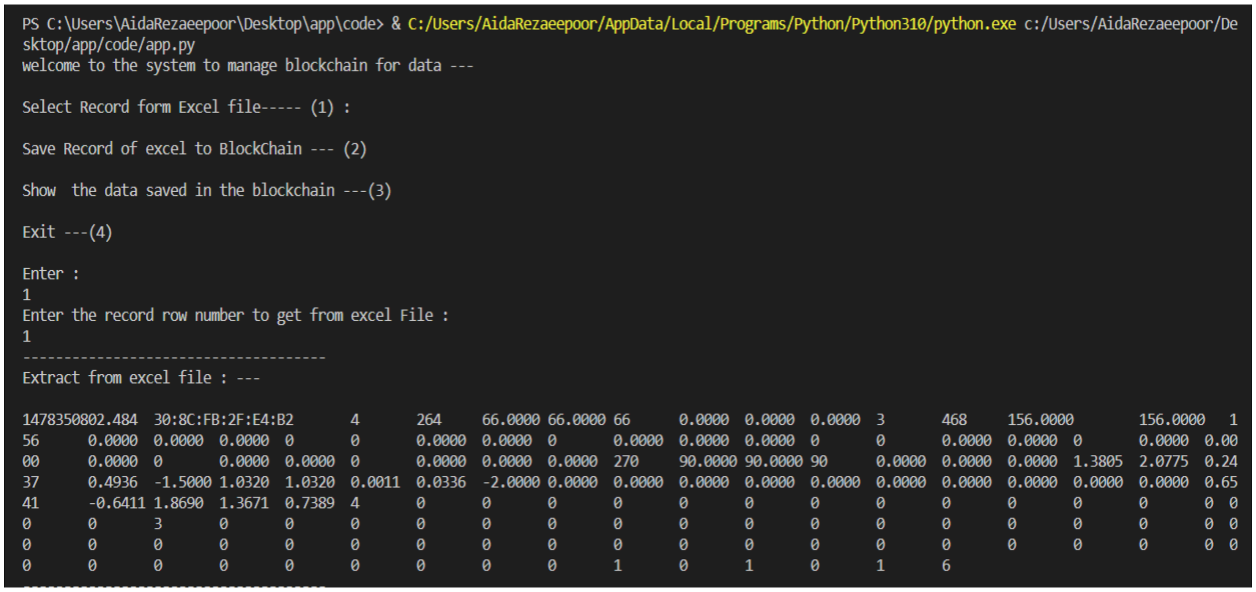
This method checks validity of blockchain with “chain_valid” method as shown in Figure 30.

```
def valid(self):
        valid = self.chain_valid(self.chain)

        if valid:
            response = {'message': 'The Blockchain is valid.'}
        else:
            response = {'message': 'The Blockchain is not valid.'}
        print(response)
```

Figure 30 Checking validity of blockchain

3.3. Display in console

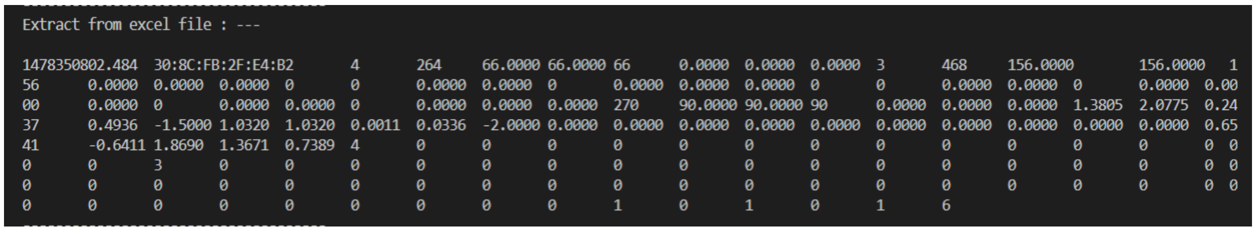
3.3.1. Select a row from excel file



```
PS C:\Users\AidaRezaeepoor\Desktop\app\code> & C:/Users/AidaRezaeepoor/AppData/Local/Programs/Python/Python310/python.exe c:/Users/AidaRezaeepoor/De
sktop/app/code/app.py
welcome to the system to manage blockchain for data ---

Select Record form Excel file----- (1) :

Save Record of excel to BlockChain --- (2)

Show  the data saved in the blockchain ---(3)

Exit ---(4)

Enter :
1
Enter the record row number to get from excel File :
1
-------------------------------------
Extract from excel file : ---

1478350802.484  30:8C:FB:2F:E4:B2       4       264     66.0000 66.0000 66      0.0000  0.0000  0.0000  3       468     156.0000        156.0000        1
56      0.0000  0.0000  0.0000  0       0       0.0000  0.0000  0       0.0000  0.0000  0.0000  0       0       0.0000  0.0000  0       0.0000  0.00
00      0.0000  0       0.0000  0.0000  0       0.0000  0.0000  0.0000  270     90.0000 90.0000 90      0.0000  0.0000  0.0000  1.3805  2.0775  0.24
37      0.4936  -1.5000 1.0320  1.0320  0.0011  0.0336  -2.0000 0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.65
41      -0.6411 1.8690  1.3671  0.7389  4       0       0       0       0       0       0       0       0       0       0       0       0       0  0
0       0       3       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0  0
0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0  0
0       0       0       0       0       0       0       0       0       1       0       1       0       1       6
-------------------------------------
```

Figure 31 Select the specific row from excel file

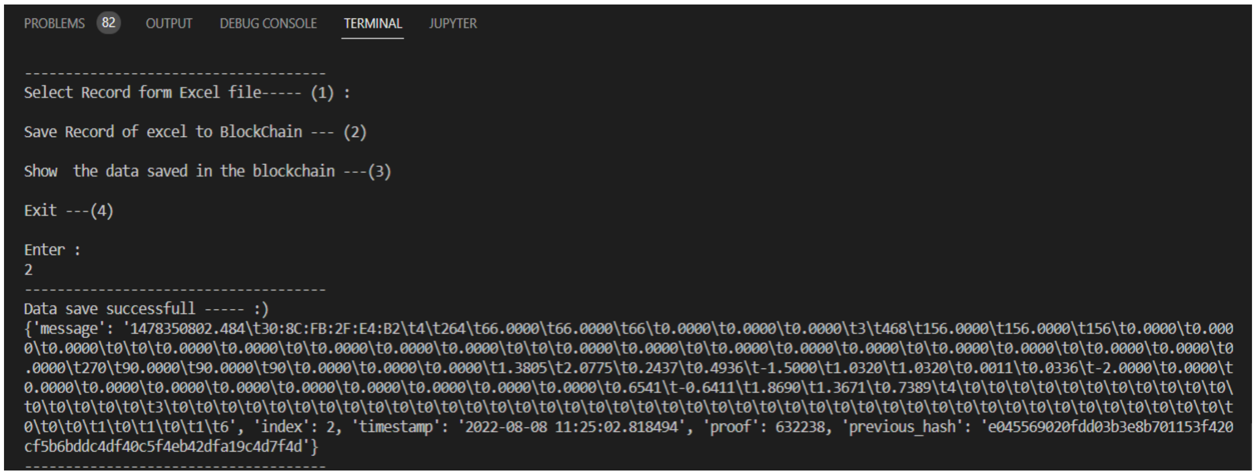
We selected this row with this information from excel file:



```
-------------------------------------
Extract from excel file : ---

1478350802.484  30:8C:FB:2F:E4:B2       4       264     66.0000 66.0000 66      0.0000  0.0000  0.0000  3       468     156.0000        156.0000        1
56      0.0000  0.0000  0.0000  0       0       0.0000  0.0000  0       0.0000  0.0000  0.0000  0       0       0.0000  0.0000  0       0.0000  0.00
00      0.0000  0       0.0000  0.0000  0       0.0000  0.0000  0.0000  270     90.0000 90.0000 90      0.0000  0.0000  0.0000  1.3805  2.0775  0.24
37      0.4936  -1.5000 1.0320  1.0320  0.0011  0.0336  -2.0000 0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.65
41      -0.6411 1.8690  1.3671  0.7389  4       0       0       0       0       0       0       0       0       0       0       0       0       0  0
0       0       3       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0  0
0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0  0
0       0       0       0       0       0       0       0       0       1       0       1       0       1       6
-------------------------------------
```

Figure 32 Row selected and displayed in the console

3.3.2. Save selected row from excel file into the blockchain

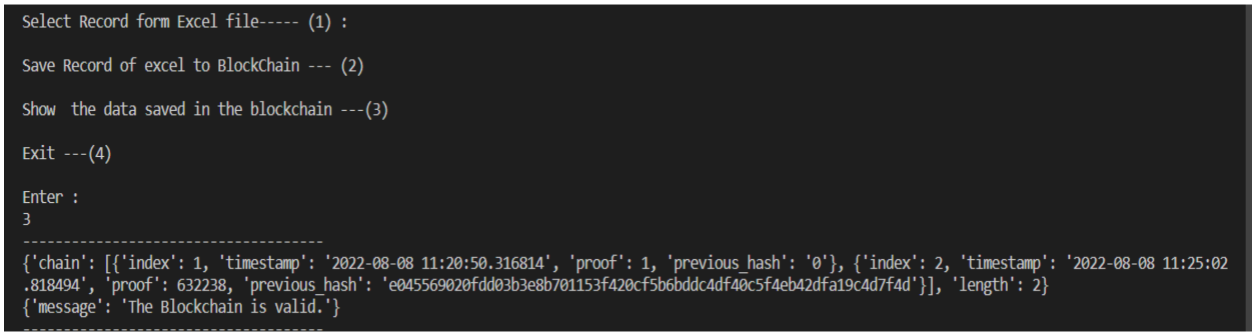


```
------------------------------------
Select Record form Excel file----- (1) :

Save Record of excel to BlockChain --- (2)

Show  the data saved in the blockchain ---(3)

Exit ---(4)

Enter :
2
------------------------------------
Data save successfull ----- :)
{'message': '1478350802.484\t30:8C:FB:2F:E4:B2\t4\t264\t66.0000\t66.0000\t66\t0.0000\t0.0000\t0.0000\t3\t468\t156.0000\t156.0000\t156\t0.0000\t0.000
0\t0.0000\t0\t0\t0.0000\t0.0000\t0\t0.0000\t0.0000\t0.0000\t0\t0\t0.0000\t0.0000\t0\t0.0000\t0.0000\t0.0000\t0\t0.0000\t0.0000\t0\t0.0000\t0.0000\t0
.0000\t270\t90.0000\t90.0000\t90\t0.0000\t0.0000\t0.0000\t1.3805\t2.0775\t0.2437\t0.4936\t-1.5000\t1.0320\t1.0320\t0.0011\t0.0336\t-2.0000\t0.0000\t
0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.6541\t-0.6411\t1.8690\t1.3671\t0.7389\t4\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\
t0\t0\t0\t0\t0\t3\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t
0\t0\t0\t1\t0\t1\t0\t1\t6', 'index': 2, 'timestamp': '2022-08-08 11:25:02.818494', 'proof': 632238, 'previous_hash': 'e045569020fdd03b3e8b701153f420
cf5b6bddc4df40c5f4eb42dfa19c4d7f4d'}
------------------------------------
```

Figure 33 Save the row in the blockchain and displayed in the console

3.3.3. Show the data with template which is saved in the blockchain

It shows all the stored data in the blockchain and checks if the blockchain is valid or not.

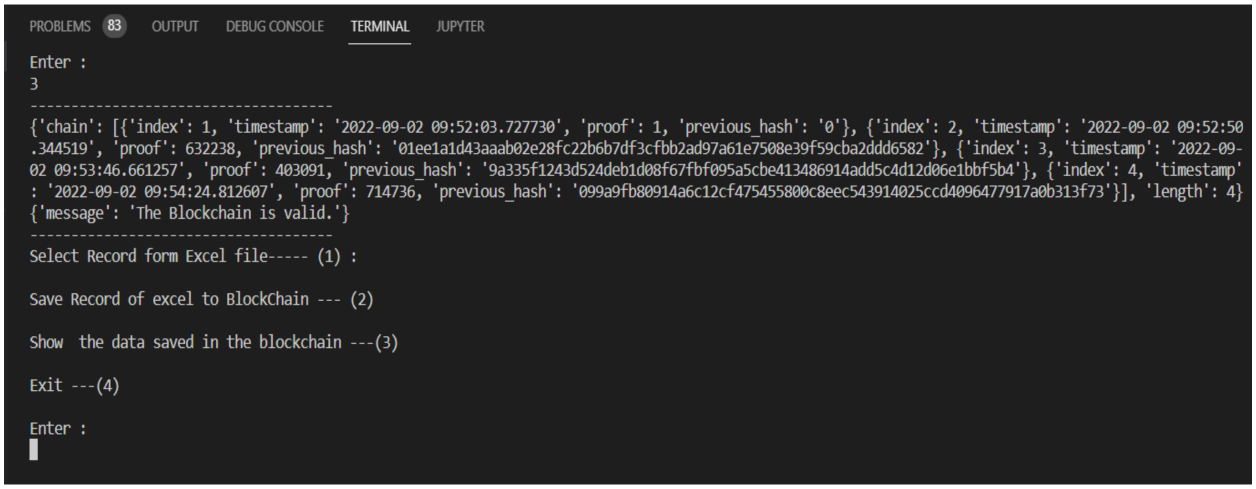


```
Select Record form Excel file----- (1) :

Save Record of excel to BlockChain --- (2)

Show  the data saved in the blockchain ---(3)

Exit ---(4)

Enter :
3
------------------------------------
{'chain': [{'index': 1, 'timestamp': '2022-08-08 11:20:50.316814', 'proof': 1, 'previous_hash': '0'}, {'index': 2, 'timestamp': '2022-08-08 11:25:02
.818494', 'proof': 632238, 'previous_hash': 'e045569020fdd03b3e8b701153f420cf5b6bddc4df40c5f4eb42dfa19c4d7f4d'}], 'length': 2}
{'message': 'The Blockchain is valid.'}
------------------------------------
```

Figure 34 Data format saved in blockchain displayed in the console

3.3.4. Display of different rows added to blockchain in console

Here we added the rows 1,20,1340 to our blockchain with this information. By entering 3, in the console, we can see the records of the rows saved in the blockchain.



```
Enter :
3
-------------------------------------
{'chain': [{'index': 1, 'timestamp': '2022-09-02 09:52:03.727730', 'proof': 1, 'previous_hash': '0'}, {'index': 2, 'timestamp': '2022-09-02 09:52:50
.344519', 'proof': 632238, 'previous_hash': '01ee1a1d43aaab02e28fc22b6b7df3cfbb2ad97a61e7508e39f59cba2ddd6582'}, {'index': 3, 'timestamp': '2022-09-
02 09:53:46.661257', 'proof': 403091, 'previous_hash': '9a335f1243d524deb1d08f67fbf095a5cbe413486914add5c4d12d06e1bbf5b4'}, {'index': 4, 'timestamp'
: '2022-09-02 09:54:24.812607', 'proof': 714736, 'previous_hash': '099a9fb80914a6c12cf475455800c8eec543914025ccd4096477917a0b313f73'}], 'length': 4}
{'message': 'The Blockchain is valid.'}
-------------------------------------
Select Record form Excel file----- (1) :

Save Record of excel to BlockChain --- (2)

Show  the data saved in the blockchain ---(3)

Exit ---(4)

Enter :
```

Figure 35 All the records saved in the blockchain

4. Evaluation

4.1. Performance and Scalability in Consensus Algorithms

Permissionless blockchains need slow block creation to keep up with the propagation speed of nodes in the network. Permissioned blockchains, on the other hand, have much lower latency, but have serious scalability issues. The network overhead introduced by the voting mechanism means that permissioned blockchains can only scale to a few hundred nodes. The worst-case complexity of permissioned blockchains is $O(N^2)$ compared to $O(N)$ as the worst-case complexity of permission-less blockchains. This limits the usability of permissioned blockchains for IoT [46]. Through the virtues of publicly anonymous accessibility and decentralization, permissionless blockchains are better suited to industry wide IoT applications. Permissioned blockchains are more suited to enterprise solutions due to their higher degree of control and permission granting capabilities.

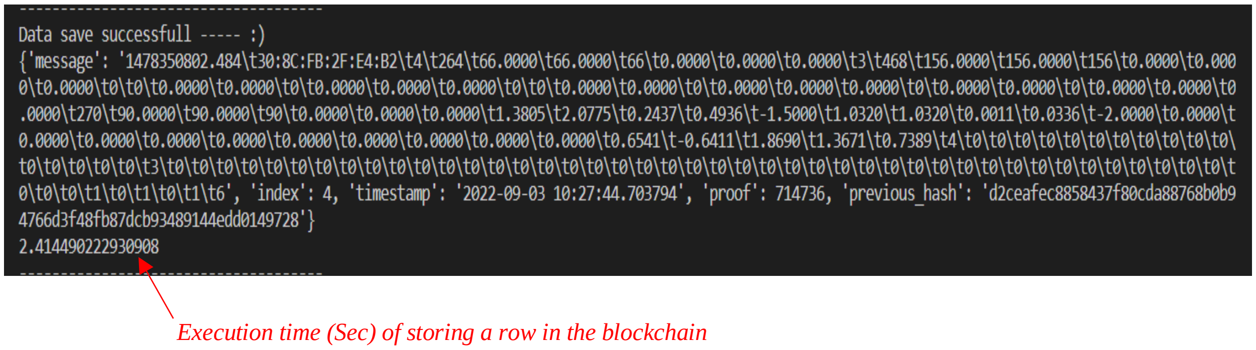
4.2. Considering the time needed to insert a new record in the blockchain

```
#save data in new block
    def mine_block(self,data):
        start_time = time.time()
        previous_block = self.print_previous_block()
        previous_proof = previous_block['proof']
        proof = self.proof_of_work(previous_proof)
        previous_hash = self.hash(previous_block)
        block = self.create_block(proof, previous_hash)

        response = {'message': data,
                    'index': block['index'],
                    'timestamp': block['timestamp'],
                    'proof': block['proof'],
                    'previous_hash': block['previous_hash']}
        print(response)
        print((time.time() - start_time))
```

Figure 36 Time of execution for inserting new record in the blockchain

4.2.1. How much time it takes to insert a new record in the blockchain

In this part, we discuss about the time needed to store a new record to our blockchain. Fist, we want to know how much time it takes to create the first block. As we can see in the figure 37, the creation time of the first block is about 2.4144902229 (sec) which results in creating the blockchain.



```
------------------------------------
Data save successfull ----- :)
{'message': '1478350802.484\t30:8C:FB:2F:E4:B2\t4\t264\t66.0000\t66.0000\t66\t0.0000\t0.0000\t0.0000\t3\t468\t156.0000\t156.0000\t156\t0.0000\t0.000
0\t0.0000\t0\t0\t0.0000\t0.0000\t0\t0.0000\t0.0000\t0.0000\t0\t0\t0.0000\t0.0000\t0\t0.0000\t0.0000\t0.0000\t0\t0.0000\t0.0000\t0\t0.0000\t0.0000\t0
.0000\t270\t90.0000\t90.0000\t90\t0.0000\t0.0000\t0.0000\t1.3805\t2.0775\t0.2437\t0.4936\t-1.5000\t1.0320\t1.0320\t0.0011\t0.0336\t-2.0000\t0.0000\t
0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.6541\t-0.6411\t1.8690\t1.3671\t0.7389\t4\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\
t0\t0\t0\t0\t0\t3\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t
0\t0\t0\t1\t0\t1\t0\t1\t6', 'index': 4, 'timestamp': '2022-09-03 10:27:44.703794', 'proof': 714736, 'previous_hash': 'd2ceafec8858437f80cda88768b0b9
4766d3f48fb87dcb93489144edd0149728'}
2.414490222930908
------------------------------------
```

Figure 37 execution time for saving the 1th row of data into the blockchain

In the figure 38, we aim to compare the original record row numbers (as the rows) to the time it takes to be stored in the blockchain.

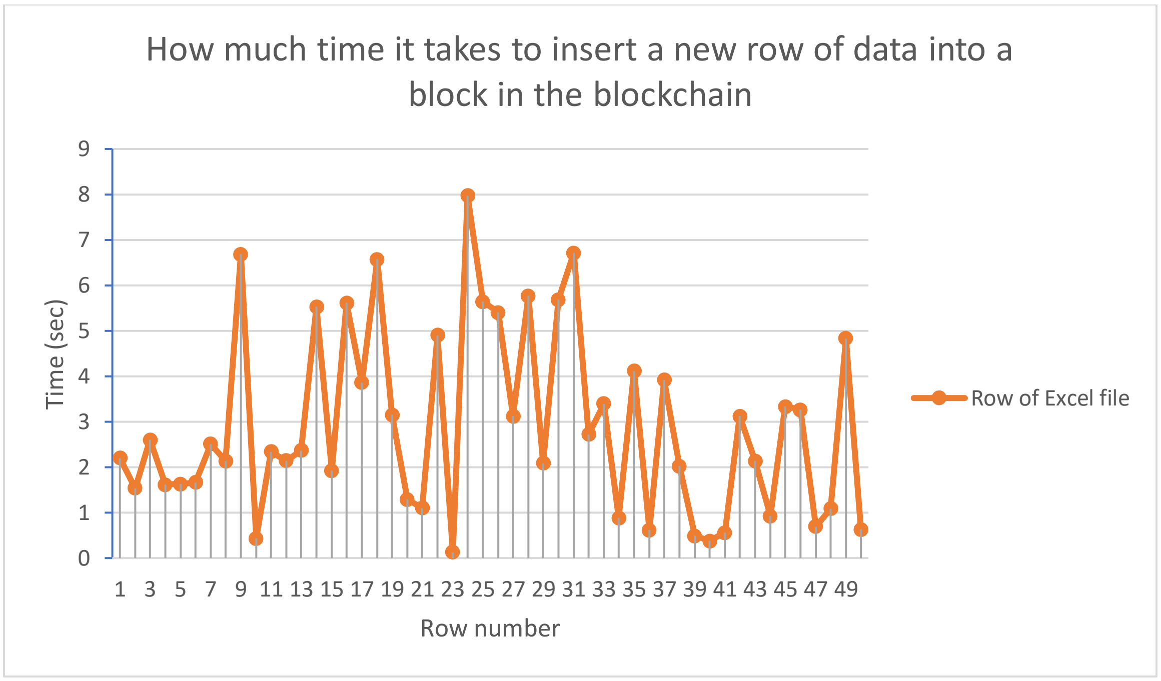


Figure 38 Time of saving new row into the blockchain

The figure 39 shows the number of characters of each row in which we want to save in the blocks of the blockchain. Here, we can count the number of characters of each row of excel file (for the first 50 records), which is going to be inserted in the blockchain. This is a partial sample of the excel file (1th to 50th row).

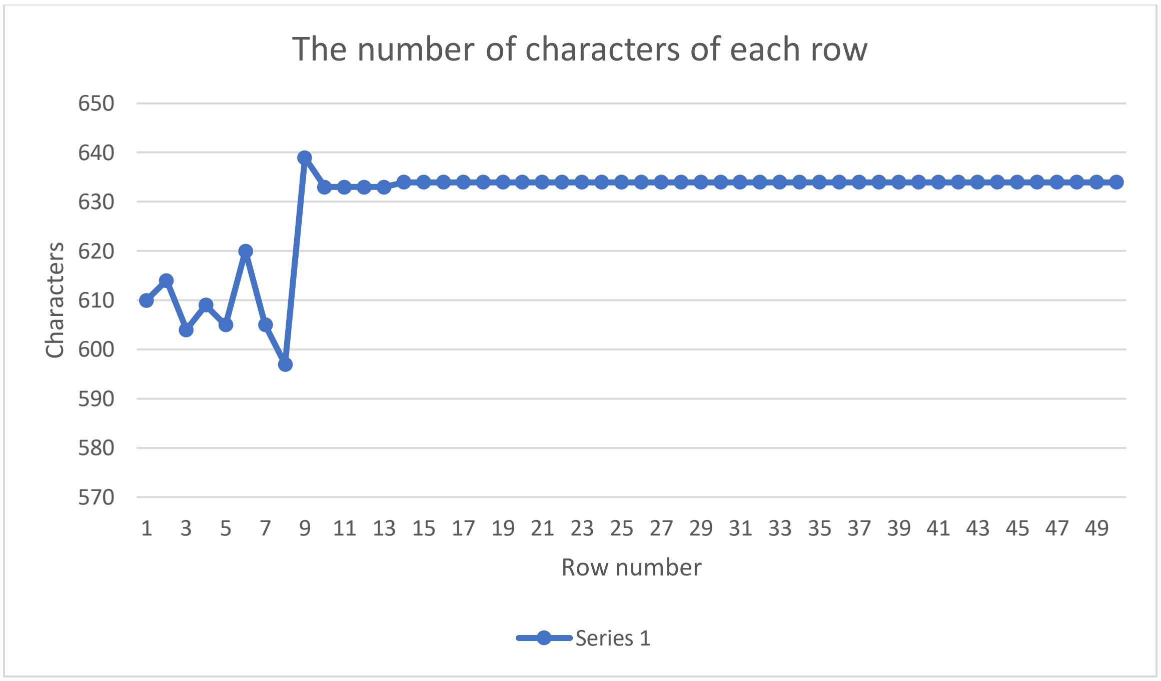


Figure 39 How many characters each record of excel file has in order to be inserted in the blockchain

As we can understand from the above figures 38 and 39, the creation time for each block to be inserted in to the blockchain, is different from other blocks because the blocks are different in size. It is obvious that the length of each record is different so the time to change the original characters to hash codes are different which results in different block creation time. The time for the creation of the blocks in the blockchain depends on several issues. Some of the issues are as following:

- The processing time and the computing power to create the hashes of the characters is different for each row of the original data.
- The mathematical calculation to insert a new row of characters in each block takes different times depending on the characters it contains.
- The current status of the Operating System and how many soft wares are running.

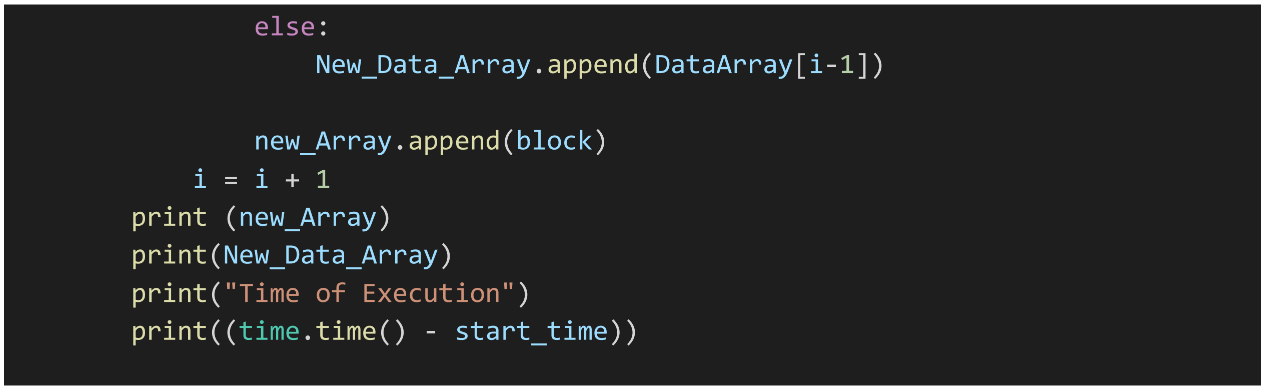
4.3. Retrieving the data from the blockchain

In this part, we aim to know the original data which are saved to the blockchain in the specified timestamps. In other words, the user wants to know the actions which are taken in the specific time stamps duration.

The number of timestamps	**Time**
The first timestamp	Between "2022-09-06 14:54:03" and "2022-09-07 16:54:03"
The second timestamp	Between "2022-09-07 08:14:11" and "2022-09-07 19:35:20"
The third timestamp	Between "2022-09-03 08:14:11" and "2022-09-08 09:00:20"

Table 3 the specified bounded timestamps for the following evaluations

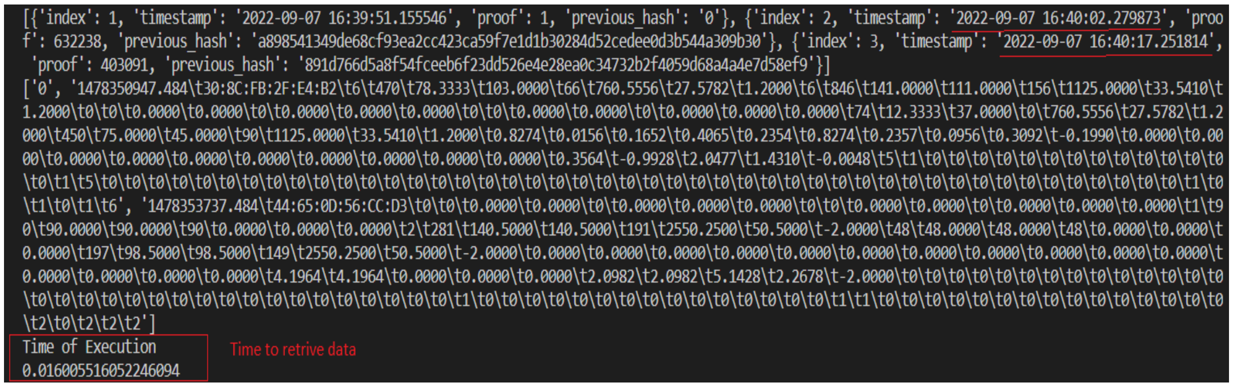
```
def limitchain(self,DataArray):

        start_time = time.time()
        Entry = "2022-09-06 14:54:03"
        Entry1 = "2022-09-07 16:54:03"
        Time_Entry = datetime.datetime.strptime(Entry, '%Y-%m-%d %H:%M:%S')
        Time_Entry1 = datetime.datetime.strptime(Entry1, '%Y-%m-%d %H:%M:%S')

        i = 0

        length = len(self.chain);

        new_Array = []
        New_Data_Array = []

        while i < length:
            block = self.chain[i]
            if (Time_Entry <(datetime.datetime.strptime(block['timestamp'], '%Y-
%m-%d %H:%M:%S.%f')) < Time_Entry1):
                if i == 0 :
                    New_Data_Array.append("0")
```



```
            else:
                New_Data_Array.append(DataArray[i-1])

            new_Array.append(block)
        i = i + 1
    print (new_Array)
    print(New_Data_Array)
    print("Time of Execution")
    print((time.time() - start_time))
```

Figure 40 Retrieve the original data between two specified time stamps:"2022-09-06 14:54:03" and "2022-09-07 16:41:03"

In the figure 41, we want to get back the information in the blockchain between 2 different timestamps with the time it takes to retrieve data. In a real scenario, we want to know the actions in which a user did between 2 different times.



```
[{'index': 1, 'timestamp': '2022-09-07 16:39:51.155546', 'proof': 1, 'previous_hash': '0'}, {'index': 2, 'timestamp': '2022-09-07 16:40:02.279873', 'proo
f': 632238, 'previous_hash': 'a898541349de68cf93ea2cc423ca59f7e1d1b30284d52cedee0d3b544a309b30'}, {'index': 3, 'timestamp': '2022-09-07 16:40:17.251814',
 'proof': 403091, 'previous_hash': '891d766d5a8f54fceeb6f23dd526e4e28ea0c34732b2f4059d68a4a4e7d58ef9'}]
['0', '1478350947.484\t30:8C:FB:2F:E4:B2\t6\t470\t78.3333\t103.0000\t66\t760.5556\t27.5782\t1.2000\t6\t846\t141.0000\t111.0000\t156\t1125.0000\t33.5410\t
1.2000\t0\t0\t0.0000\t0.0000\t0\t0.0000\t0.0000\t0.0000\t0\t0\t0.0000\t0.0000\t0\t0.0000\t0.0000\t0.0000\t74\t12.3333\t37.0000\t0\t760.5556\t27.5782\t1.2
000\t450\t75.0000\t45.0000\t90\t1125.0000\t33.5410\t1.2000\t0.8274\t0.0156\t0.1652\t0.4065\t0.2354\t0.8274\t0.2357\t0.0956\t0.3092\t-0.1990\t0.0000\t0.00
00\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.3564\t-0.9928\t2.0477\t1.4310\t-0.0048\t5\t1\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0
\t0\t1\t5\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t1\t0
\t1\t0\t1\t6', '1478353737.484\t44:65:0D:56:CC:D3\t0\t0\t0.0000\t0.0000\t0\t0.0000\t0.0000\t0.0000\t0\t0\t0.0000\t0.0000\t0\t0.0000\t0.0000\t0.0000\t1\t9
0\t90.0000\t90.0000\t90\t0.0000\t0.0000\t0.0000\t2\t281\t140.5000\t140.5000\t191\t2550.2500\t50.5000\t-2.0000\t48\t48.0000\t48.0000\t48\t0.0000\t0.0000\t
0.0000\t197\t98.5000\t98.5000\t149\t2550.2500\t50.5000\t-2.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t
0.0000\t0.0000\t0.0000\t0.0000\t4.1964\t4.1964\t0.0000\t0.0000\t0.0000\t2.0982\t2.0982\t5.1428\t2.2678\t-2.0000\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0
\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t1\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t1\t1\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0
\t2\t0\t2\t2\t2']
Time of Execution
0.016005516052246094
```

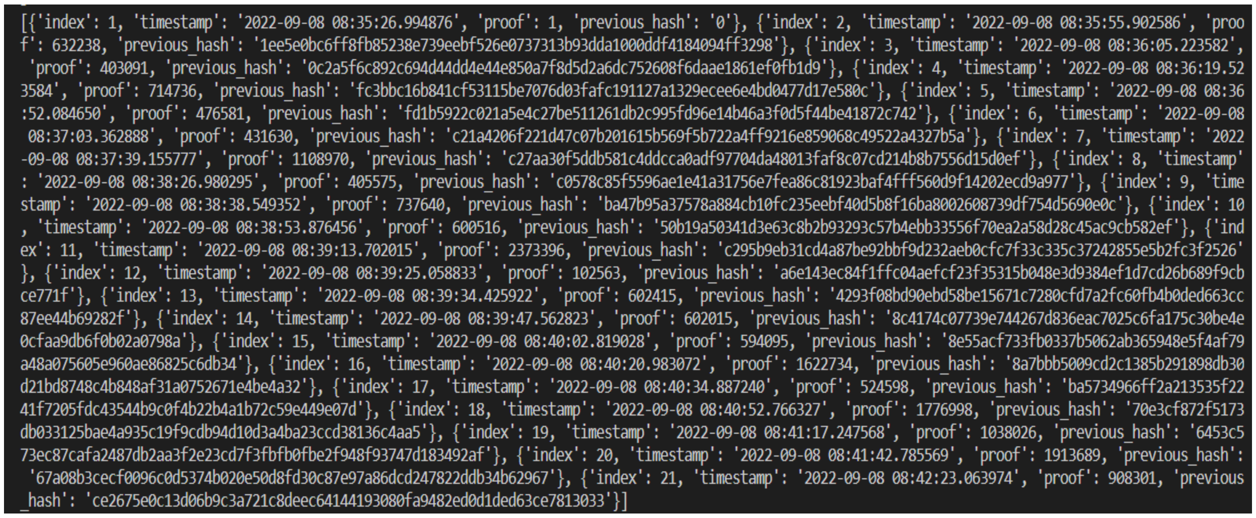
Figure 41 The retrieved data from the blockchain in the first timestamp

In the first timestamp, according to table 3 between "2022-09-06 14:54:03" and "2022-09-07 16:54:03", we retrieved two rows of the original data. The original data are of the row numbers 123 and 2345 with these original values as seen in the figure 41.

As we can see in the figure 41, the time to retrieve the data between these two timestamps is 0.016005516052246094 (sec).

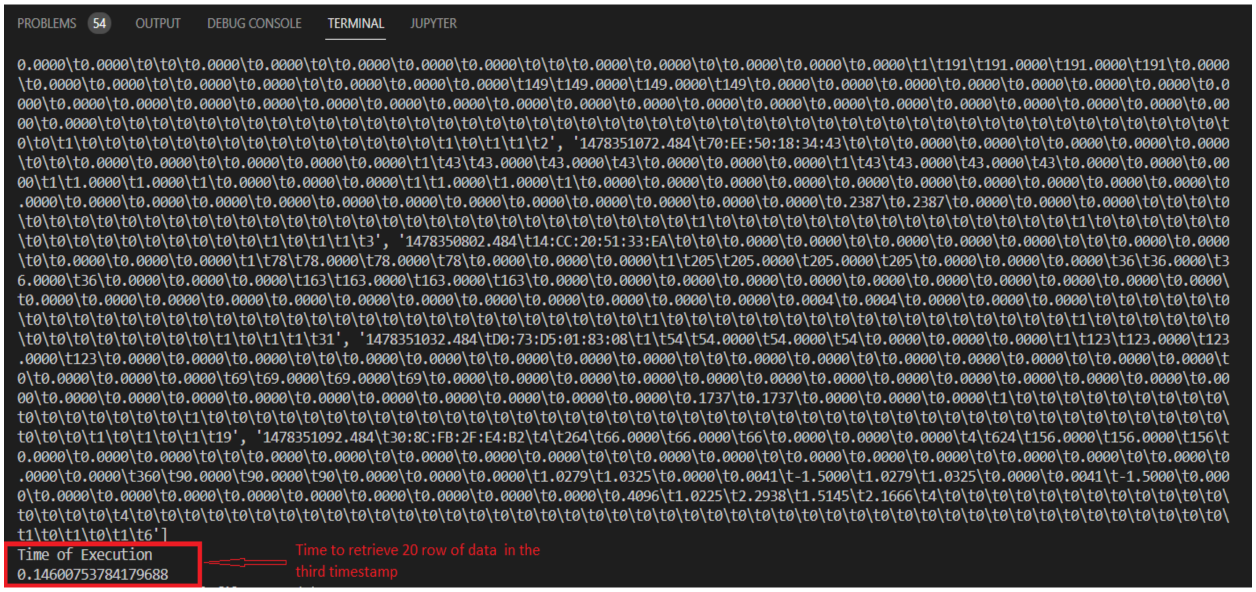
Considering that we have a sample of 100 rows of raw data in the blockchain, we can understand that from the second time stamp between "2022-09-07 08:14:11" and "2022-09-07 19:35:20", we retrieved 9 blocks of information from the row number 1301 to row number 1308.

For the third timestamp as seen in the table 3, we want to retrieve the data between the total 100 blocks of data saved in the blockchain. The time to retrieve the data is higher than the other timestamps because there are more records of data to retrieve as in the figure 42. This time, we retrieve 20 blocks of records from blockchain.



```
[{'index': 1, 'timestamp': '2022-09-08 08:35:26.994876', 'proof': 1, 'previous_hash': '0'}, {'index': 2, 'timestamp': '2022-09-08 08:35:55.902586', 'proo
f': 632238, 'previous_hash': '1ee5e0bc6ff8fb85238e739eebf526e0737313b93dda1000ddf4184094ff3298'}, {'index': 3, 'timestamp': '2022-09-08 08:36:05.223582',
 'proof': 403091, 'previous_hash': '0c2a5f6c892c694d44dd4e44e850a7f8d5d2a6dc752608f6daae1861ef0fb1d9'}, {'index': 4, 'timestamp': '2022-09-08 08:36:19.52
3584', 'proof': 714736, 'previous_hash': 'fc3bbc16b841cf53115be7076d03fafc191127a1329ecee6e4bd0477d17e580c'}, {'index': 5, 'timestamp': '2022-09-08 08:36
:52.084650', 'proof': 476581, 'previous_hash': 'fd1b5922c021a5e4c27be511261db2c995fd96e14b46a3f0d5f44be41872c742'}, {'index': 6, 'timestamp': '2022-09-08
 08:37:03.362888', 'proof': 431630, 'previous_hash': 'c21a4206f221d47c07b201615b569f5b722a4ff9216e859068c49522a4327b5a'}, {'index': 7, 'timestamp': '2022
-09-08 08:37:39.155777', 'proof': 1108970, 'previous_hash': 'c27aa30f5ddb581c4ddcca0adf97704da48013faf8c07cd214b8b7556d15d0ef'}, {'index': 8, 'timestamp'
: '2022-09-08 08:38:26.980295', 'proof': 405575, 'previous_hash': 'c0578c85f5596ae1e41a31756e7fea86c81923baf4fff560d9f14202ecd9a977'}, {'index': 9, 'time
stamp': '2022-09-08 08:38:38.549352', 'proof': 737640, 'previous_hash': 'ba47b95a37578a884cb10fc235eebf40d5b8f16ba8002608739df754d5690e0c'}, {'index': 10
, 'timestamp': '2022-09-08 08:38:53.876456', 'proof': 600516, 'previous_hash': '50b19a50341d3e63c8b2b93293c57b4ebb33556f70ea2a58d28c45ac9cb582ef'}, {'ind
ex': 11, 'timestamp': '2022-09-08 08:39:13.702015', 'proof': 2373396, 'previous_hash': 'c295b9eb31cd4a87be92bbf9d232aeb0cfc7f33c335c37242855e5b2fc3f2526'
}, {'index': 12, 'timestamp': '2022-09-08 08:39:25.058833', 'proof': 102563, 'previous_hash': 'a6e143ec84f1ffc04aefcf23f35315b048e3d9384ef1d7cd26b689f9cb
ce771f'}, {'index': 13, 'timestamp': '2022-09-08 08:39:34.425922', 'proof': 602415, 'previous_hash': '4293f08bd90ebd58be15671c7280cfd7a2fc60fb4b0ded663cc
87ee44b69282f'}, {'index': 14, 'timestamp': '2022-09-08 08:39:47.562823', 'proof': 602015, 'previous_hash': '8c4174c07739e744267d836eac7025c6fa175c30be4e
0cfaa9db6f0b02a0798a'}, {'index': 15, 'timestamp': '2022-09-08 08:40:02.819028', 'proof': 594095, 'previous_hash': '8e55acf733fb0337b5062ab365948e5f4af79
a48a075605e960ae86825c6db34'}, {'index': 16, 'timestamp': '2022-09-08 08:40:20.983072', 'proof': 1622734, 'previous_hash': '8a7bbb5009cd2c1385b291898db30
d21bd8748c4b848af31a0752671e4be4a32'}, {'index': 17, 'timestamp': '2022-09-08 08:40:34.887240', 'proof': 524598, 'previous_hash': 'ba5734966ff2a213535f22
41f7205fdc43544b9c0f4b22b4a1b72c59e449e07d'}, {'index': 18, 'timestamp': '2022-09-08 08:40:52.766327', 'proof': 1776998, 'previous_hash': '70e3cf872f5173
db033125bae4a935c19f9cdb94d10d3a4ba23ccd38136c4aa5'}, {'index': 19, 'timestamp': '2022-09-08 08:41:17.247568', 'proof': 1038026, 'previous_hash': '6453c5
73ec87cafa2487db2aa3f2e23cd7f3fbfb0fbe2f948f93747d183492af'}, {'index': 20, 'timestamp': '2022-09-08 08:41:42.785569', 'proof': 1913689, 'previous_hash':
 '67a08b3cecf0096c0d5374b020e50d8fd30c87e97a86dcd247822ddb34b62967'}, {'index': 21, 'timestamp': '2022-09-08 08:42:23.063974', 'proof': 908301, 'previous
_hash': 'ce2675e0c13d06b9c3a721c8deec64144193080fa9482ed0d1ded63ce7813033'}]
```

Figure 42 Rows retrieved for the third timestamp (in the case we have 100 rows saved in the blockchain)



```
0.0000\t0.0000\t0\t0\t0.0000\t0.0000\t0\t0.0000\t0.0000\t0.0000\t0\t0\t0.0000\t0.0000\t0\t0.0000\t0.0000\t0.0000\t1\t191\t191.0000\t191.0000\t191\t0.0000
\t0.0000\t0.0000\t0\t0.0000\t0.0000\t0\t0.0000\t0.0000\t0.0000\t149\t149.0000\t149.0000\t149\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0
000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.00
00\t0.0000\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t
0\t0\t1\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t1\t0\t1\t1\t2', '1478351072.484\t70:EE:50:18:34:43\t0\t0\t0.0000\t0.0000\t0\t0.0000\t0.0000\t0.0000
\t0\t0\t0.0000\t0.0000\t0\t0.0000\t0.0000\t0.0000\t1\t43\t43.0000\t43.0000\t43\t0.0000\t0.0000\t0.0000\t1\t43\t43.0000\t43.0000\t43\t0.0000\t0.0000\t0.00
00\t1\t1.0000\t1.0000\t1\t0.0000\t0.0000\t0.0000\t1\t1.0000\t1.0000\t1\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0
.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.2387\t0.2387\t0.0000\t0.0000\t0.0000\t0\t0\t0\t0
\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t1\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t1\t0\t0\t0\t0\t0\t0\t0
\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t1\t0\t1\t1\t3', '1478350802.484\t14:CC:20:51:33:EA\t0\t0\t0.0000\t0.0000\t0\t0.0000\t0.0000\t0.0000\t0\t0\t0.0000\t0.0000
\t0\t0.0000\t0.0000\t0.0000\t1\t78\t78.0000\t78.0000\t78\t0.0000\t0.0000\t0.0000\t1\t205\t205.0000\t205.0000\t205\t0.0000\t0.0000\t0.0000\t36\t36.0000\t3
6.0000\t36\t0.0000\t0.0000\t0.0000\t163\t163.0000\t163.0000\t163\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\
t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0004\t0.0004\t0.0000\t0.0000\t0.0000\t0\t0\t0\t0\t0\t0
\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t1\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t1\t0\t0\t0\t0\t0\t0\t0
\t0\t0\t0\t0\t0\t0\t0\t0\t1\t0\t1\t1\t31', '1478351032.484\tD0:73:D5:01:83:08\t1\t54\t54.0000\t54.0000\t54\t0.0000\t0.0000\t0.0000\t1\t123\t123.0000\t123
.0000\t123\t0.0000\t0.0000\t0.0000\t0\t0\t0.0000\t0.0000\t0\t0.0000\t0.0000\t0.0000\t0\t0\t0.0000\t0.0000\t0\t0.0000\t0.0000\t0.0000\t0\t0.0000\t0.0000\t
0\t0.0000\t0.0000\t0.0000\t69\t69.0000\t69.0000\t69\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.00
00\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.1737\t0.1737\t0.0000\t0.0000\t0.0000\t1\t0\t0\t0\t0\t0\t0\t0\t0\
t0\t0\t0\t0\t0\t0\t0\t1\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\
t0\t0\t0\t1\t0\t1\t0\t1\t19', '1478351092.484\t30:8C:FB:2F:E4:B2\t4\t264\t66.0000\t66.0000\t66\t0.0000\t0.0000\t0.0000\t4\t624\t156.0000\t156.0000\t156\t
0.0000\t0.0000\t0.0000\t0\t0\t0.0000\t0.0000\t0\t0.0000\t0.0000\t0.0000\t0\t0\t0.0000\t0.0000\t0\t0.0000\t0.0000\t0.0000\t0\t0.0000\t0.0000\t0\t0.0000\t0
.0000\t0.0000\t360\t90.0000\t90.0000\t90\t0.0000\t0.0000\t0.0000\t1.0279\t1.0325\t0.0000\t0.0041\t-1.5000\t1.0279\t1.0325\t0.0000\t0.0041\t-1.5000\t0.000
0\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.4096\t1.0225\t2.2938\t1.5145\t2.1666\t4\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\
t0\t0\t0\t0\t4\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\
t1\t0\t1\t0\t1\t6']
Time of Execution
0.14600753784179688
```

Figure 43 time to retrieve data in the third timestamp

In the following chart, we can see the time is needed to retrieve the original data from blockchain for different time stamps. Also, we can see how many rows are retrieved in those specific different timestamps in figure 44.

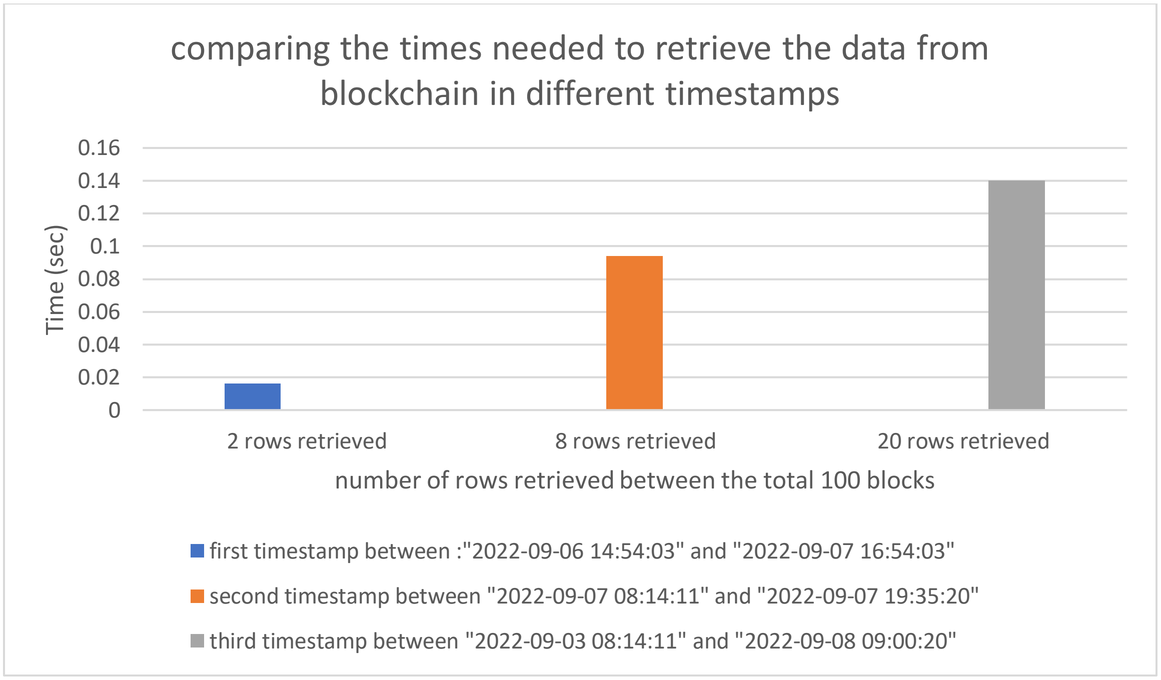


Figure 44 comparing the time it takes to retrieve data in different timestamps

In the next part in the figure 45, we want to calculate the total number of characters (the original length of data) in these 3 different time stamps. As the length of the block increases, the time to retrieve the data will increase too.

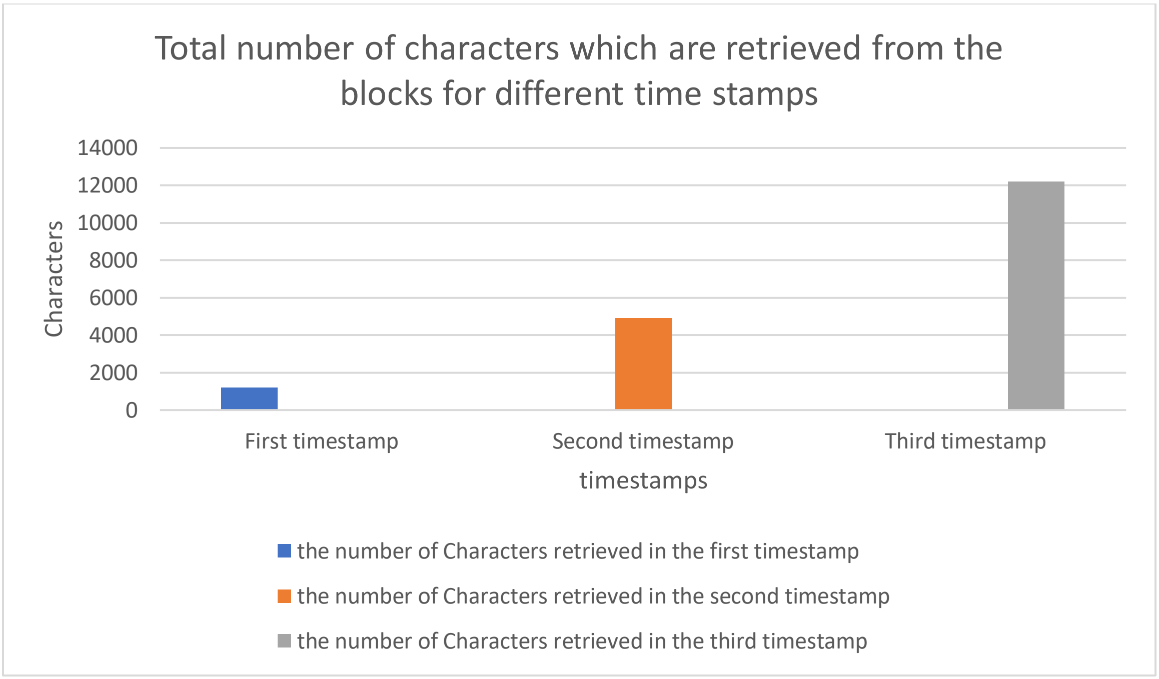
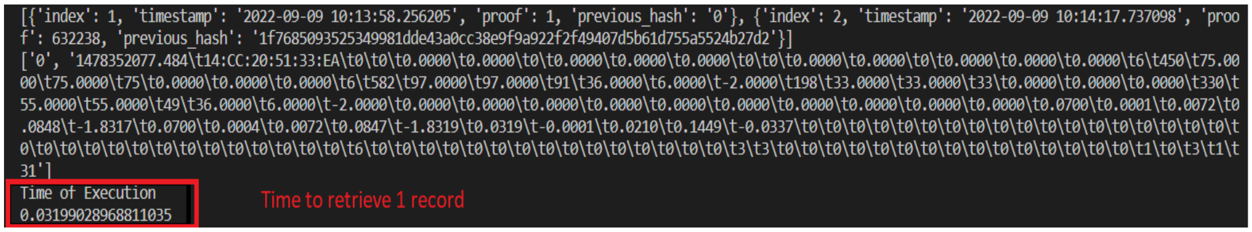


Figure 45 The length of original data in different timestamps

4.3.1. The time is need to retrieve 1 record in different time stamps

In this part, we defined 3 different timestamps to retrieve the data in these three timestamps. For each timestamp, just one row of record in the blockchain is being extracted. In the figure 46, It shows the retrieved data of the row number 1000 of the original data in the specified timestamp.



```
[{'index': 1, 'timestamp': '2022-09-09 10:13:58.256205', 'proof': 1, 'previous_hash': '0'}, {'index': 2, 'timestamp': '2022-09-09 10:14:17.737098', 'proo
f': 632238, 'previous_hash': '1f7685093525349981dde43a0cc38e9f9a922f2f49407d5b61d755a5524b27d2'}]
['0', '1478352077.484\t14:CC:20:51:33:EA\t0\t0\t0.0000\t0.0000\t0\t0.0000\t0.0000\t0.0000\t0\t0\t0.0000\t0.0000\t0\t0.0000\t0.0000\t0.0000\t6\t450\t75.00
00\t75.0000\t75\t0.0000\t0.0000\t0.0000\t6\t582\t97.0000\t97.0000\t91\t36.0000\t6.0000\t-2.0000\t198\t33.0000\t33.0000\t33\t0.0000\t0.0000\t0.0000\t330\t
55.0000\t55.0000\t49\t36.0000\t6.0000\t-2.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0000\t0.0700\t0.0001\t0.0072\t0
.0848\t-1.8317\t0.0700\t0.0004\t0.0072\t0.0847\t-1.8319\t0.0319\t-0.0001\t0.0210\t0.1449\t-0.0337\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t
0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t6\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t3\t3\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t0\t1\t0\t3\t1\t
31']
Time of Execution
0.03199028968811035
```

Figure 46 Retrieve 1 record between "2022-09-09 08:14:11" and "2022-09-09 10:15:20"

In the figure 47, while the number of characters in the row 1000 has the largest length, It doesn’t take the most time to retrieve the data related comparing to other original block sizes.

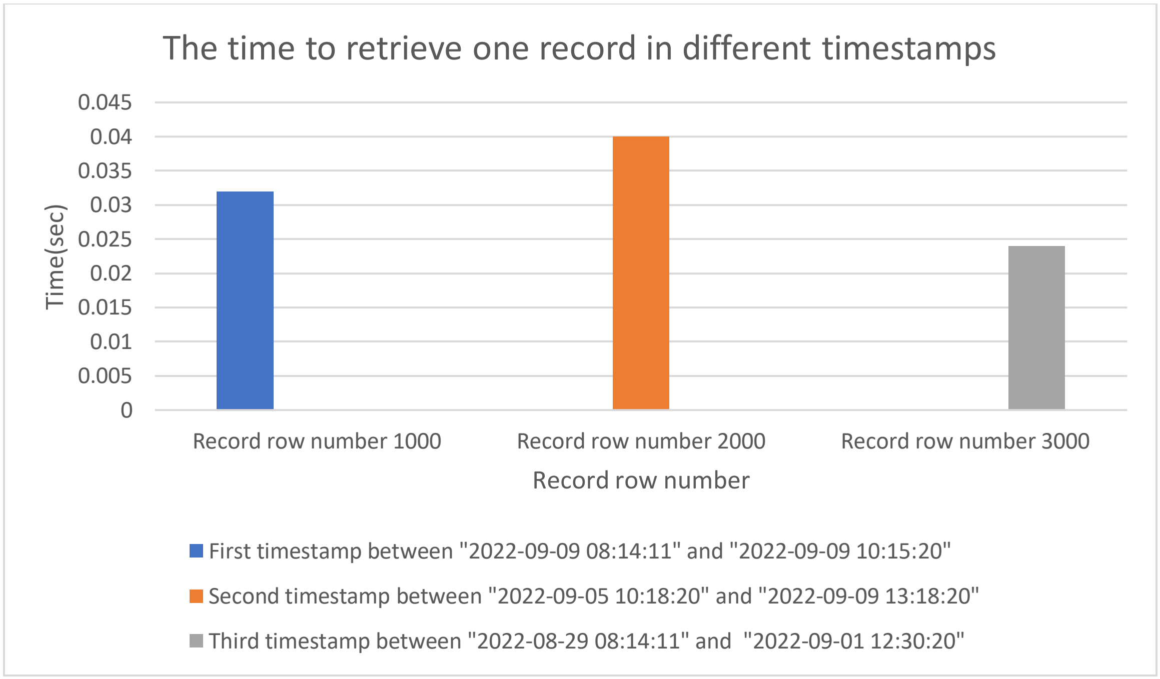


Figure 47 Time to retrieve one record from blockchain

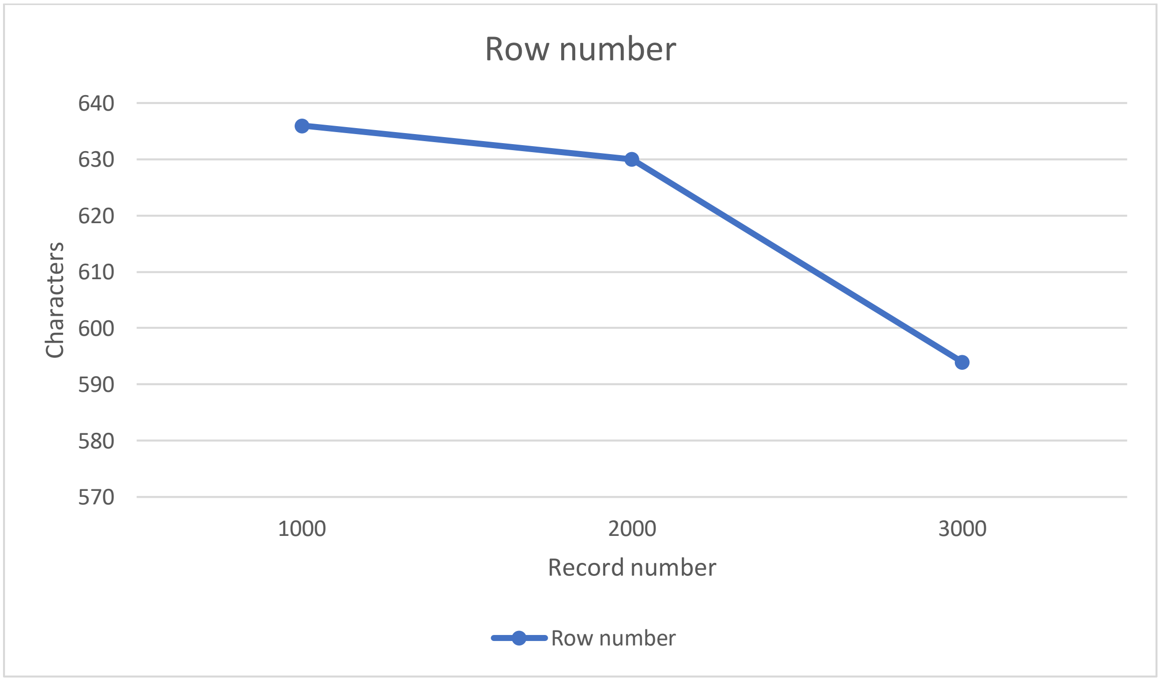


Figure 48 How many characters each specified row has

4.4. Comparing Proof of Work and Proof of Stake

The chances of a successful double-spending attack decrease when a transaction receives confirmation in a PoW-based currency and depend on the amount of mining power possessed by the attacker [55]. To decrease the risk of funds double spending, it is recommended to wait for a certain number of confirmations. Additionally, For both of consensuses (proof of work and proof of stake), some types of attacks such as DoS attack is common. The purpose of DoS attack is to disrupt the normal work of the cryptocurrency network by flooding the nodes. A Sybil attack disrupts a network by creating a series of rogue nodes. Whether a network is susceptible to DoS and Sybil attacks also depends on details of the network protocol. There is no reasons for PoS to make it less susceptible to this kind of attack when it is compared to PoW. Selfish mining is inherent in proof of working consensus. In selfish mining, an attacker selectively reveals mined blocks to waste an honest miner's computing resources. Attacks are ineffective against PoS currencies as block generation does not use expensive resources. On the other hand, there are no known examples of successful selfish bitcoin mining attacks, and some studies have argued that explanations for attacks are based on false assumptions.

For PoW consensus, the level of vulnerability to attacks can be easily predicted based on the overall hash rate of the system [56] . In the case of PoS systems, there is no equivalent scale of the network “health status”:

- Systems are vulnerable to blockchain fork-based attacks if stakes are evenly distributed among many users.
- If there are users with large stakes, they can disrupt the operation of the network (for example, by censoring transactions).

Type of attack	Vulnerabiliy		
	PoW	PoS	Delegated PoS
Short range attack(e.g., bribe)	–	+	–
Long range attack	–	+	+

Coin age accumulation attack	−	+/–	−
Pre computing attack	−	+	−
Denial of service	+	+	+
Sybil attack	+	+	+
Selfish mining	+/–	−	−

Table 4 The vulnerability of proof of work and proof of stake consensus mechanisms to attack types

A pure proof-of-stake approach poses significant security threats that cannot be replicated by any proof-of-work system (including Bitcoin). These problems are inherent in Proof of Stake algorithms, since the Proof of Stake consensus is not fixed in the physical world (see Proof of Work Hashing Device) [57] . For this reason, virtually all currencies that rely on proof-of-stake use additional mechanisms to solve security problems.

Unlike Proof of Work, Proof of Stake consensus is not objective. The state of a PoS system cannot be reliably determined by a new user based solely on protocol rule and block lists and other network messages received from peers. In order to prevent long range forks of the blockchain, a proof of stake system needs to implement weak subjectivity by combining protocol rules with social-driven security [58] . The social component of PoS systems weakens decentralization and mathematical robustness. A recent development in

Proof of Stake is the delegated system. Although these systems solve some key problems in simple PoS implementations, they have not yet been widely deployed, making their security difficult to assess. Still, delegated PoS solves the "nothing is at stake" problem and prevents short-range attacks on your system. [58] .

5. Conclusion and future work

5.1. Conclusion

This work at first collects traffic features from IoT devices for forensic purposes. We proposed a tool to collect traffic features to be installed in the access point and controlled through the LuCI web interface, part of the OpenWrt services. Then, to show possible uses of the tool, we have presented two application cases with corresponding experimental results in which the output of the tool is used to train some Machine Learning classifiers for different purposes and to be stored in the blockchain. Blockchain has been hailed as a channel for decentralizing the IoT as it has provided a truly democratic and decentralized structure for performing data transactions. In his traditional IoT architecture, centralized third parties provide critical services such as authentication, authorization, access control and data management. Decentralized services using blockchain have the potential to fundamentally change IoT service delivery without relying on central intermediaries. This paper presents a blockchain-based framework for IoT data that aims to provide a secure, decentralized system for storing IoT data.

The framework presented in this work includes proof-of-work consensus within blockchain architectures to scale blockchain-based security and protect IoT Edge's large attack surface. The framework is agnostic of the blockchain platform used if there are permissionless blockchains like Ethereum. Therefore, within this blockchain architecture, every row of an Excel file can be easily and securely stored with a hash. Applying the proposed architecture to IoT data shows its application to general traceability. A traceability system was implemented on Ethereum to emphasize the framework's agnosticity to the underlying blockchain platform.

For the purpose of traceability solutions, the permissionless Ethereum blockchain offers to co-operate to have secure data by hashing. Consensus deployed on a blockchain-based framework allows for tailored attacks on IoT data. To demonstrate this, we implemented a framework via the Distributed IoT Data Framework. The implementation on the Ethereum platform allowed us to observe how the framework behaves in real-life scenarios. Our results highlight the potential of injecting data into blockchains via existing decentralized blockchains.

We demonstrated the applicability of this framework in securing the data. Functionally, periodic hashes of the data being streamed can be stored within blockchain data base as proof of integrity. We made important observations by conducting a performance analysis.

Our results showed that on one hand, the advantage of using the Proof of Work consensus algorithm is that it has high level of security, decentralization and accepted level of scalability.

On the other hand, the main disadvantage of this consensus is that the functions of mining and validating the blocks waste a lot of energy. Moreover, the speed and success rate of this hash function highly depends on the computational abilities of the hardware running the hash [59] . Although the complexity of hash functions is scalable, solving this puzzle takes time due to the complexity of solving hash functions. Therefore, this algorithm is not suitable for large and fast-growing networks that require many transactions. every second [60]. So, the cons of Proof of Work can be as following:

- Less throughput
- High block creation time
- Energy inefficiency
- Special hardware dependency
- High computational cost
- Extensive bandwidth requirements.

In summary, the blockchain-based IoT framework presented in this work takes steps to build a scalable and secure structure for IoT data communication without the need for a centralized trusted authority. To do. The vision of decentralized IoT is one that does not rely on centralized authorities. Blockchain continues to stimulate research progress towards realizing this vision by offering a potential and fundamental paradigm shift in how the Internet of Things will be organized in the future.

5.2. Future work

Research is a constant flow of collective consciousness, so it's important to recognize where the flow will take us next. This paper introduced his contribution to designing a decentralized architecture for IoT using blockchain. The architecture remains independent of the underlying proof-of-work consensus mechanism as long as a distributed consensus mechanism is used.

Blockchain will remain a battleground between glamor and disillusionment for the foreseeable future. We are fortunate to witness and participate in this battle. Our future work includes working on blockchain consensus in combination with a proposed framework that can improve the adaptability and scalability of blockchains for IoT.

Furthermore, there is the plan to analyze new PoW protocols. New protocols lacking rigorous security analysis are still published in top venues or implemented to process financial transactions. We will continue our line of research in PoW security analysis and expose their vulnerabilities. Due to their complexity, new techniques in reinforcement learning will be employed, which further extends our PoW-based method.

There is also an opportunity to design PoW protocols that are resistant to 51% attacks. There is a PoP (Publish or Perish) design that increases the quality of the chain at the cost of longer convergence times when the network is published and reunited. This trade-off can be avoided if a compliant miner is aware of the state of the network and can rely on its local clock. Additionally, introducing a subjective view helps the protocol defend against attackers who own more than half of the total computing power. We will design stronger protocols in this direction and prove the effectiveness of new protocols.

6. References

1. Li, S., L.D. Xu, and S. Zhao, *The internet of things: a survey.* Information Systems Frontiers, 2015. **17**(2): p. 243-259.
2. Stoyanova, M., et al., *A Survey on the Internet of Things (IoT) Forensics: Challenges, Approaches, and Open Issues.* IEEE Communications Surveys & Tutorials, 2020. **22**: p. 1191-1221.
3. Sivanathan, A., et al., *Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics.* IEEE Transactions on Mobile Computing, 2019. **18**: p. 1745-1759.
4. Alaba, F.A., et al., *Internet of Things security: A survey.* Journal of Network and Computer Applications, 2017. **88**: p. 10-28.
5. Kolias, C., et al., *DDoS in the IoT: Mirai and other botnets.* Computer, 2017. **50**: p. 80-84.
6. Bertino, E. and N. Islam, *Botnets and Internet of Things Security.* Computer, 2017. **50**: p. 76-79.
7. Cardullo, P., *'Hacking multitude' and Big Data: Some insights from the Turkish 'digital coup'.* Big Data & Society, 2015. **2**(1): p. 2053951715580599.
8. Ali, M.S., et al., *Applications of Blockchains in the Internet of Things: A Comprehensive Survey.* IEEE Communications Surveys & Tutorials, 2018. **PP**: p. 1-1.
9. Reyna, A., et al., *On blockchain and its integration with IoT. Challenges and opportunities.* Future Generation Computer Systems, 2018. **88**: p. 173-190.
10. Nakamoto, S. *Bitcoin : A Peer-to-Peer Electronic Cash System*. 2009.
11. Böhme, R., et al., *Bitcoin: Economics, Technology, and Governance.* Journal of Economic Perspectives, 2015. **29**(2): p. 213-38.
12. Pilkington, M., *Blockchain Technology: Principles and Applications*. 2016.
13. Beck, R., et al., *BLOCKCHAIN – THE GATEWAY TO TRUST-FREE CRYPTOGRAPHIC TRANSACTIONS.* 2016.
14. Nakamoto, S., *Bitcoin: A Peer-to-Peer Electronic Cash System.* Cryptography Mailing list at https://metzdowd.com, 2009.
15. Sultan, K., U. Ruhi, and R. Lakhani, *Conceptualizing Blockchains: Characteristics & Applications*. 2018.
16. Stinson, D.R.P.M., *Cryptography : Theory and Practice, Fourth Edition.* 2018.
17. Cuccuru, P., *Beyond bitcoin: an early overview on smart contracts.* International Journal of Law and Information Technology, 2017. **25**(3): p. 179-195.
18. Courtois, N., M. Grajek, and R. Naik, *The Unreasonable Fundamental Incertitudes Behind Bitcoin Mining.* 2013.
19. Khanal, Y.P., et al., *Utilizing blockchain for iot privacy through enhanced ECIES with secure hash function.* Future Internet, 2022. **14**(3): p. 77.
20. Kuznetsov, A., et al., *Performance Analysis of Cryptographic Hash Functions Suitable for Use in Blockchain.* International Journal of Computer Network & Information Security, 2021. **13**(2).
21. Gupta, G. and S. Sharma. *Enhanced SHA-192 algorithm with larger bit difference*. in *2013 International Conference on Communication Systems and Network Technologies*. 2013. IEEE.
22. Panarello, A., et al., *Blockchain and IoT Integration: A Systematic Survey.* Sensors, 2018. **18**: p. 2575.

23. Ali, M.S., et al., *Applications of Blockchains in the Internet of Things: A Comprehensive Survey*. IEEE Communications Surveys & Tutorials, 2019. **21**: p. 1676-1717.
24. Merkle, R.C. *A Digital Signature Based on a Conventional Encryption Function*. in *CRYPTO*. 1987.
25. Wood, D.D. *ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER*. 2014.
26. Jaquet-Chiffelle, D.-O., E. Casey, and J. Bourquenoud, *Tamperproof timestamped provenance ledger using blockchain technology*. Forensic Science International: Digital Investigation, 2020. **33**: p. 300977.
27. Biryukov, A., D. Khovratovich, and I. Pustogarov, *Deanonymisation of Clients in Bitcoin P2P Network*, in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. 2014, Association for Computing Machinery: Scottsdale, Arizona, USA. p. 15–29.
28. Lantz, L.C.D.S.a.O.R.M.C., *Mastering Blockchain.* 2020.
29. He, Y., et al., *A Blockchain Based Truthful Incentive Mechanism for Distributed P2P Applications.* IEEE Access, 2018. **PP**: p. 1-1.
30. Lauslahti, K., J. Mattila, and T. Seppälä, *Smart Contracts – How will Blockchain Technology Affect Contractual Practices?* 2017.
31. Omohundro, S., *Cryptocurrencies, smart contracts, and artificial intelligence.* AI Matters, 2014. **1**: p. 19-21.
32. Luu, L., et al., *Making Smart Contracts Smarter.* Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016.
33. Solat, S., P. Calvez, and F. Naït-Abdesselam, *Permissioned vs. Permissionless Blockchain: How and Why There Is Only One Right Choice.* Journal of Software, 2020. **16**: p. 95-106.
34. Hassan, M.U., M.H. Rehmani, and J. Chen, *Privacy preservation in blockchain based IoT systems: Integration issues, prospects, challenges, and future research directions.* Future Generation Computer Systems, 2019. **97**: p. 512-529.
35. Tschorsch, F. and B. Scheuermann, *Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies.* IEEE Communications Surveys & Tutorials, 2016. **18**: p. 2084-2123.
36. Dinh, T.T.A., et al., *Untangling Blockchain: A Data Processing View of Blockchain Systems.* IEEE Transactions on Knowledge and Data Engineering, 2018. **30**(7): p. 1366-1385.
37. Kus Khalilov, M.C. and A. Levi, *A Survey on Anonymity and Privacy in Bitcoin-Like Digital Cash Systems.* IEEE Communications Surveys & Tutorials, 2018. **20**: p. 2543-2585.
38. Li, Y., L. Qiao, and Z. Lv, *An optimized byzantine fault tolerance algorithm for consortium blockchain.* Peer-to-Peer Networking and Applications, 2021. **14**(5): p. 2826-2839.
39. Androulaki, E., et al., *Hyperledger fabric: a distributed operating system for permissioned blockchains.* Proceedings of the Thirteenth EuroSys Conference, 2018.
40. Liu, Y., et al., *Effective Scaling of Blockchain Beyond Consensus Innovations and Moore's Law: Challenges and Opportunities.* IEEE Systems Journal, 2022. **16**: p. 1424-1435.
41. Takefuji, Y., *Consensus algorithms in blockchain must be cared for achieving the robust system*. 2020.
42. Nguyen, G.-T. and K. Kim, *A Survey about Consensus Algorithms Used in Blockchain.* J. Inf. Process. Syst., 2018. **14**: p. 101-128.

43. Tran, T.H., P. Hoai Luan, and Y. Nakashima, *A High-Performance Multimem SHA-256 Accelerator for Society 5.0.* IEEE Access, 2021. **PP**: p. 1-1.
44. Zheng, Z., et al., *Blockchain challenges and opportunities: A survey.* International Journal of Web and Grid Services, 2018. **14**: p. 352.
45. Malone, D. and K.J. O'Dwyer, *Bitcoin Mining and its Energy Footprint*. 2014. 280-285.
46. Zheng, Z., et al., *An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends.* 2017 IEEE International Congress on Big Data (BigData Congress), 2017: p. 557-564.
47. John, K., M. O'Hara, and F. Saleh, *Bitcoin and beyond.* Annual Review of Financial Economics, 2021. **14**.
48. Kabla, A.H.H., et al., *Applicability of Intrusion Detection System on Ethereum Attacks: A Comprehensive Review.* IEEE Access, 2022.
49. Nathan, S., T. Parth, and B. Vishwanathan, *Performance Benchmarking and Optimizing Hyperledger Fabric Blockchain Platform*. 2018.
50. Makhdoom, I., et al., *Blockchain's adoption in IoT: The challenges, and a way forward.* Journal of Network and Computer Applications, 2019. **125**: p. 251-279.
51. Castro, M. and B. Liskov, *Practical Byzantine fault tolerance*, in *Proceedings of the third symposium on Operating systems design and implementation*. 1999, USENIX Association: New Orleans, Louisiana, USA. p. 173–186.
52. Mazières, D. *The Stellar Consensus Protocol: A Federated Model for Internet-level Consensus*. 2015.
53. Ismailisufi, A., et al. *A private blockchain implementation using multichain open source platform*. in *2020 24th International Conference on Information Technology (IT)*. 2020. IEEE.
54. Samaniego, M. and R. Deters, *Internet of Smart Things - IoST: Using Blockchain and CLIPS to Make Things Autonomous.* 2017 IEEE International Conference on Cognitive Computing (ICCC), 2017: p. 9-16.
55. Gervais, A., et al., *On the Security and Performance of Proof of Work Blockchains*, in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 2016, Association for Computing Machinery: Vienna, Austria. p. 3–16.
56. Nayak, K., et al., *Stubborn Mining: Generalizing Selfish Mining and Combining with an Eclipse Attack.* 2016 IEEE European Symposium on Security and Privacy (EuroS&P), 2016: p. 305-320.
57. Vashchuk, O. and R. Shuwar, *Pros and cons of consensus algorithm proof of stake. Difference in the network safety in proof of work and proof of stake.* Electronics and Information Technologies, 2018. **9**(9): p. 106-112.
58. Vashchuk, O. and R. Shuwar, *Pros and cons of consensus algorithm proof of stake. Difference in the network safety in proof of work and proof of stake.* Electronics and Information Technologies, 2018. **9**.
59. Zheng, Z., et al. *An overview of blockchain technology: Architecture, consensus, and future trends*. in *2017 IEEE international congress on big data (BigData congress)*. 2017. Ieee.
60. Alsunaidi, S.J. and F.A. Alhaidari, *A Survey of Consensus Algorithms for Blockchain Technology.* 2019 International Conference on Computer and Information Sciences (ICCIS), 2019: p. 1-6.