**Project report: Analysis of Consensus Algorithms of Public Blockchain**

Submitted Towards the Completion of Internship in Blockchain/IOT & ML

Under the Mentorship of :-

**Dr. Naveen Kumar Gupta**

*(Assistant Professor, Dept. of IT, NITJ)*



**Department of Information Technology**

**Dr B R Ambedkar National Institute of Technology**

**Jalandhar-144008, Punjab (INDIA)**

*Fall of 2024(June-July)*

(Date)

**Undertaking**

I/We declare that the project work presented in this report entitled Analysis of Consensus Algorithm of Public Blockchain, submitted to the Department of Information Technology, Dr B R Ambedkar National Institute of Technology Jalandhar, for the award of the Bachelor of Technology degree in Information Technology, is our/my original work. I/We have not plagiarized or submitted the same work for the award of any other degree. In case this undertaking is found incorrect, I/we accept that our degree may be unconditionally withdrawn.

**Sarthak Patial**

*B tech. CSE* (22010103056)

*xarthak@proton.me*

*Jawaharlal Nehru Government Engineering College, Sundar Nagar (Mandi, H.P.)*

**Nitin Kumar**

*B tech. CSE* (220100103036)

*nk.nitn2005@gmail.com*

*Jawaharlal Nehru Government Engineering College, Sundar Nagar (Mandi, H.P.)*

**Abstract:**

This report gives an analytical view of a blockchain implementation using a Python-based modules, focusing on various consensus algorithm used in blockchain like Proof of Work (Pow), Proof of Stake (PoS), and Delegated Proof of Stake (DPoS) consensus mechanisms. The data collected from the model is used for statistical and analytical study of the algorithms, comparison with real-world Bitcoin data confirms these findings, showing a correlation between difficulty and mining time used by PoW. Furthermore, PoS and DPoS mechanisms, which rely less on computational power, exhibit lower time and energy consumption compared to Pow. The study highlights the trade-offs between computational efficiency and network security, illustrating how these dynamics influence real-world blockchain implementations.

**Table of Content**

**1 Introduction**

**1.1 Blockchain**

**1.2 Need of Consensus Algorithm**

1.2.1 Act 1: Decentralization

1.2.2 Act 2: Consensus Mechanism

**2 Consensus Mechanisms**

**2.1 Various Consensus Algorithms**

**2.2 Research Issues**

**2.3 Contributions**

**2.4 Related Work / Literature Review**

**3 Peer-to-Peer Network Construction**

**3.1 System Model**

3.1.3 Model’s Process Architecture

**3.2 Network Architecture**

3.2.1 Broadcast Protocol

3.2.2 Design Considerations

**4 Data Analysis**

**4.1 Pow**

4.1.1 Introduction to Analysis

4.1.2 Data Collection & Data Setup

4.1.3 Statistical Analysis of Difficulties

**4.2 Time Analysis of Pow**

**4.3 Time/Security Analysis of Pow vs PoS, DPoS**

**5 Conclusion**

**1 Introduction** 

**1.1 Blockchain:**

A blockchain is a distributed ledger of history of transaction across a network. Transaction is an umbrella term used to describe an individual record of data exchange & ledger is the complete record of all the transaction backed up by modern cryptography. Its first description can be traced backed up-to the early 20th century where Stuart Haber and W Scott Stornetta **[1]**firstly describe the concept of cryptographically secured chain of blocks in 1991 which is not exactly the blockchain as we know today but the blockchain technology encapsulates these old ideas and many more like 1998’s Computer scientist Nick Szabo’s work on ‘bit gold’ **[2]**, a decentralized digital currency, Stefan Konst’s theory of cryptographic secured chains **[3]**, plus ideas for implementation in 2000 and many more researches.

Mostly it was treated as a theoretical concept for research study and has not been publicly or practically been used until the year 2008 when an anonymous individual / group of individuals using the pen name Satoshi Nakamoto  **[4]**, firstly proposed the 2008 white-paper, "Bitcoin: A Peer-to-Peer electronic Cash System". It was this white paper which described the invention of Bitcoin, type of a cryptocurrency which is implemented on blockchain technology and was the first ever widely used derivative of the modern blockchain technology when in the year 2009 Nakamoto  **[4]** converted the idea into reality. We will also discuss about the achievement of this white-paper in further sections what made Nakamoto [4]’s paper  **[4]** so useful that made the theoretical concept into practical and started a arms race of new technological era.

The blockchain as mentioned above an open, distributed immutable ledger that can record transactions between a network efficiently and in a verifiable and permanent way. The fundamental core of blockchain is that it works as a chain contains blocks and each block present with the list of transactions. These transactions are cryptographically secure and connected to the previous block, creating an unchangeable chain This architecture makes it impossible to change data retroactively without the alteration of all subsequent blocks, a task that would require consensus from and majority support by the network. This property of immutability and because of this property, blockchain is especially appealing to industries requiring secure and practically immutable records for transactions - finance, supply chain management, web 3.0 or healthcare being some such revolutionaries examples.

Blockchain can be useful for making transactions more secure, transparent and efficient compared to traditional centralized systems. Security is based on cryptographic safeguards to ensure the sanity of data and reducing unauthorized access. The fact of the ledger existing across a vast number of nodes means that every party in the network has access to precisely all corresponding phenomenon, and this attribute is likely helping instill trust amongst its users. Moreover, blockchain removes the intermediary from direct peer-2-peer transaction which can lower costs and lead to faster transactions. Apart from financial applications such as cryptocurrencies, blockchain is being evolved to be used in a broad array of fields like supply chain traceability, legal agreements ice., smart contracts and maybe even voting systems where the integrity and transparency are key.

This tech has bring a worldwide change as it is applied in cryptocurrencies, smart contract, decentralized apps (D-apps) and boosted economy, removed the need of central authorities and give the power to the users, so we can conclude that it has tremendous potential for transforming industries by enhancing security, transparency, and efficiency a question intuitively arises ‘If this technology has so much potential & we have the components on which it relies on like cryptography and decentralization which we have from 1960’s as a practical equipment, So why it took for such a transformative technology to take half a century to became reality’. We will discuss this question and also what extraordinary Nakamoto [4]’s research paper did that made blockchain reality of today as we know today in the next section.

**1.2   Need of Consensus Algorithm:** 

**1.2.1 Act 1: Decentralization**

  In this section we will briefly talk about what blockchain is doing at core, what is a consensus algorithm & why do we need that , we shall also talk about the question we put in the previous section, to answer these we have to look from the starting why are we using blockchain, simple answer is it provides *Immutability, Decentralization & Security* as Grant Sanderson **[5]** founder of 3Blue1Brown  beautifully put in *“it’s a clever system of decentralized trust-less verification based on some of the math born of cryptography”.*

*Immutability* refers to concept of something being unchangeable once it has been created, it comes from the cryptographic part that each block is connected to previous by some hash function mainly SHA256, as SHA256 is not a normal hash function it is a cryptographic hash so it maps any input of arbitrary length of input to a string of ones and zeroes of 256 bit length and from here also comes the 256 in the name, the output string looks totally random and is totally dependent on the input string such that if we changed any bit of the input the output will be changed completely known as the *Avalanche* effect. This property of cryptographic hash makes them fundamentally irreversible, one may argue that if we dig deeper into the fundamental of the algorithm we can trace back the input from a output, but that’s the beauty of it no one ever had done that. You might think it’s a small value, but it’s not the search space is of order of 2256possibilities.  2256is a huge number, saying its Astronomically large number gives away too much credit to the astronomy. So there is currently no other way to get the input string that will produce a predetermined output of 256 bits of ones and zero rather than checking each input one by one and matching the desired output. This property of cryptographic hash function is used everywhere in block chain from verification to digital signatures to creating chain. We will discuss this further how this works in the end of this session, for now this property provides the blockchain immutability and security.

Now comes the main part for which is why this session is created the second and most important property of blockchain: *Decentralization,* but you might think session is about the consensus algorithm and we haven't discussed a thing about it, but believe me and trust the process it will eventually make sense at the end of this session and why we are doing this report. So, continuing on decentralization, Decentralization in simple terms is the distribution or delegation of authority, power, functions, and responsibilities from a central authority to local or regional authorities. It also reduce the concentration of power and increase the participation of more localized entities in decision-making processes. So what is block chain is doing to be decentralized and its really a good question because once we try to answer it we will eventually get the answer to the main question.

While dealing with someone in a trust less environment over some exchange, here exchange refers to exchange of anything some contracts, some physical good or transactions etc. How can you be so sure that another party will stick to the protocol over there self interests and not defraud you? Well for that reason we look for a third party as a broker between the other two parties so that they can make the exchange more transparent, accountable and decreases failures, frauds in the exchange. But this makes the network centralized, here network refers to the system or environment in which similar type of exchanges happens. So whole integrity of the system is purely dependent on the sake of central authority. If the central authority is fraudulent the whole exchange network dies.

              So then we come to the decentralization where power of authority, responsibility and decision making in the hands of the users of the network. So blockchain establishes the decentralization by giving the copy of the ledger of transaction or record we call blockchain to each and every node (user). So, each user will have their own copy of the ledger, and each Node can broadcast or receive transactions / records from or to others respectively, they can also verify the transactions they receive by Public-Private key cryptography that the transaction is surely broadcasted by the user intended and no fraudulent.

So here are we at last as we know until now any one can make transactions, anyone can receive transactions and no one can make others transaction from there end. But here comes the problem how can you stop someone making there own fraudulent transaction or to be more precise let us see a example to understand this problem and our current protocol: -

 lets say n1,n2,n3 are the user we will call them nodes from now own, belong to the network N having the above mentioned protocol i.e. all nodes in N have their private ledger l1….l3 ∈L, all can send & receive transaction to & from every other node in the network N, history of ledger is the currency & also transaction that incorporate in the name of nth node can only be verified when it was digitally signed by the nth node, or if n1 node wants to send some exchange in name of transaction to any one, he will digitally sign the transaction with his private key and anyone in the network can verify it with  n1’s public key that if the transaction is verified than only  n1  is the one who can commit that transaction as in n1 receives a 5$ from n2& now n1 have to pay back the 5$ he got from n2, so if this was signed by  n1 only then it would be verified, if   n2 tries to replicate it without n1’s private key it will be near to impossible to verify it by other nodes say n3 & hence they will not conclude the transaction in there private ledger.   But what if n2 sends a $5 and broadcast it only to n1  and don’t follow the protocol, in this case according to n1 he receives  $5 of the digital currency and now is the current owner of the $5, but for the rest of the network that $5 still belongs to n2 so  n2  can spend the same $5 one more time.

This problem is also known as the double spending problem which is a fundamental issue in digital currency systems, where a single digital token can be spent more than once. And not only in the finances this duplication can be done in any exchange of transaction between two parties within a network.

To resolve this we need a mechanism or extend our protocol so that each and every node in the N network have the same ledger L. So we need something to build consensus among these nodes to resolve what ledger should we agree upon to remove faults like we mentioned above, this mechanism is also known as Consensus Mechanism and we will discuss this in upcoming part.

**1.2.1 Act 2: Consensus Mechanism** 

Until now we have created a very strong foundation on why we need of consensus in the first place, consensus is what act likes a glue and binds every component of blockchain cryptography, immutability and decentralization. And consensus is not a simple thing to establish, it’s a mess of its on, how do you set consensus between diverse structures each with their own logic, priorities, and interests. Achieving agreement across such varied elements is crucial, yet it can be quite messy and intricate. In more technical terms its like harmonizing heterogeneous systems, each with distinct protocols, computational logic, and priority schemas. Reaching agreement across these disparate elements is essential, but the process can be inherently difficult and prone to further complications.

### This problem is an old problem in game theory named as the “Byzantine General’s Problem” developed by Leslie Lamport, Robert Shostak, and Marshall Pease in 19 82 **[6]**.

Byzantine Generals Problem is an impossibility result which means that the solution to this problem has not been found yet as well as helps us to understand the importance of [blockchain](https://www.geeksforgeeks.org/blockchain-technology-introduction/). It is basically a game theory problem that provides a description of the extent to which decentralized parties experience difficulties in reaching consensus without any trusted central parties .

* The Byzantine army is divided into many battalions in this classic problem called the Byzantine General’s problem, with each division led by a general.
* The generals connect via messenger in order to agree to a joint plan of action in which all battalions coordinate and attack from all sides in order to achieve success.
* It is probable that traitors will try to sabotage their plan by intercepting or changing the messages.
* As a result, the purpose of this challenge is for all of the faithful commanders to reach an agreement without the imposters tampering with their plans.

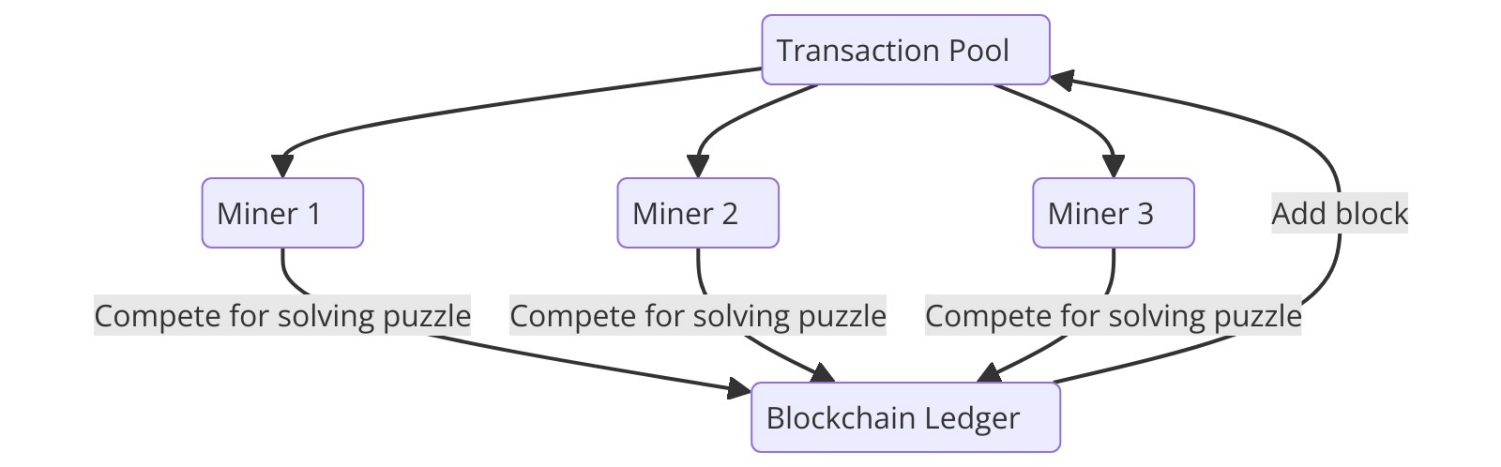
It was a seem to be impossible problem whose answer was proposed in the original paper of Nakamoto **[4]**. It was first possible solution to that problem which was practical enough to be implemented and solved the consensus problem in a decentralized system known as Proof of Work (Pow). Then after that we saw many alternates to this mechanism, which enhances some disadvantages of the Pow we will see in next section.

**2 Consensus Mechanisms:** 

**2.1 various consensus algorithms**

In blockchain technology, consensus algorithms are crucial for ensuring that all nodes in the network agree on the same state of the blockchain. Here are some of the most common consensus algorithms:

**Proof of Work (Pow)**: Proof of Work is the agreement method used by Bitcoin and other early cryptocurrencies. In this system miners compete to solve challenges requiring substantial computational power and energy. The miner who successfully solves the puzzle first can add a block to the blockchain. Earns cryptocurrency as a reward. This method ensures security and decentralization since it is challenging for any entity to dominate the networks hashing power. However, Proof of Work has faced criticism, for its energy consumption and slower transaction speeds, which make it less environmentally friendly and efficient compared to algorithms.



**Proof of Stake (PoS)**: Proof of Stake serves as an alternative consensus algorithm aimed at tackling some of the drawbacks of Pow. In PoS validators are selected to generate blocks and validate transactions based on their coin holdings and willingness to "stake" them as collateral. This approach is more energy efficient as it doesn't demand computational processes. Ethereum 2.0 and Cardano stand out as examples of platforms leveraging PoS. While PoS offers transaction speeds and reduced energy consumption it may result in the concentration of wealth given that those, with more coins wield greater influence over the network. Moreover, implementing PoS is necessitates security measures, for safeguarding the system.



**Delegated Proof of Stake (DPoS)**: Delegated Proof of Stake is simply a PoS modification that further enhances efficiency and scalability. In DPoS, coin holders vote for few individuals who validate transactions or make blocks. This technique reduces the number of actors participating in forming consensus and thereby enables quicker and more scalable operations. For instance, EOS and TRON. Nevertheless, DPoS can result to centralization since authority lies with several delegates only. It also depends on these delegates’ honesty and dependability which may act as a weakness in this sense.

**Proof of Authority (PoA)**: Proof of Authority is a consensus algorithm in which validators are chosen for their identity and reputation rather than for their computational power or coin holdings. This pattern is often seen in private or permissioned blockchains where a few validators are known and reputable. PoA offers high throughput and low latency, making it suitable for enterprise applications. However, it leads to centralization risks as it relies on a few validators. Examples include VeChain and POA Network which use PoA to achieve fast effective consensus throughputs.

**Proof of Burn (PoB)**: Proof of Burn (PoB) is a consensus mechanism, which allows participants to melt away or destroy a certain amount of cryptocurrency so as to acquire the permission to mine new blocks. By using this, the digital currency being transacted is reduced in supply increasing its value. Slim coin uses PoB as an example. It can be seen as an approach to waste since valuable resources are deliberately destroyed though the PoB requires lesser power compared to the Pow. This method has also seen low adoption rates compared to other consensus algorithms, owing in part to its unconventional approach.

**Proof of Capacity (PoC)**: Proof of Capacity permits miners to exploit their Hard Disk space for cryptocurrency mining where the miners allocate storage spaces for storing answers to solving such puzzles, and the bigger the space, the higher the probability of creating a block. This method is more power efficient than Pow and encourages use of storage devices. In fact, Burstcoin is an example of blockchain that uses PoC. Nonetheless, it necessitates large scale storage thereby resulting in hardware centralization. Even though it reduces some of PoW’s energy problems, it brings in new issues relating to storage requirements and costs related to hardware.

**Proof of Elapsed Time (PoET)**: Proof of Elapsed Time (PoET) is a consensus algorithm for permissioned blockchain networks. It uses trusted execution environments (TEEs) to make participants wait for a specific, randomly chosen amount of time before they can produce blocks. The one whose timer runs out first creates the next block. Though it can be efficient energy-wise and guarantee fair distribution of resources, implementation may pose challenges on account of possible security holes in trusted hardware; hence, its applicability is limited.

**Practical Byzantine Fault Tolerance (PBFT)**: Practical Byzantine Fault Tolerance is designed to work well in environments where the number of nodes is small. It guarantees consensus even if some of its nodes are byzantine. In PBFT, nodes in the network reach consensus via a message-passing replication strategy, based on a majority agreement that is Byzantine fault-resistant. PBFT has low latency and high throughput, which makes it appropriate for private and consortium blockchains; examples are Hyperledger Fabric and Zilliqa. However, PBFT does not scale effectively for large networks because of the need for coordination among nodes, thus limiting its usability in larger public blockchains.

From all these consensus algorithms, Pow and PoS are the most widely used, with around 90% of cryptocurrencies utilizing these methods. Therefore, it is important to identify their flaws and work on testing and improving these algorithms. We have constructed a platform in Python that allows us to test and run various consensus algorithms. This platform is efficient and easy to use with different consensus algorithms, requiring only minor changes to the program to switch between them. Our primary focus has been on testing Proof of Work (Pow) and Proof of Stake (PoS) algorithms to analyze their performance, but the platform is flexible enough to implement other consensus algorithms as well.

**2.2 Research Issues:**

Proof of Work (Pow) presents a number of vital issues and concerns, mostly because of the high level of energy that its process consumes to solve complex mathematical puzzles. This process which consumes too much energy has environmental implications as Bitcoin mining alone uses more power than some small nations (Binance Academy) . Also, Pow has made mining power to be centralized in a few big mining pools due to the huge cost associated with buying specialized mining tools such as ASICs. Such centralization undermines the rationale behind decentralized cryptocurrencies and raises the likelihood of 51% attacks where one party can potentially control most of the network’s mining power (Binance Academy). Additionally, such Pow networks have often struggled with scalability since time and resources required to crack Pow puzzles set a maximum limit on how many transactions can go through per second hence resulting in slower transaction periods characterized by high fees during peak hours (Binance Academy) (CoinMarketCap). It is very difficult to develop a decentralized system for Proof-of-Work(Pow). Creating the network as well as getting nodes connected effectively requires lots of expertise. This is a difficult task considering its technical complexity and coordination requirements; thus demands an insightful understanding of twork protocols, security, and consensus algorithms.

On the flip side, Proof of Stake (PoS) has its own set of challenges. Despite reducing the need for energy-intensive mining, PoS can result in centralization whereby only a few validators with huge stakes control the network; this may expose it to an attack or manipulation and thereby undermine its security (Binance Academy) (CoinMarketCap). Another issue is how initial stakes are distributed. This can be problematic if a few entities hold a large number of coins as this gives them an uneven amount of power over the network thus affecting how fair or decentralized it is (Decrypt). In addition, to qualify as a validator, there are usually high staking requirements which lock out small investors and limit participation to wealthier individuals or organizations hence making the network less inclusive (Decrypt) (Binance Academy). Thus, these issues reveal that both Pow and PoS consensus mechanisms are not without their complexities and trade-offs, while further efforts are being made to address them for better overall security, efficiency and decentralization of blockchain networks.

Both Pow and PoS are applicable, and they have their positive and negative sides. Division between these two consensus mechanisms, Proof of Work (Pow) and Proof of Stake (PoS), entails security, reliability as well as low power consumption in the long run. Specifically, in PoW we need to find a number of leading zeros that are indicated in the hash. This means searching for nonce which is an energy-intensive task. Although less energy is utilized in PoS there is slight compromise on security. Additionally, to construct a decentralized system, the program should not be limited to single-threaded execution. Single-threaded execution means that only one task can be performed at a time, leading to significant performance bottlenecks, especially when dealing with I/O-bound or CPU-bound tasks like transaction handling and block mining. We should have the ability to perform tasks in parallel, such as receiving and sending transactions simultaneously, and sending and receiving blocks while a node continues to mine and stops only after the block is verified. This is essential for efficient mining and transaction processing.

**2.3 Contributions:**

We have constructed a decentralized system from scratch in Python. In this decentralized system, we established connections using Python's socket library, enabling nodes to communicate with each other. This communication allows for transactions to be sent from one node to another and for the verification of blocks within the network. We used Python's multiprocessing library to run three different processes simultaneously on multiple nodes (each considered a full node). We created a function that generates random transactions. These transactions are stored in a shared memory (mempool). All nodes randomly select transactions from the mempool and start the mining process on their respective systems. The first node to mine a block broadcasts the block to all other nodes All nodes in the system are full nodes, and we utilized Python's hashlib library to generate the hash of each block.

Using the multiprocessing library, we've created a system that isn't limited to single-thread execution. This enables simultaneous sending and receiving of transactions, as well as the broadcasting of blocks to other nodes. Meanwhile, other nodes can verify blocks simultaneously and continue mining until they successfully verify and append the block to their respective blockchains. Additionally, we constructed a database using the SQLite library to store the mined blocks.

And use that data for solving various research issues. We have stored the data of each block, including the time of mining (nonce) for a particular difficulty. We have analyzed the data to find out how the average time to mine a block change with an increase in difficulty, and how the average hash rate depends on system specifications. We have analyzed the effect of difficulty on the variance, standard deviation, and median of the mining time data. We examined how network difficulty relates to mining time and hash rate per second. Why is there a need to progressively increase difficulty over time, making it more challenging for miners?

We have compared Proof of Work (Pow) and Proof of Stake (PoS) based on their security, reliability, time, energy consumption, and scalability. We attempted to determine if there could be a better approach than Pow and PoS.

**2.4 Related work / Literature Review:**

We have studied various articles that discuss different consensus algorithms, along with their advantages and disadvantages. We have also read research papers addressing the challenges of these algorithms and comparing their effectiveness. Additionally, we have studied the basics of networking and the construction of a peer-to-peer setup for implementing blockchain technology Below is a list of the articles and websites that assisted us in constructing our peer-to-peer network and studying the consensus algorithms.

### Related Work:

1. **Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System [4]:**

This work introduces Bitcoin and the Proof of Work (PoW) consensus algorithm. As discussed earlier it serves as a starting point for any blockchain-related project, This paper implemented PoW and further lead the devolopment of Bitcoin

1. **Buterin, V. (2014). A Next-Generation Smart Contract and Decentralized Application Platform[7].**
2. **Larimer, D. (2014). Delegated Proof-of-Stake (DPoS)[8]: For DPoS implementation and its field work**
3. **Eyal, I., & Sirer, E. G. (2014). Majority is not Enough: Bitcoin Mining is Vulnerable [9]:** This paper explores vulnerabilities in Bitcoin's PoW system, such as selfish mining & also provides insights into security issues in PoW.

### Literature Review

1. **Dwork, C., & Naor, M. (1992). Pricing via Processing or Combatting Junk Mail [10] work on PoW implementation**
2. **King, S., & Nadal, S. (2012). PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake [11]** introduces the Proof of Stake (PoS) consensus mechanism, contrasting it with PoW and highlighting its advantages in terms of energy efficiency.
3. **Gencer, A. E., Basu, S., Eyal, I., Van Renesse, R., & Sirer, E. G. (2018). Decentralization in Bitcoin and Ethereum Networks[12]** analyzes the decentralization of Bitcoin and Ethereum, focusing on node distribution and control.
4. **Decker, C., & Wattenhofer, R. (2013). Information Propagation in the Bitcoin Network[13] helped us to study how information is propagated through bitcoin.**
5. **Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J. A., & Felten, E. W. (2015). Sok: Research Perspectives and Challenges for Bitcoin and Cryptocurrencies [14]** surveys the state of Bitcoin research and highlights various challenges in the field.

**3   PEER TO PEER NETWORK CONSTRUCTION**

**3.1   System Model**

The system is designed to replicate a simplified blockchain network on which we can different consensus algorithm, it uses multiprocessing and inter-process communication to replicate decentralization as in actual blockchain network and to manage transactions, mine blocks, and maintain consensus across nodes. The model employs a list-based data structure to represent blocks and the blockchain, facilitating the simulation of core blockchain operations such as transaction handling, block mining, and distributed consensus. In the development of the blockchain project, several key libraries and technologies were employed to meet the system's requirements for cryptographic security, decentralization, concurrent processing, peer-to-peer communication, and data serialization. This section provides a detailed overview of each component utilized in the system, along with the rationale behind their selection.

**3.1.1 Libraries Used:**

**1. Hashlib:** Hashlib is a Python library that provides secure hashes and message digests. In this blockchain project, hashlib is utilized to compute cryptographic hashes during the mining process. Each block's hash is calculated to ensure the integrity and security of the blockchain. The hash acts as a digital fingerprint of the block, making it tamper-evident. Due to this blockchain acquire immutability & on larger use case it also provides digital signature/ digest of the data or transaction, It is also used while verifying the block.

Rationale: Hashlib offers a reliable and efficient means to generate hashes using algorithms such as SHA-256, Providing immutability, security and speed. This is crucial for maintaining the integrity of the blockchain.

**2. JSON:** JSON stands for JavaScript Object Notation is a data interchange format.  JSON is used for data serialization and decentralization so that data doesn't looses its format while transfer. It enables the encoding of transactions and blocks into a format that is easily transmitted over the network.

 Rationale: JSON provides human-readability, easier to parse, and widely supported across many languages, making it ideal for exchanging structured data between nodes in our project. It ensures inter-operability between different components of the system.

1. **Random:** The random library is used to generate random numbers within the system. We use this    generate random transaction tags which act like a digitally signed transaction that can be verified for the simplicity of the network. This   simulates a realistic environment where transactions are dynamically and unpredictably created.

Rationale: Generating random transactions is essential for testing the system's ability to handle varying loads and for comparing different consensus mechanisms under diverse conditions.

1. **Socket:** The socket library provides a low-level network interface for data transmission between computers. Sockets are employed to establish a peer-to-peer communication platform, allowing every computer in the network to connect and exchange data. This is fundamental to the decentralized nature of the blockchain, enabling direct communication between nodes providing base to implement decentralizeed network.

Rationale: Sockets offer a robust and flexible approach to implementing peer-to peer communication, which is critical for creating the network layer of the blockchain also it handles network calls at low level so that we can transfer data to specific data structures in the project at same time. They ensure that nodes can synchronize their states, share transactions, and broadcast newly mined blocks effectively.

**5. Multiprocessing:** The multiprocessing library in Python enables the creation of multiple processes that can run concurrently. This library facilitates the execution of various tasks by each node or computer in parallel. These tasks include receiving blocks and transactions from other nodes, mining new blocks, generating random transactions, and disseminating mined blocks to other nodes. Concurrent execution of these tasks enhances the system's efficiency and performance.

Rationale: Multiprocessing is essential for simulating a distributed network of nodes in a blockchain system. It allows for better utilization of available resources and significantly enhances overall throughput by enabling parallel task execution.

1. **Other mentions:**

**Time & date :** To handle time management & get time stamps for creation, mine duration etc.

**SQLite3:** Provide a light virtual database for the blockchain based on SQL database

**String:** To handle various string operation like creation, concatenation, slice etc.

**3.1.2 Data Structures:**

The core data structure used in the project are Blockchain, Mempool, Received Blocked handler all which of uses some arrangement of list data structure provided by python library & inherit simplicity, dynamic re-sizing, and fast access to elements.

**Blockchain**: The blockchain is represented as a 3D list of blocks, where each block is a 2-dimensional list. Each block contains a set of transactions, a timestamp, a nonce, and a hash of the previous block, current hash, time of creation which constitutes the block header too. This structure ensures that each block is linked to the preceding one, maintaining the integrity of the chain. Further this arrangement provide robust access to the blockchain and can perform operation on the block chain easily.

**Mempool**: The mempool is a list that holds unconfirmed transactions specific to the node on which the program is running. Transactions in the mempool are pending inclusion in a new block and are continuously updated as the node receives new transactions from the network. Further when some transaction are added to block and the block is verified then the that set of transaction are removed from the mempool. Every node n in N have there own mempool which are synchronized using socket library.

**Received Blocked handler:**  This is a temporary data structure used to store received blocks that are yet to verified by the system. If verified then they are appended to the personal ledger.

Further we use many local state pointer for each data structure for recording, manipulating the state of the data structure.

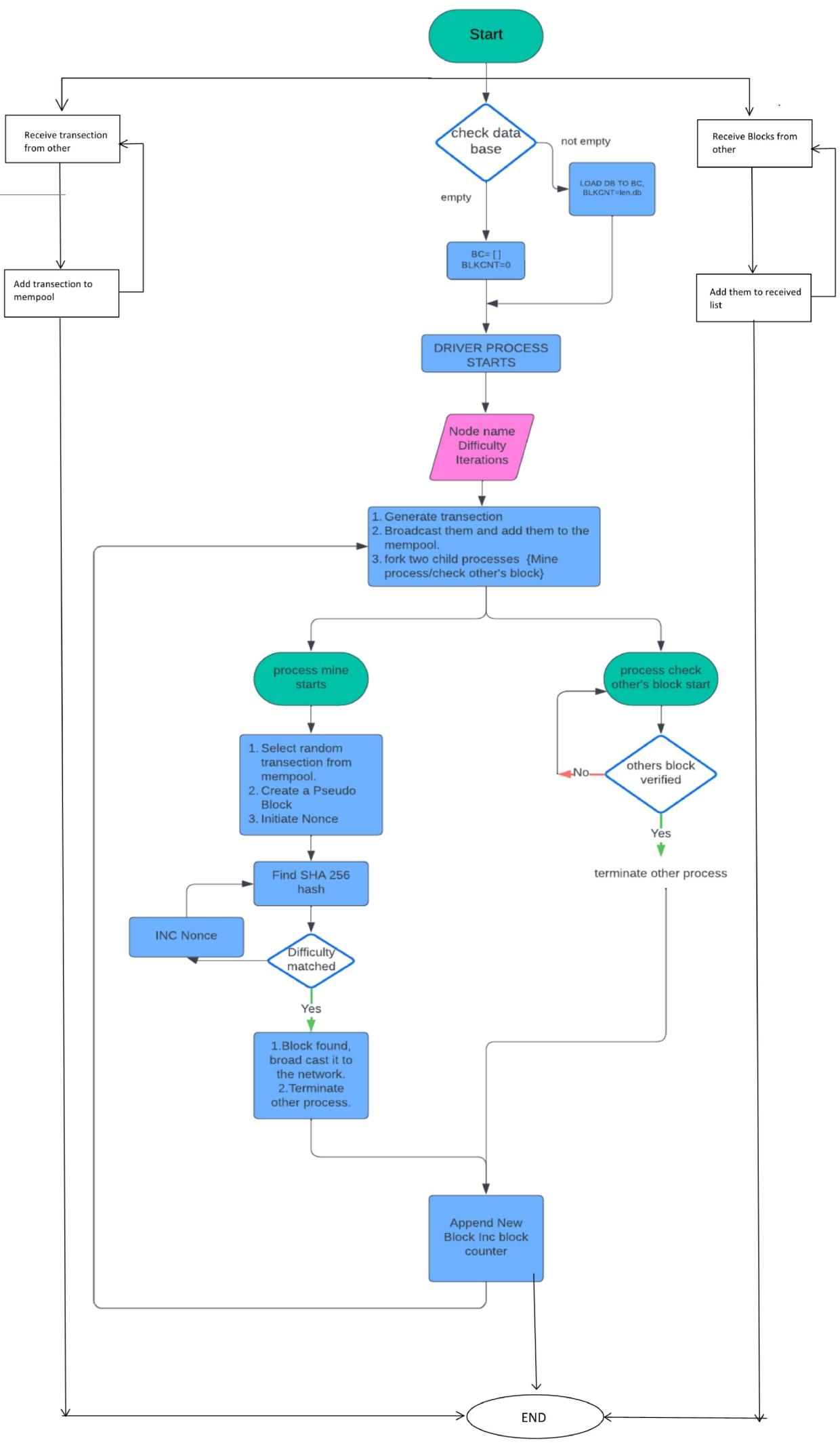
**3.1.3 Model’s Process architecture**

This is the heart of the model, it allows the model to make a decentralized network , replicate blockchain protocols and consensus algorithms . The system model highly depends on this process's framework and in this section we will have a brief overview of this . The model architecture constitutes of three main processes one for handling incoming transactions, one for handling incoming blocks and verify them and the last one work as a driver process which operates on the above-mentioned data structure in 2.1.2 all these processes also utilize python’s libraries and modules that are discussed in 2.1.1.

As discussed above the architecture constitute three process:

1. **Transaction receive server: -** Act as a server forlistening and receives transactions from other nodes which work as client for sending data, adding them to the local mempool of the node n.
2. **Block receive server: -**  Do the same work as the above server but it appends the received blocks from other miners in the received block list to further verify by the Node process.
3. **Node (Driver process): -** This process does the main part of running the consensus algorithm, generating transaction, broadcasting them to others LAO generating, verifying blocks. It does work in iteration as shown in the figure so that network model can work autonomously to collect data of different consensus mechanism.

* **Transaction Generation**: Randomly generates transaction IDs and appends them to the mempool.
* **Block Creation**: Forks a child process to select transactions from the mempool, mine a block by finding a valid nonce, and broadcast the new block to the network & in parallel one another process those checks whether some other node had mined the block or not, then verifies it, if verified the personal mining stops and the race starts for next block, in mean time if personal block is found it broadcast to the network and continues to next block.
* **Block Verification**: Another child process checks for incoming blocks, verifies them, and, if valid, stops ongoing mining operations.



**Figure 3.1.3a System Process Flowchart**

The above figure shows an overview of how different processes interact with each other. In context these both processes (a & b) provide parallel two way communication among nodes, as they are independent of driver process, they all uses an abstract version of the data structures that we mention above known as multiprocessing manager, which will help to distribute the data structure among this process and remove deadlocks, false change etc.After completing a set number of block iterations, the blockchain is saved to persistent storage, ensuring that the node's state can be restored in future sessions.

For storage at the end of the script the data is stored in a SQL database.

**3.2 Network Architecture:**

To implement the various consensus algorithms, we have considered a decentralized network of n number of nodes. These are represented by set N = {N1, N2, N3, .... Nn}. The systems are connected through a communication channel. The channel might be wired or wireless. Further, the network has been arranged with special kind of nodes mainly, full nodes which work as blockchain storage, verified and a miner node. Miner node is a node responsible for adding a new block in the blockchain. It gathers the transactions from a pool of transactions, verifies its correctness and  solves a mathematical puzzle. A node which completes solution become a candidate for adding a new block.  winner nodes gets a block reward that is added to the personal wallet of the node.

In our practical model The blockchain project utilizes a network of five computers connected configuring a full duplex network via a Local Area Network (LAN) using a switch. The computers used are of configuration processor Intel i9 13900k, This setup forms a peer-to-peer decentralized network where each node has the following capabilities:

### Node Communication:

The network model facilitates communication between nodes using socket-based networking. Each node runs a transaction server and a block receiver server to handle the exchange of transactions and blocks that we discussed in the system model, respectively. This setup enables decentralized operations where nodes operate independently while maintaining synchronization with the broader network.

### 3.2.1 Broadcast Protocol:

The system employs a broadcast protocol for the dissemination of transactions and blocks across the network:

1. ****Transaction Broadcasting**:** When a node generates a new transaction, it broadcasts the transaction ID to all connected nodes. Each node receiving this broadcast adds the transaction to its local mempool, preparing it for potential inclusion in a new block.
2. ****Block Broadcasting**:** Upon successfully mining a block, the nth node broadcasts the block to the network N. Other nodes receiving the block verify its validity. If verified, the block is added to their local blockchain, and any competing mining processes are terminated to maintain consensus.
3. **Mining**: Each node can participate in the mining process to create new blocks, this part is not compulsory for each consensus algorithm, as we had seen Pow uses this, but other consensus algorithm uses much simpler and efficient way to establish consensus among the network like PoS, DPoS.
4. **Transaction Management:** Nodes can send and receive transactions, ensuring the network remains active and transactions are processed continuously.
5. **Mempool:** Each node maintains its own mempool, where unconfirmed transactions are stored before being included in a block. Further the transaction that are used by the blocks are removed from the mempool by each node so that the transaction pool remains synchronized.
6. **Blockchain:** Every node maintains a copy of the blockchain as a 3d list where each 2d vector is a block of data and header constituents, ensuring redundancy and reliability of the data.
7. **Block Verification:** Nodes can verify blocks received from other nodes, ensuring that only valid blocks are added to the blockchain. To keep the model’s simplicity for our purpose, kept the verification part simple as when a node n receives a block it checks we the transaction as whether their respected signature is present the nodes personal mempool or not, then after which it verifies the block header and nonce of the data.

### 8. Synchronization and State Consistency: The network model ensures that all nodes in the system remain synchronized through continuous communication and validation of transactions and blocks. By frequently exchanging blockchain data, nodes maintain a consistent and up-to-date state, ensuring the integrity and stability of the blockchain network.

**Five Node Network setup using LAN**

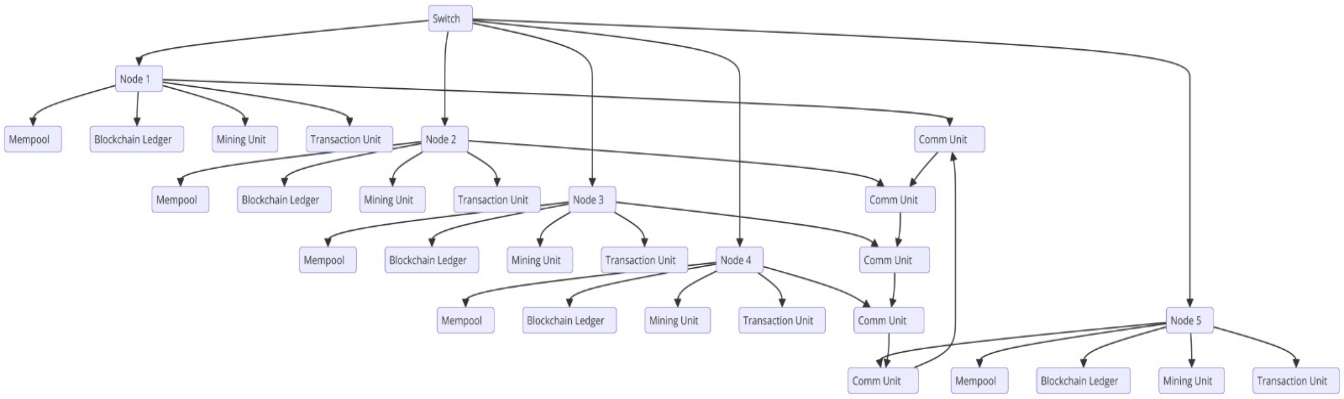


Fig. 3.2 a

**3.2.2 Design Considerations :**

The system model is crafted to facilitate the execution and comparison of different consensus mechanisms mainly efficient for Pow, PoS & DPoS. Consensus mechanisms are protocols that achieve agreement on the state of the blockchain among distributed nodes. By leveraging the aforementioned libraries and technologies, the system can simulate a network of nodes that concurrently process transactions, mine blocks, and communicate with each other. This design allows for an effective evaluation of various consensus mechanisms, highlighting their strengths and weaknesses in terms of performance, security, and scalability.

Further with some little tweaks this model can be used for many other consensus algorithms providing a good way & effective way to test new consensus algorithm.

**4 Data Analysis:**

In this part we will get the insights of our model, the consensus algorithms their efficiency, Model reliability with real world data in statistical and analytical approach.

**4.1 Pow:**

**4.1.1 Introduction to Analysis:**

Here we will analyse the system performance and see PoW’s Time & Difficulty analysis, check the co-relationality with bitcoin data, how we collected the data and the setup for the data. The data here used as discussed, in the System & Network Model is collected during our experiment on five PCs that are using i9-13900k processors. The frequency of system is around 5.4-5.9 GHz, so it performs on an average around 5 billion operation per second. The data used is combination of key matrices of the difficulties on which we ran our model to mine a block, Further data includes the average time to mine at a particular difficulty, time taken to mine on different initial conditions and the experiment is repeated at a minimum of 20-30 iteration for accuracy of the data so that the statistical analysis we perform may give accurate and reliable result.

**4.1.2 Data Collection & Data Setup**

As discussed previously data incudes the block data for multiple difficulties for mining a block under a P2P competitive network model, it include block data like the time to mine or generate a nonce for that difficulty, the node tag that mined the

Block, hashing power indulged by the network. Then this mining competition is iterated many times in between 25-30 for each difficulty. As we know that for each increase in a difficulty from Dn to Dn+1 the search space for the nonce doubles its size as probability to find reduces by 2.

For this scenario we had used difficulty of 25, 26, 27, 28, 29 for the network, Hash power is number of nonce checked for the block to meet a certain difficulty level per second. Hash power is proportional to CPU's frequency, each hash contributes forming a block, finding a nonce for it from our experiments the average hash power for single node comes out to be 193khz.

**4.1.2 Statistical analysis of difficulties:**

For statistical analysis we used tools like mean, median, variance, standard deviation & probability density function for different difficulties and there corresponding nonce.

The mean or average nonce represents the average number of hashes done by the network to crack a specific difficulty calculated using the formula where n is the total number of iterations for each difficulty & xi is each individual value of nonce that is mined by the network and comes out to be 4806655.9, 13990282.17, 27782563.93, 57239628.93, 115795306.9 for respective difficulties 25, 26, 27, 28 ,29 respectively. As we can see as the search space is doubled with each difficulty increase the average nonce also increased exponentially by a factor of two.

The variance / standard variance of the experimental data is a better metric tool as it shows the spread of how disperse the nonce values are in respect to each other, higher variance shows that the probability of finding the correct nonce decreases rapidly as the nonce value become more distributed throughout the search space. variance is calculated using the formula where μ is the mean of the dataset x is each individual data point & n is the number of data points.

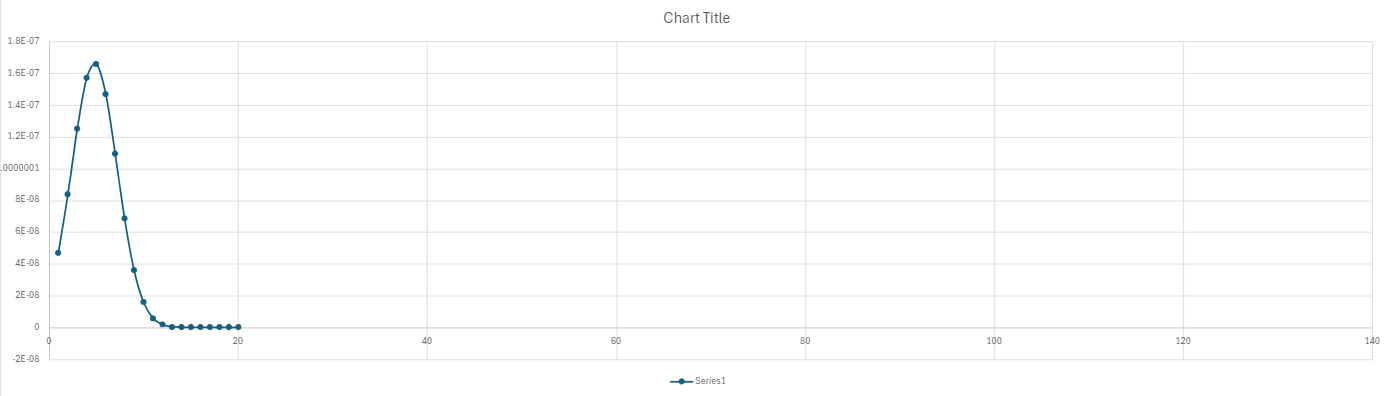
Variance of the data is 573.6988889, 8713.688889, 22328.49889, 99055.04556, 153865.7289 for the respective difficulty 25, 26, 27, 28 29. As we know in statistic Standard Deviation is considered more effective than the variance for seeing the spread is calculated using σ = , calculated std. deviation is 23.95201221, 93.347114184, 149.4272361, 314.7301154, 392.257229.

For better understanding of the effect of increasing the difficulty we will analyse the data by converting it into a Probability Density Function (PDF). We will use a normal distribution for this also known as gaussian curve or bell curve A normal distribution, also known as a Gaussian distribution, is a symmetric, bell-shaped distribution where most of the data points cluster around the mean. In a normal distribution:

* The mean, median, and mode is all same.
* The standard deviation controls the spread of the distribution in normal curve.
* Approximately 68% of the data falls within one standard deviation of the mean, 95% within two standard deviations, and 99.7% within three standard deviations (the 68-95-99.7 rule).

Representing the raw nonce data into bell curve with small sample size results in a jagged sine wave like curve so we had normalized it by firstly putting the data in bins of ranges e.g. For nonce 4089 it will fall in range bin 4000-5000, then we calculate the frequency of the bin ranges to get the likelihood of the nonce to fall in specific range bin. After this we normalized the data using the normal distribution over the frequency and got the following curves as the result.

A) For difficulty 25:

 Fig 4.1.2 a

B) For difficulty 26 :



Fig 4.1.2 b

C) For difficulty 27:

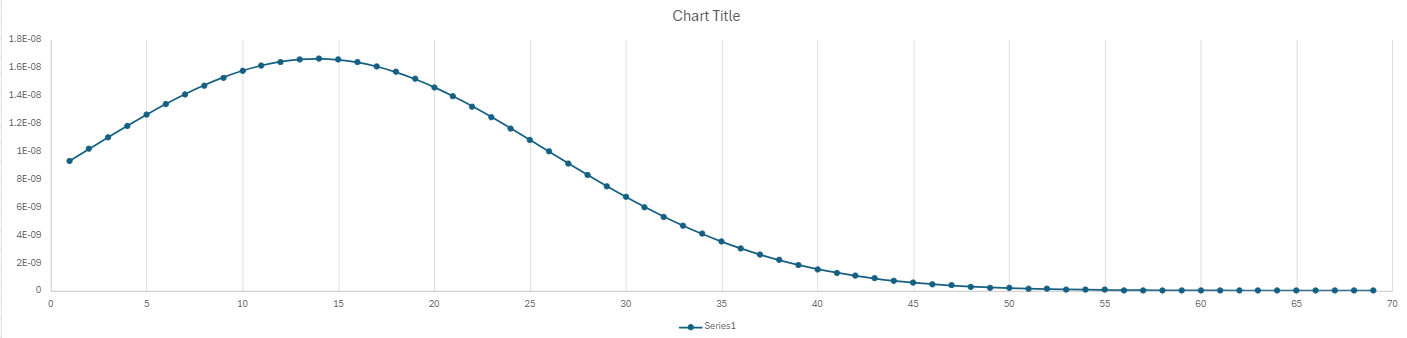
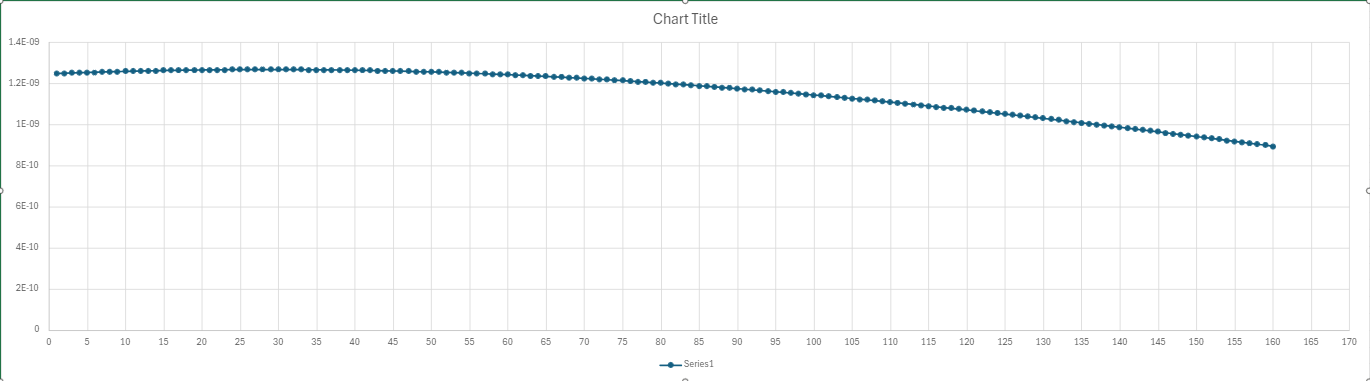


Fig 4.1.2 c

D) For difficulty 28:

 Fig 4.1.2 d

E) For difficulty 29:

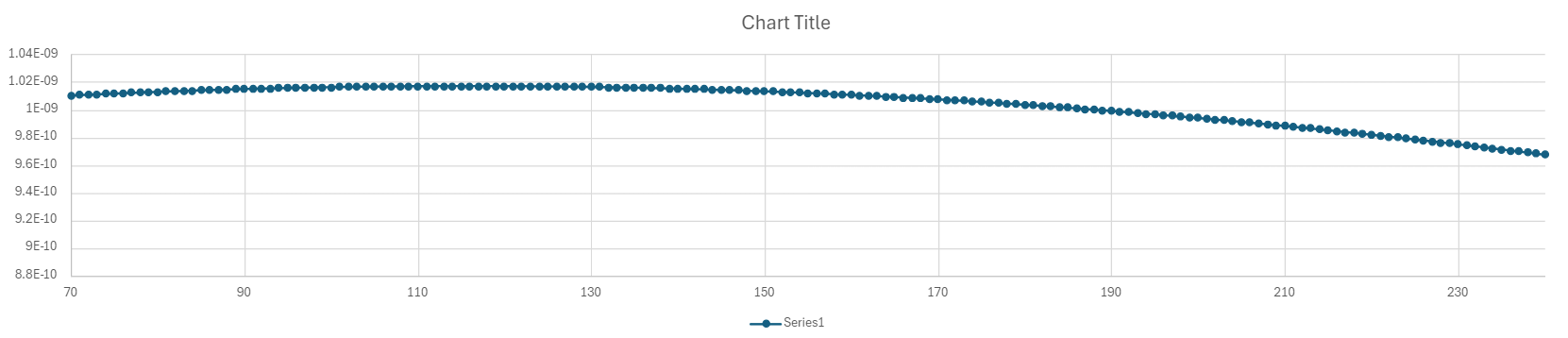


Fig 4.1.2 e

As we get the insight from the data for various difficulties, we see saw that as difficulty increased the normal curve became flatter and flatter to the x axis. This shows that the probability distribution of finding a nonce spreads uniformly over x axis or the search space, which indicates that the there is no other way then calculating the nonce one by one for higher difficulties also known as brute force approach making the Pow more reliable and secure than any other consensus algorithm.

Further these curves maximum height shows there respective mean and spread is proportional to the variance of the data to both side of the curve as normal curve is symmetric to its peak, But as we seen earlier it converges to a flatter curve when difficulty increases making the reward probability scattered evenly.

**4.2 Time Analysis of Pow:**

In this section we will compare our model with real world blockchain, mainly Bitcoin. The relational analysis between the time taken to mine in sec. , hash rate of computer, total no. Of nodes taking part in the network & the difficulty set. In case of Bitcoin the difficulty is changed every 2 weeks or after every 1024 blocks mined in such a way so that average time to mine a block remains constant I.e. 10 min or 600 sec.

As observed from the data time taken to mine t is directly proportional to 2 to the power of difficulty d,

t **∝** 2d

& is inversely proportional to the average hash rate h of the single node in the network

t **∝** 1/h

& also inversely proportional to the no. of participants in the network n.

t **∝** 1/n

So from this we get

t **∝** 2d/(h\*n)

h\*n is also referred as net hash power of network

As bitcoin net time is constant, we can alter the equation to have a relation between the network difficulty and the hash power of network (h\*n)

t\*(h\*n) **∝** 2d

Taking log at base two both side to move difficulty factor d from exponential to linear proportionality.

Log2 (t\*h\*n) **∝** d

As we know t for bit coin is 600 second, putting this in above equation

Log2 (600\*h\*n) **∝** d

So, this equation is our final relation for described the difficulty so that it took only about 10 min to mine a block in network with n participant with h as avg. Hash power h.

Checking this equation on our model we get accuracy of 97.76 percent, The results accuracy also depends on the assumption that each individual is competing for some different nonce in each iteration or number of nodes competing for same block of data is improbable or less likely to happen as this condition leads to wastage of hashing power so more hashing power is needed to get the same time within the given difficulty. For our model this is very less likely to happen as we have a small mempool and small number of nodes so chances of two nodes making a block with same transactions in same order on same time is very less. But in case of bitcoin as mempool is huge on an average consist of around 200k transaction and with millions of nodes this condition became feasible resulting in decrease in the accuracy of the given equation, but can be increased we know the contribution of individual mining authority or pools in the network, for this we have taken data from blockchain.com which provides real time data of cryptocurrencies like bitcoin & Ethereum. The data we used from that site is the bitcoin’s network hash rate and constituent difficulty.



Fig. 4.3 a (bitcoin network hash rate over past 1 year) **[15]**

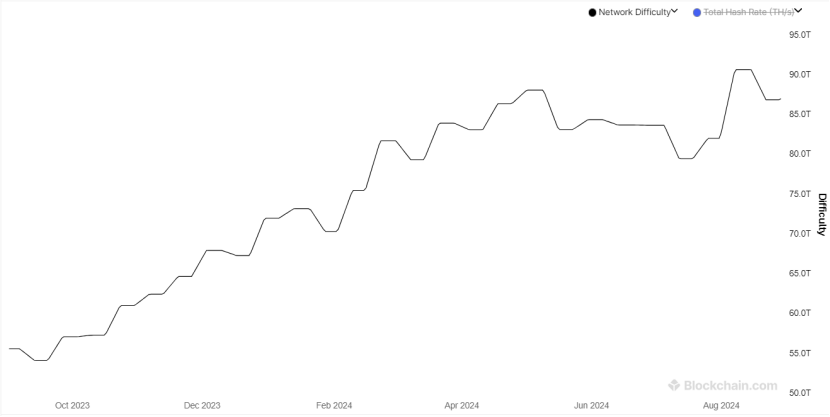
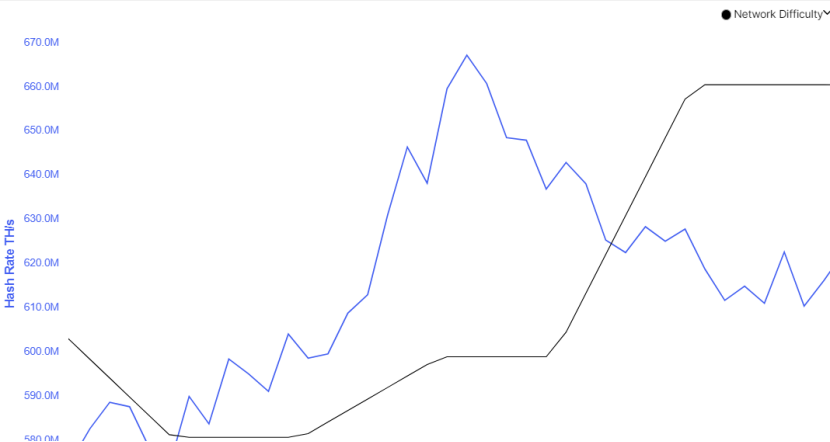


Fig. 4.3 b (bit coin network difficulty over past 1 year)**[15]**

From this data it also shows the linear relationship between network difficulty (considered after log) and hash rate as derived from our equation. Checking our equation on the real data taking in intervals as bitcoins difficulty changes within every

two weeks, in that time hash rate fluctuates many times (fig 4.3 c), so we took avg hash rate in that mean time for our purpose in intervals.

 Fig 4.3 c (hash rate & network difficulty) **[15]**

Putting data in equation and checking accuracy by real results accuracy comes out to be 67.565%, taking consideration the mining pools distribution, it increases to around 82.17 %. So, this equation shows the relation with some good accuracy, to set network difficulty for blockchains using Pow such that average time remains constant.

**4.3 Time/security analysis of** **Pow vs** **PoS, DPoS:**

As PoS & DPoS doesn't rely on computational power to secure the network, its time & energy consumption is way less then Pow. To replicate this in our network we randomly assign a node for verification of transactions & in case of DPoS the validator node works same as the PoS the only catch is validator are the nodes which also met the certain condition like they have a minimum holding of that particular asset mainly currency also known as stake. This gives a rise to a trade-off between time and energy used by a consensus algorithm and security / immutability of the data or the network that is using that consensus algorithm. This trade of can also be seen in our model as time increase complexity, energy and security of network also increases, in real world this leads to multiple derivation of blockchain implementation like Bitcoin, Ethereum, Doge, Lite coin etc.

On spectrum of security and time/energy Pow falls on the high end which is why it uses time and energy but also provide more reliable secure network on other hand PoS lies at left mid of spectrum which is why Ethereum which uses it is transaction fast & energy efficient, rest algorithms on the spectrum are shown below.

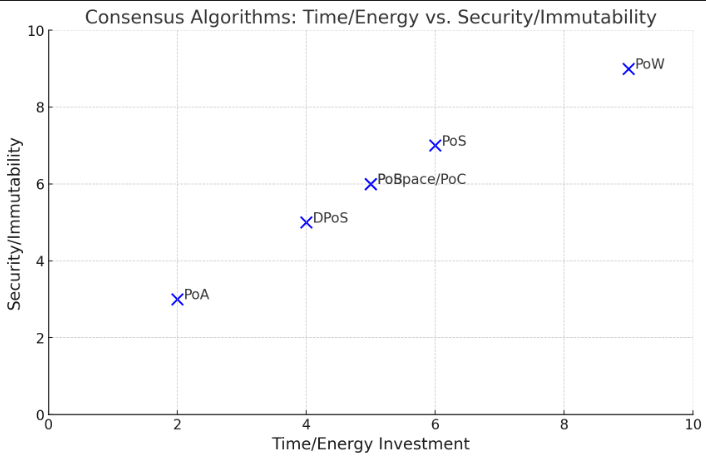


Fig 4.3 a Security vs Time/Energy Spectrum

**5. Conclusion:**

In this project the implementation and analysis of Blockchain and its consensus Algorithm are done, this project also underscores the depth analysis of PoW mining procedure. Further the experimental data shows as mining difficulty increases, the variation in nonce values are higher, which corresponds to the decline of successful block mining probability making the block chain secure and reliable. This effect aligns with the real-world findings from Bitcoin and other cryptocurrencies.

Further we also concluded the difficulty and mining time relation for PoW, this relation works well for our experiment and also shows good results over real world data. Also, algorithm like PoS and DPoS make it possible to achieve the same result as PoW but with lower time and energy consumption with a little bit of trade-off between time & security of the network since they employ different validation methods. These results shed light on the challenges that arise when developing blockchain systems and the trade-off between performance, security, and power usage. The findings of this study extend knowledge on the functioning of blockchain technology and its use in different forms of cryptocurrency & other implementation .

**References:**

1. Haber, S., Stornetta, W.S. How to time-stamp a digital document. *J. Cryptology* **3**, 99–111 (1991). https://doi.org/10.1007/BF00196791

[2] http://szabo.best.vwh.net/

[3] <http://www.konst.de/stefan/seclog.pdf>

[4]<https://www.researchgate.net/publication/228640975_Bitcoin_A_Peer-to-Peer_Electronic_Cash_System>

[5]https://youtu.be/bBC-nXj3Ng4?si=1JuuGs91oCYBHPeB

[6]Leslie Lamport, Robert Shostak, and Marshall Pease. 1982. The Byzantine Generals Problem. ACM Trans. Program. Lang. Syst. 4, 3 (July 1982), 382–401. <https://doi.org/10.1145/357172.357176>

[7]Buterin, V. (2014). A next-generation smart contract and decentralized application platform. *Ethereum White Paper*.

[8] Larimer, D. (2014). Delegated proof-of-stake (dpos). *Bitshare whitepaper*, *81*, 85.

[9] Eyal, I., & Sirer, E. G. (2018). Majority is not enough: Bitcoin mining is vulnerable. *Communications of the ACM*, *61*(7), 95-102.

[10]Dwork, C., & Naor, M. (1992, August). Pricing via processing or combatting junk mail. In *Annual international cryptology conference* (pp. 139-147). Berlin, Heidelberg: Springer Berlin Heidelberg.

[11]King, S., & Nadal, S. (2012). Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August*, *19*(1).

[12]Gencer, A. E., Basu, S., Eyal, I., Van Renesse, R., & Sirer, E. G. (2018). Decentralization in bitcoin and ethereum networks. In *Financial Cryptography and Data Security: 22nd International Conference, FC 2018, Nieuwpoort, Curaçao, February 26–March 2, 2018, Revised Selected Papers 22* (pp. 439-457). Springer Berlin Heidelberg.

[13]Decker, C., & Wattenhofer, R. (2013, September). Information propagation in the bitcoin network. In *IEEE P2P 2013 Proceedings* (pp. 1-10). IEEE.

[14]Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J. A., & Felten, E. W. (2015, May). Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *2015 IEEE symposium on security and privacy* (pp. 104-121). IEEE.

[15] <https://www.blockchain.com/explorer/charts/hash-rate>

|  |
| --- |
|  |