

# Ledger on internet of things: a blockchain framework for resource-constrained devices

Suresh Jaganathan<sup>1</sup>, Karthika Veeramani<sup>2</sup>

<sup>1</sup>Department of Computer Science and Engineering, Sri Sivasubramaniya Nadar College of Engineering, Chennai, India

<sup>2</sup>School of Computer Science and Engineering, Vellore Institute of Technology, Chennai, India

## Article Info

### Article history:

Received Apr 11, 2024

Revised Feb 1, 2025

Accepted Mar 15, 2025

### Keywords:

Blockchain

Consensus

Internet of things

Leader election

Skip list

## ABSTRACT

The increasing use of resource-constrained devices such as the internet of things (IoT) in various applications has led to the need for an optimized blockchain framework for these devices. Blockchain-based IoT networks allow businesses to access and share IoT data within their organization without centralized authority. However, existing frameworks are not designed for IoT applications and lack features like decentralization, scalability, and network overhead. To overcome these limitations, a new blockchain framework is proposed: ledger on internet of things (LIoT), which has a new consensus-based leader election algorithm to address the challenges of existing algorithms with high block creation time and communication overhead. Moreover, a novel data structure has been developed to reduce the storage size of the ledger effectively. The proposed framework also employs a docker for deployment, which provides an efficient and easy setup of blockchain nodes without requiring the individual configuration of each machine, increases the efficiency of the consensus process, and enables convenient deployment and management of the blockchain framework on resource-constrained devices. Furthermore, the performance of the proposed consensus method is analyzed using various performance parameters, including CPU usage, memory usage, transaction execution time, and block generation time.

This is an open access article under the [CC BY-SA](#) license.



## Corresponding Author:

Karthika Veeramani

School of Computer Science and Engineering, Vellore Institute of Technology

Chennai, Tamilnadu, India

Email: karthika.v@vit.ac.in

## 1. INTRODUCTION

Blockchain technology [1], [2] is built upon the foundation of distributed ledger technology (DLT), which serves as a decentralized database for sharing transaction information. This system chronologically adds operations to the DLT and stores them as blocks within the ledger. Each block in a blockchain refers to the block that precedes it, creating an interconnected chain. In this context, transactions are stored in blocks within the ledger. To ensure data integrity, these blocks are replicated to prevent loss during transmission errors. Additionally, the data within each block is encrypted using private keys, making it inaccessible to unauthorized nodes. These replicated and encrypted blocks are then distributed among decentralized nodes across the globe.

Blockchain technology has the potential to create immense value across various industries. According to a report by the World Economic Forum, it is estimated to generate 1.5 trillion in new value by 2030. With its ability to provide greater transparency and security in financial transactions, improve supply chain management, and increase efficiency in numerous industries [3]–[6], blockchain technology is on the

rise. However, as blockchain technology continues to evolve, so do the challenges and limitations of existing algorithms used to ensure the security and reliability of the blockchain network [7]. Also, the blockchain's data structure exhibits characteristics similar to a linked list, resulting in linear growth and increased space complexity. Furthermore, existing consensus algorithms and leader election processes often impose higher time and communication overhead [8], [9].

To address these challenges, this article proposes a new consensus-based leader election algorithm that guarantees consensus attainment while simultaneously reducing the number of messages exchanged. Moreover, a novel data structure designed explicitly for ledger storage to enhance overall efficiency has also been designed. In terms of deployment, the entire framework is embedded within docker containers. Docker's containerization technology [10] streamlines the setup process, eliminating the need for individual machine configurations. By leveraging docker, the deployment of the blockchain framework becomes more efficient and convenient, facilitating rapid setup without compromising configuration requirements.

By improving the efficiency and reliability of the blockchain network, this proposed framework has the potential to contribute significantly to the widespread adoption and success of blockchain technology. A more secure, efficient, and reliable blockchain network could positively impact various industries, create new opportunities for businesses and individuals, and drive innovation [11], [12]. Moreover, the proposed novel data structure is designed to optimize the search functionality, enabling faster and more efficient querying of transactions. By departing from the conventional linked list data structure, the proposed solution offers a more streamlined and effective data storage and retrieval approach, ensuring improved performance and responsiveness. A peer-to-peer network is constructed to facilitate seamless data sharing and communication, enabling nodes across different branches to exchange the stored data. As each node possesses a copy of the ledger, data backups are consistently maintained, ensuring data availability and reliability even when specific nodes experience downtime within the peer-to-peer network.

The primary objective of this work is to design a new blockchain framework incorporating a novel consensus method that significantly reduces communication overhead. Introducing a new consensus mechanism tailored to the application's specific requirements can improve transaction execution efficiency and overall system performance. The rest of the paper is organized as follows: section 2 focusses on recent works on blockchain architectures and internet of things (IoT). The proposed work Ledger on internet of things (LIoT) is detailed in section 3. The experimental results and analyses can be found in section 4. Finally, section 5 concludes the paper with possible future work.

## 2. RELATED WORK

This section overviews existing blockchain architectures, functionalities, applications, and limitations. It also summarizes the key findings from these systems. According to Bandara *et al.* [13], a new blockchain architecture called Tikiri is proposed. Tikiri is designed to be scalable, lightweight and optimized for the unique demands of IoT devices. Unlike traditional monolithic blockchains, Tikiri is implemented as a collection of distributed microservices. The architecture consists of four primary services: gateway service, Lokka service, storage service, and Kafka message broker. These services work together to ensure efficient transaction processing and high security. Tikiri offers high scalability, concurrent transaction execution, and fast search capabilities using a distributed database.

Bandara *et al.* [14] address the challenges of integrating blockchain with big data. They propose a blockchain storage system, Mystiko, based on the Apache Cassandra distributed database. Mystiko provides high transaction throughput, big data storage, and scalability, overcoming the limitations of traditional blockchains. The system consists of storage, chain, and miner services, which collectively handle transactions, verification, and block creation.

Bandara *et al.* [15] present Rahasak, a new blockchain system that tackles storage, execution order, and smart contract structure issues. Rahasak utilizes a "validate, execute, group" blockchain architecture with Apache Kafka-based consensus for real-time transaction execution. It employs functional programming and actor-based smart contracts for concurrent transactions. Data replication is achieved through sharding and a microservice-based architecture, which enhances scalability. Rahasak also incorporates full-text search capabilities and handles high transaction throughput using Apache Kafka message broker and reactive stream methodology.

In another research [16]–[19] a lightweight blockchain system called light chain is introduced for integration with the industrial internet of things (IIoT). Light chain focuses on resource efficiency and power-constrained IIoT devices. It employs a green consensus mechanism called synergistic multiple proof (SMP), and a lightweight data structure called light block to reduce computational and storage requirements. Light chain offers improved performance, reduced storage needs, and secure communication for IIoT applications. Zamani *et al.* [20] propose rapid chain, a sharding-based blockchain protocol to address slow transaction processing and limited scalability. Rapid chain divides the network into smaller committees, allowing

parallel processing of transactions and improving throughput. Its sharding-based consensus algorithm can handle Byzantine faults and achieves high transaction throughput.

Another studies [21]–[23] propose a decentralized blockchain-based architecture for mobile IoT systems. The framework attempts to ensure security, transparency, and efficiency while addressing the drawbacks of centralized architectures. It tackles data collection, analytics, and storage challenges in resource-constrained IoT devices. The sensor-chain framework offers a lightweight solution with reduced resource consumption while retaining vital information about IoT systems. It enables secure and scalable communication, unlocking the potential of blockchain technology for mobile IoT.

The studies and experiments conducted on blockchain architectures such as Rahasak utilizing the Kafka consensus algorithm and Mystiko based on the Apache Cassandra distributed database have been observed to incur significant communication overhead. This can impact the overall performance and efficiency of the system, especially when handling a large number of transactions. Additionally, the linked list data structure used for data storage may not be optimal for efficient querying of transactions, potentially leading to slower search operations.

### 3. METHOD

#### 3.1. Ledger of internet of things

LIoT is a blockchain framework comprising three essential layers: the network, blockchain, and application layers. The network layer establishes connectivity among multiple nodes through a peer-to-peer network architecture. Each node utilizes docker, a versatile platform for deploying, shipping, and running applications within isolated containers. These containers provide lightweight and portable environments that facilitate the efficient deployment of applications across the network. In the blockchain layer, LIoT implements a unique leader election algorithm. This algorithm selects a leader among the connected nodes, creates a new block, and propagates it to other nodes. The blockchain layer also incorporates a novel data structure to securely organize and store data, ensuring its integrity and tamper-proof nature. Figure 1 depicts the architecture of LIoT.

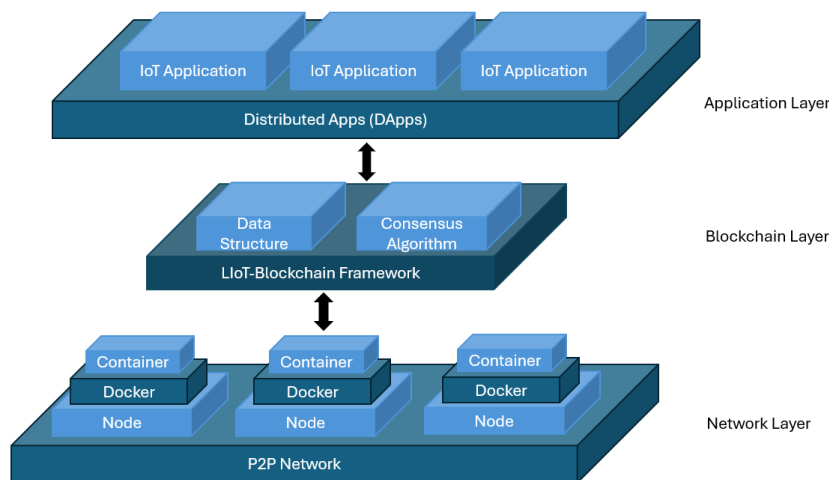


Figure 1. Architecture of LIoT

In the application layer of LIoT, a specific use case is implemented: the attendance ledger. The attendance ledger leverages the blockchain infrastructure provided by the lower layers to create a secure and decentralized system for tracking attendance records. With LIoT, the attendance ledger application benefits from the inherent features of blockchain technology, such as immutability, transparency, and decentralization. Each attendance record can be securely stored in a block within the blockchain, ensuring that the data remains tamper-proof and resistant to unauthorized modifications.

Embedding LIoT in docker enhances the framework's portability. Docker allows LIoT to be encapsulated within a container, including all necessary dependencies and configurations. This containerization ensures that all the components and configurations required for running LIoT are bundled together, making it easy to deploy on different platforms and environments. With docker, developers can

create a docker image once and run it consistently across various systems, eliminating the need for manual setup and configuration. This streamlined deployment process improves portability and accelerates the deployment of LIoT applications. By leveraging docker's capabilities [24], including overlay networks, users can efficiently manage containers and enjoy several benefits. One significant advantage of docker is the seamless communication between containers, which significantly enhances the system's scalability. Containers can be easily scaled up or down to accommodate changing demands, enabling LIoT applications to handle increased workloads without disruptions. Another critical benefit is docker's built-in security mechanisms, ensuring container communication is secure and isolated. Containers are encapsulated within their environment, minimizing the risk of unauthorized access or interference. This heightened security is crucial for protecting sensitive data and maintaining the system's integrity.

### 3.2. Design of consensus-based leader election algorithm for ledger of internet of things

The design of a consensus-based leader election algorithm for LIoT addresses the need for a reliable and decentralized process to select a leader node in a LIoT network. The leader node is crucial in various distributed applications, including the attendance ledger system discussed later. In a LIoT network, where resources and computing power are limited, it is essential to devise an efficient and lightweight algorithm for leader election. The consensus-based approach ensures that the selection process is fair, transparent, and resistant to manipulation or malicious attacks [25], [26]. The leader election process begins by generating 10 random pandigital numbers with unique digits. These numbers are then evaluated against six specific conditions to determine if they meet certain requirements. The conditions are as follows:

- Condition 1: check if the 2<sup>nd</sup>, 3<sup>rd</sup>, and 4<sup>th</sup> digits together is divisible by 2.
- Condition 2: check if the 3<sup>rd</sup>, 4<sup>th</sup>, and 5<sup>th</sup> digits together is divisible by 3.
- Condition 3: check if the 4<sup>th</sup>, 5<sup>th</sup>, and 6<sup>th</sup> digits together is divisible by 5.
- Condition 4: check if the 5<sup>th</sup>, 6<sup>th</sup>, and 7<sup>th</sup> digits together is divisible by 7.
- Condition 5: check if the 6<sup>th</sup>, 7<sup>th</sup>, and 8<sup>th</sup> digits together is divisible by 11.
- Condition 6: check if the 7<sup>th</sup>, 8<sup>th</sup>, and 9<sup>th</sup> digits together is divisible by 13.

Each condition represents a divisibility test that the generated numbers must satisfy. The algorithm iterates through the generated numbers and counts the number of conditions each number meets. The number that satisfies the highest conditions with the minimum number of iterations is elected as the leader. Once elected, the leader node takes responsibility for creating a new block in the blockchain. This consensus-based leader election algorithm ensures that the leader is selected relatively and that the subsequent block-creation process proceeds decentralized. Algorithm 1 details the steps involved in the leader election.

#### Algorithm 1. Leader election

```
Function leader election
  Pan_digital ← Empty list;
  Max_conditions_satisfied ← 0;
  primes ← [2, 3, 5, 7, 11, 13];
  min_iterations ← ∞;
  pan_number ← None;
  for i in range(1, 11) do
    random_number ← generate_pandigital();
    conditions_satisfied ← check_conditions(random_number, primes);
    num_conditions_satisfied ← length(conditions_satisfied);
    if num_conditions_satisfied > max_conditions_satisfied then
      max_conditions_satisfied ← num_conditions_satisfied;
      pan_number ← random number;
      min_iterations ← i;
    end
  end
  if max_conditions_satisfied > 0 then
    output ← "PAN DIGITAL NUMBER" + pan_number + " - Maximum Conditions Satisfied: " +
      max_conditions_satisfied + " - Min Iterations: " + min_iterations;
    pan_digital.append([self.id, pan_number, max_conditions_satisfied, min_iterations]);
  else
    print("None of the generated numbers satisfied any condition.");
  end
return pan_digital
```

### 3.3. Skip list: a data structure for ledger of internet of things

The need for a more efficient data structure arises due to the scale and complexity of managing data. As the number of data grows, traditional data structures like linked lists struggle to maintain optimal performance. The introduction of a novel data structure known as "skip list" efficiently stores and retrieves data in the blockchain [27], [28]. A skip list is initially a sorted linked list, which cannot be binary searched due to its nature. However, additional layers are added at the top of the bottom list. Each new layer includes

elements from the previous layer with a certain probability, often  $1/2$ . These new layers maintain the ordering. Some elements are kept while others are discarded probabilistically, resulting in a skip list with multiple ordered layers. The skip list has three significant properties: height (h), which is the number of linked lists in it; the number of distinct elements (n); and probability (p), which is normally  $1/2$ . The highest element appears in  $\log_{1/p}(n)$  lists on average. If  $p = 1/2$ , there are  $\log_2(n)$  lists on average. Every element in the skip list has four pointers: left, right, top, and bottom. These pointers enable efficient search operations in the skip list.

The time complexity of operations in a skip list is typically  $O(\log n)$ , where n is the number of elements in the skip list. This complexity holds for insertion and search operations. These are expected or average-case bounds because skip lists use randomisation in their data structure. The worst-case bounds for the skip list are  $O(n)$ , but these are less commonly analysed due to the probabilistic nature of the skip list.

The space complexity of a skip list is  $O(n)$ , where n is the number of elements in the skip list. This means that the space the skip list utilises grows linearly with the number of elements. To reason about the space complexity, let us consider the number of positions in the skip list. It can be represented by  $n \sum_{i=0}^h \frac{1}{2^i}$  which is equal to  $n \left(1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \dots\right) = n \times 2$ . Therefore, expected space utilisation is simply  $O(n)$ . Algorithm 2 details the working of the insertion process in the skip list:

#### Algorithm 2. Skip list insertion

```
Function Insert(key):
  p ← Search(key);
  q ← null;
  i ← 0;
  h ← 0;
  n ← 1;
  repeat
    i ← i + 1 ; // height of tower for new element
    if i ≥ h then;
      h ← h + 1;
      createNewLevel(); // creates a new linked list level
    end
    while p.above = null do
      p ← p.prev ; //scan backwards
    end
    p ← p.above;
    q ← insertAfter(key, p) ; // insert key after position p
    n ← n + 1;
  until (n == 10);
  return q
```

Algorithm 3 details the steps for the searching process in a skip list, and this function inputs a search key (key). This function returns a position (p), where the value at this position is the largest that is less than or equal to the key. The function performs a downward scan in the skip list and then scans forward to find the appropriate position.

#### Algorithm 3. Skip list search

```
Function Search(key):
  p ← top-left node;
  while p.below ≠ null do
    p ← p.below
    while key ≥ p.next do
      p ← p.next
    end
  end
  return p
```

The skip list is initialized with parameters such as the maximum level and the probability for determining the level of each node. These parameters are set during the initialization process and govern the structure and behavior of the skip list. The skip list starts with a header node, which serves as the starting point for traversal and provides a reference to the first node at each level. Each node in the skip list contains two main components: the data itself and a list of forward pointers. The forward pointers enable efficient navigation through the skip list, allowing faster search and insertion operations. During the insertion process, the level of each node is determined randomly based on the given probability parameter. This probabilistic

approach allows for a balanced distribution of nodes across different levels, promoting efficient search and traversal. Higher-level nodes act as shortcuts, allowing faster movement through the skip list.

#### 4. EXPERIMENTS AND RESULT ANALYSIS

A mesh network is a decentralized architecture where multiple devices establish direct peer-to-peer connections. Implementing a mesh network allows two or more devices to connect and exchange messages directly, eliminating the reliance on a centralized server. The devices can establish direct communication channels by assuming dual roles as clients and servers. To set up a mesh network, the code initializes the network by specifying the IP addresses of the devices involved. These IP addresses are typically stored in a configuration file. Each device is represented by a node object, created with its respective IP address and a designated port number. The network is then initiated by starting each node, allowing time for the connections to be established. The devices within the mesh network establish connections by connecting to the IP addresses listed in the network configuration. Delays are introduced between connection attempts to ensure proper connection establishment as shown in Figure 2.

The implementation of docker container communication involves utilizing docker's overlay network feature. Overlay networks in docker enable seamless communication between containers running on different hosts, providing a virtual network that allows secure interaction regardless of the container's physical locations. Four steps are needed to establish docker container communication: i) create an overlay network, ii) deploy docker containers, iii) connect containers to the overlay network, and iv) utilize container DNS resolution.

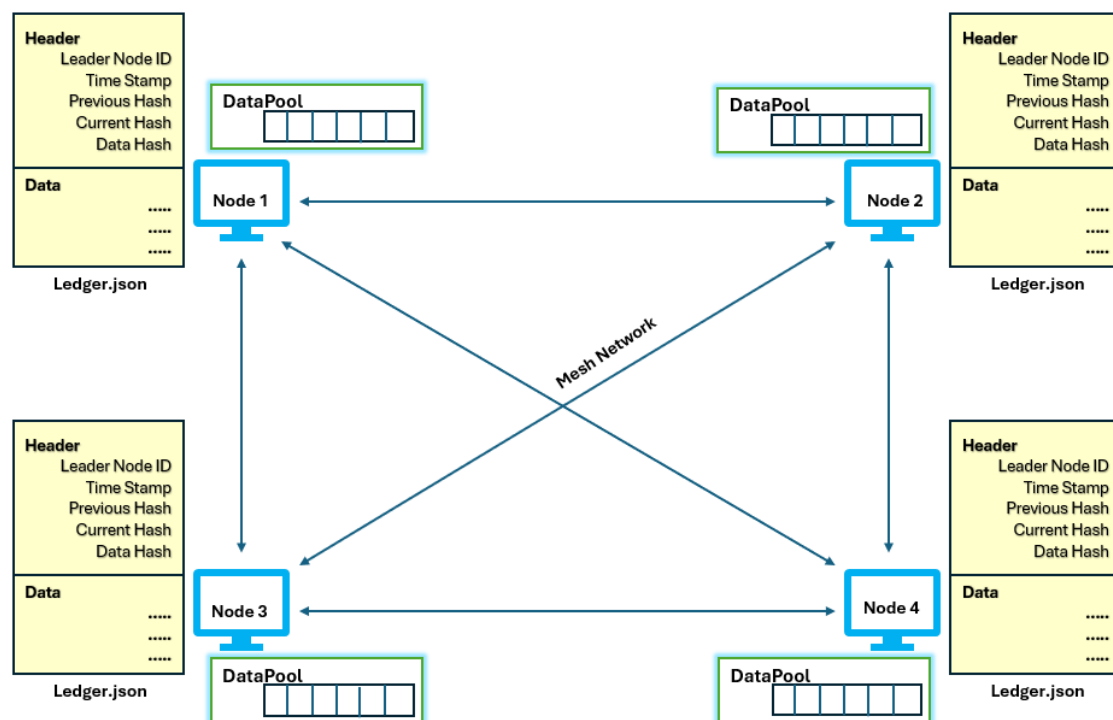


Figure 2. Implementation of IoT

##### 4.1. Embedding ledger of internet of things in docker

A vital step in embedding a LIoT in docker is creating a text file, "dockerfile", which contains instructions for building the docker image. Within the dockerfile, the base image is defined, necessary dependencies are installed, and the LIoT is configured to ensure its proper functioning within the docker environment. Once the dockerfile is ready, the next step involves building the docker image using the docker build command `sudo docker build -t myimage`. During this process, a tag is specified to identify the image uniquely. Docker compiles the container based on the instructions provided in the dockerfile, resulting in a complete docker image of the LIoT. With the docker image successfully built, the LIoT can run by creating containers from the image using the docker run command. By executing this command, instances of the LIoT

are started within separate containers. This allows for isolated execution and management of the LloT, ensuring its availability and functionality.

#### 4.2. Leader election algorithm for ledger of internet of things

The algorithm employs the concept of pandigital numbers, which are numbers that contain all digits from 0 to 9 without repetition. By generating 10 random pandigital numbers, each node participating in the election has an equal chance of becoming the leader. The conditions imposed on these pandigital numbers ensure the elected leader can satisfy specific divisibility tests. These tests help maintain the integrity and security of the leader election process. By meeting more conditions, a number demonstrates its capability to serve as a reliable leader. Once elected, the leader node takes responsibility for creating a new block in the blockchain. This block contains both a header and a body. The block's header includes the ID and the timestamp indicating the exact time the block was created, as well as the current, previous, and data hash. This information is vital for tracking and verifying the block's authenticity within the blockchain. The body section of the block contains a collection of data entries or transactions being added to the blockchain. These data entries are represented by a list of image hashes obtained by iterating over the files in the specified folder path and calculating the SHA256 hash for each image file.

After the block is created, it is appended to the genesis block, which serves as the initial block in the blockchain. The genesis block contains the fundamental information that establishes the foundation of the blockchain. Then, the leader node propagates it to other connected nodes in the network, as shown in Figure 3. These nodes play a crucial role in validating the received block. They verify the block's integrity by calculating its hash and comparing it with the provided current hash. If the calculated hash matches the current hash, it indicates that the block has not been tampered with during transmission. Upon successful validation, the nodes append the received block to their local copy of the blockchain.

```

pglab1@pglab1pc39-OptiPlex-3010:~/Downloads/phase25 python3 totnode_dylp_gb.py
Initialization of the Node on port: 10001 on node (10.6.2.39)
/bin/sh: 1: curl: not found
Genesis Block Created for node 10.6.2.39
Node connection overview:
- Total nodes connected with us: 0
- Total nodes connected to : 1
GENERATED : PAN DIGITAL NUMBER 5310846279 - Maximum Conditions Satisfied: 3 - Min Iterations: 4
[[['10.6.2.39', '5310846279', 3, 4]]]
RECEIVED FROM 10.6.2.140: PAN DIGITAL NUMBER 4310258697 - Maximum Conditions Satisfied: 3 - Min Iterations: 6
[[['10.6.2.39', '5310846279', 3, 4], ['10.6.2.140', '4310258697', 3, 6]]]
Leader Node ID: 10.6.2.39
Created Block: BLK{
  "block": {
    "header": {
      "LeaderId": "10.6.2.39",
      "timestamp": "2023-06-20 09:10:53",
      "prevHash": "832b511b804c339ff80f7790d786cd4a1f9e44a582cbd30b41805ea0e4051523ec810a6a0be3e833aef1929e454565a9660f11bc1d8c55bfc09b26db1b",
      "currentHash": "2b667e653e2821b6749aa745eb1ebfbbf444bc5243ee92f037d3e336b7d85ca",
      "dataHash": "c5ee7b3bba436fa5b50b4fd49771843cf232b3648532fd8eeff2207c60dd679"
    },
    "body": {
      "data": [
        {
          "da510f565d99da1c48cdab75e933f34a32d24f0bf71dbf0d03e9ebcd89cf",
          "230affb87eb07c3c85a59286866099464ef9716fbc99faef4269b9bdf9d9f957",
          "f0d0d7da03b1e14c2ae231c31f583541abf88d8addded1e7577b9ac1eca224987"
        ]
      ]
    }
  }
}

pglab1@pglab1-OptiPlex-3010:~/Downloads/phase25 python3 totnode_dylp_gb.py
Initialization of the Node on port: 10001 on node (10.6.2.140)
Genesis Block Created for node 10.6.2.140
connect_with_node: Already connected with this node.
Node connection overview:
- Total nodes connected with us: 1
- Total nodes connected to : 0
RECEIVED FROM 10.6.2.39: PAN DIGITAL NUMBER 5310846279 - Maximum Conditions Satisfied: 3 - Min Iterations: 4
[[['10.6.2.39', '5310846279', 3, 4]]]
GENERATED : PAN DIGITAL NUMBER 4310258697 - Maximum Conditions Satisfied: 3 - Min Iterations: 6
[[['10.6.2.140', '4310258697', 3, 6]]]
RECEIVED FROM 10.6.2.39: I'm the LEADER
RECEIVED FROM 10.6.2.39: BLK{
  "block": {
    "header": {
      "LeaderId": "10.6.2.39",
      "timestamp": "2023-06-20 09:10:53",
      "prevHash": "832b511b804c339ff80f7790d786cd4a1f9e44a582cbd30b41805ea0e4051523ec810a6a0be3e833aef1929e454565a9660f11bc1d8c55bfc09b26db1b",
      "currentHash": "2b667e653e2821b6749aa745eb1ebfbbf444bc5243ee92f037d3e336b7d85ca",
      "dataHash": "c5ee7b3bba436fa5b50b4fd49771843cf232b3648532fd8eeff2207c60dd679"
    },
    "body": {
      "data": [
        {
          "da510f565d99da1c48cdab75e933f34a32d24f0bf71dbf0d03e9ebcd89cf",
          "230affb87eb07c3c85a59286866099464ef9716fbc99faef4269b9bdf9d9f957",
          "f0d0d7da03b1e14c2ae231c31f583541abf88d8addded1e7577b9ac1eca224987"
        ]
      ]
    }
  }
}
Block Added

```

Figure 3. Block creation and forwarding

#### 4.3. Data structure for ledger of internet of things

The LloT utilised "skip list" as its underlying data structure. The skip list is a probabilistic data structure that provides an alternative to a traditional linked list with improved search and insertion times. It allows for fast and efficient traversal of elements, making it suitable for managing large amounts of data. The skip list data structure consists of multiple levels, each representing a subset of the elements in the list. Each level contains a linked list of nodes, where each node stores a value and a set of forward pointers. The forward pointers allow quick navigation through the levels, effectively skipping over elements and reducing the number of comparisons required during search operations. The "ledger.json" file data is inserted into the skip list, which organizes and indexes the blockchain data for easy retrieval and manipulation. Additionally, the primary function performs search operations on the skip list, allowing for efficient querying based on given criteria or conditions.

LloT optimizes the blockchain's storage, retrieval, and search operations by utilizing the skip list data structure. The skip list's efficient traversal and balanced structure make it well-suited for managing large amounts of data while maintaining fast access times and preserving the integrity of the blockchain. Compared



to a traditional linked list, skip list offers several advantages. First, skip list provides a logarithmic search time complexity, meaning that searching for an element increases slowly as the number of elements grows. In contrast, a linked list has a linear search time complexity, resulting in slower search operations for larger data sets. Secondly, when inserting a new element, the skip list dynamically adjusts its structure to maintain the desired balance, ensuring optimal search time. In linked lists, insertions require modifications to the neighboring nodes, potentially leading to slower performance for frequent updates.

Additionally, skip list is self-balancing, meaning it maintains a relatively even distribution of elements across the levels. This self-balancing property ensures that the performance of the skip list remains consistent over time, regardless of the order in which elements are inserted. The skip list consistently outperforms the linked list in terms of time taken for various numbers of records. This performance improvement is particularly significant as the number of records increases. The skip list's logarithmic search and insertion complexity allow it to scale more efficiently, making it a suitable choice for managing the attendance ledger in LIoT.

#### 4.4. Results analysis and discussion

##### 4.4.1. CPU usage

CPU usage statistics are obtained using the command `docker stats`, which provides real-time resource usage statistics. Figure 4 shows the recorded CPU consumption for LIoT and Tikiri as a share of the overall CPU. These measurements were taken during the testing phase of the lightweight implementation of LIoT. The primary reason for the lower CPU usage in LIoT can be attributed to its light blockchain architecture, which incorporates the pan digital leader election consensus mechanism.

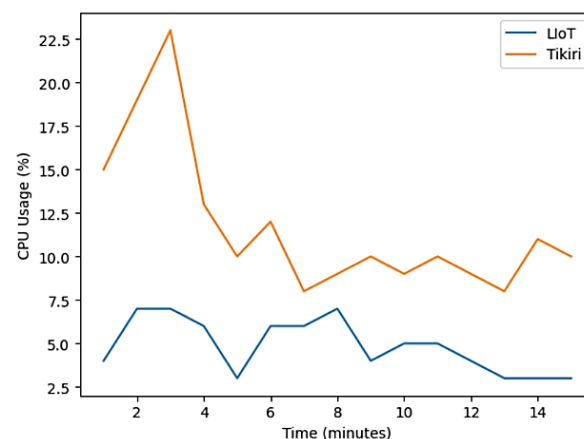


Figure 4. CPU usage

Tikiri has higher CPU usage than LIoT because of the communication between the Lokka service and the Kafka message broker. The interaction between these components introduces additional computational overhead, increasing CPU utilization. When Tikiri communicates with the Lokka service, it must send and receive messages through the Kafka message broker. This communication involves various operations, including message encoding, decoding, routing, and network transmission. These tasks require CPU resources to handle the data transformation and transmission processes.

The Kafka message broker also requires computational resources to manage the messaging infrastructure. It handles message storage, replication, and distribution across multiple nodes, which can significantly load the CPU. On the other hand, LIoT's lightweight blockchain architecture, combined with the pan digital leader election, allows for optimized resource usage and reduced CPU overhead. The leader election process in LIoT is designed to minimize computational requirements while maintaining efficient consensus within the network.

##### 4.4.2. Memory usage

Memory usage can be recorded periodically to evaluate the resource utilization and efficiency of the blockchain framework. In Figure 5, the comparison of memory usage between LIoT and Tikiri is illustrated, clearly demonstrating LIoT's lower memory usage in contrast to Tikiri. LIoT adopts a minimalist approach by utilizing minimal services. Only essential services are employed, resulting in a reduced memory footprint.



LIoT optimizes memory usage and allocates resources more efficiently by avoiding unnecessary or redundant services. Additionally, LIoT maintains the ledger in a JSON format, a lightweight data-interchange format that is easy to parse and consumes less memory than other formats such as XML. LIoT minimizes the memory requirements for storing and processing the ledger information by storing the ledger data in JSON format.

On the other hand, Tikiri runs multiple microservices, including gateway, Lokka, storage, and Kafka. Each microservice contributes to the system's overall memory usage. The gateway handles the communication between external entities and the Tikiri network, while the Lokka service facilitates the interaction with the Kafka message broker. The Storage service manages the data storage and retrieval, and Kafka acts as a message broker for exchanging data between different components. Running multiple microservices simultaneously increases the memory overhead. Each service requires memory allocation for executing tasks and storing data, resulting in higher memory usage than LIoT's minimalist approach.

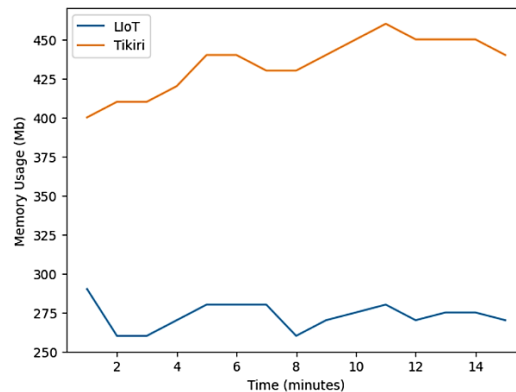


Figure 5. Memory usage comparison: LIoT vs Tikiri

#### 4.4.3. Transaction and block generation time

In this evaluation, the comparison of transaction execution time between LIoT and Tikiri is conducted, as shown in Figure 6. Transaction execution time is when a transaction is added to a block from the data pool until all nodes have confirmed it. LIoT takes less time than Tikiri. There are several factors contributing to this difference. One of the critical factors is the efficiency of the consensus mechanism employed by LIoT in electing the leader and generating blocks that contain the transactions. LIoT utilizes a consensus algorithm that enables a faster consensus among the participating nodes in the network.

Data hash generation time, block hash generation time, the time necessary for transaction validation by the leader node, and block broadcast time between peers contribute to block generation time. In Figure 7, the block generation time is compared with the number of transactions in the block for LIoT and Tikiri. It is evident that as the number of transactions increases, the block generation time also increases.

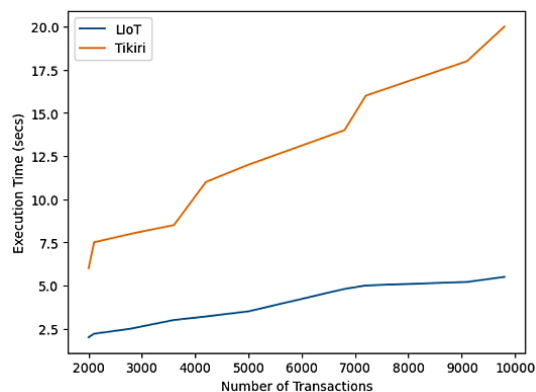


Figure 6. Transaction execution time with different transaction sets: LIoT vs Tikiri

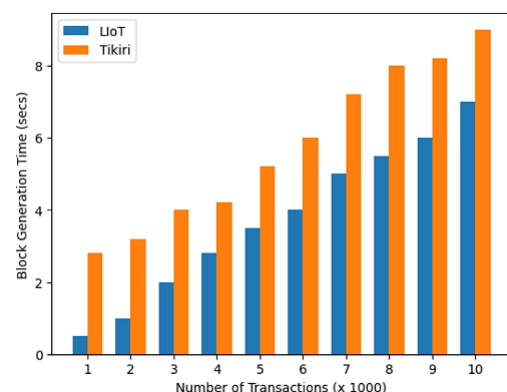


Figure 7. Comparison of block generation time: LIoT vs Tikiri

However, when comparing LIoT with Tikiri, it can be observed that LIoT demonstrates better block generation time. This can be attributed to the two blockchain's architectural differences and design choices. In Tikiri, block generation involves communication with the Kafka service to obtain transactions. This communication adds additional overhead and latency to the block generation time. On the other hand, LIoT stores transactions directly in the node's data pool, eliminating the need for external communication. This streamlined approach reduces the block generation time in LIoT.

#### 4.4.4. Comparison of data structures (linked list vs skip list)

The comparison between the linked list and skip list data structures reveals the need for more efficient data structures such as skip list when handling large-scale and complex data. As the volume of data grows, traditional structures like the linked list face challenges in maintaining optimal performance. To address these limitations, the skip list data structure has emerged as a viable solution for storing and retrieving data in blockchain systems, such as LIoT.

Compared to the linear search time complexity of the linked list, the skip list provides a logarithmic search time complexity. As the number of elements grows, the time to search for a specific element increases slowly. Consequently, skip list significantly outperforms the linked list in search operations, particularly for larger datasets. Regarding insertion operations, the skip list dynamically adjusts its structure to maintain balance and optimize search times. In contrast, linked lists require modifications to neighboring nodes during insertions, potentially resulting in slower performance for frequent updates.

Furthermore, the self-balancing nature of skip list ensures a relatively even distribution of elements across the levels. This property guarantees consistent performance over time. Such consistency and scalability make skip list a preferred choice for managing the attendance ledger in LIoT.

In Table 1, the comparison table depicts the performance of the skip list and linked list based on the manipulated number of records. In Figure 8, the chart illustrates the time taken (in milliseconds) for different numbers of records. The skip list is represented by the blue bars, while the linked list is by the red bars. The x-axis represents the number of records, and the y-axis represents the time taken. This comparison chart visually represents the efficiency and performance advantages of the skip list over the linked list.

Table 1. Query transaction results

Number of records	Skip list (ms)	Linked list (ms)
42	0.5	4.2
84	0.6	6.2
168	0.7	14.5
250	0.8	16.9
250	0.9	17.3
500	1.3	18.6
750	1.5	19.8
1000	2.7	20.3

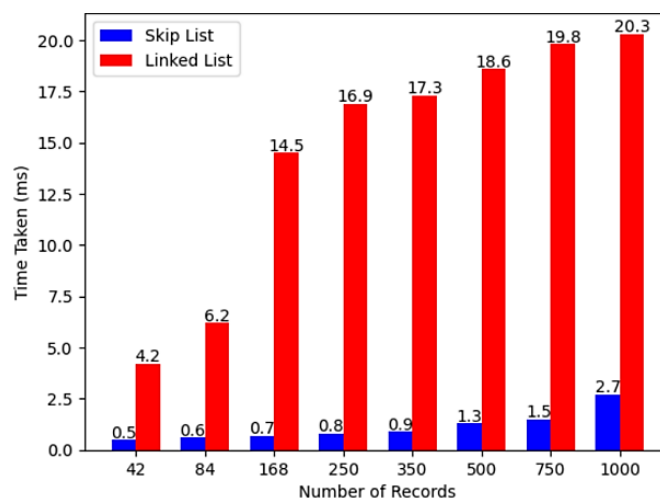


Figure 8. Comparison of query transaction

#### 4.4.5. Performance of LIoT with different scenarios (low, medium, and high transactions)

LIoT demonstrates robust performance across different transaction scenarios: low, medium, and high. Implementing the skip list data structure further enhances LIoT's efficiency and scalability. In low transaction scenarios, with a relatively minor number of transactions, LIoT achieves remarkable response times. For instance, when handling 42 transactions, LIoT exhibits a negligible latency of 0.5 milliseconds. As the transaction volume increases to the medium range, such as 250 transactions, LIoT maintains excellent performance with a minimal latency of 0.8 milliseconds. Even in high transaction scenarios, where the workload intensifies, LIoT delivers impressive results. With 1000 transactions, LIoT handles the increased load efficiently, with a low latency of 2.7 milliseconds. Figure 9 highlights the effectiveness of the skip list data structure in enabling LIoT to process transactions swiftly and reliably across different workload scenarios.

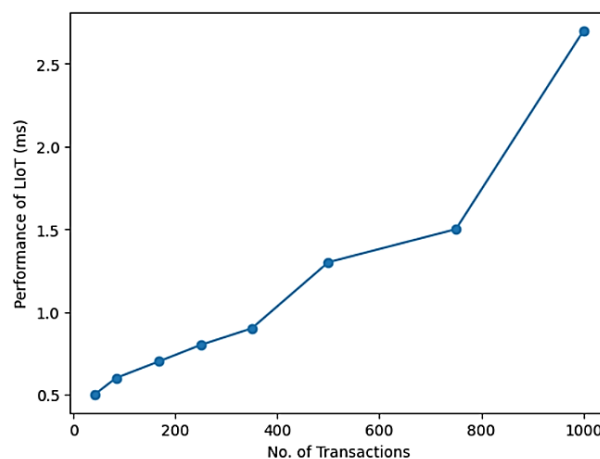


Figure 9. Performance of LIoT with different scenarios (low, medium and high transactions)

## 5. CONCLUSION

The proposed LIoT framework is designed to deploy dApps (a kind of application which uses blockchain technology) in IoT devices, and its primary objective is to establish a highly secure and decentralized network infrastructure. By enabling direct communication and data exchange between nodes without relying on a centralized authority, this approach significantly enhances the overall security and resilience of the blockchain framework. Another critical aspect of the LIoT framework is the customized consensus algorithm explicitly tailored for the LIoT framework. The Pan Digital Leader Election algorithm ensures a fair and decentralized leader selection process within the LIoT framework, enabling the efficient and secure creation of blocks in the blockchain. Furthermore, the LIoT framework incorporates the Skip List data structure, revolutionizing the storage and retrieval of transactions and ledger information. The Skip List offers superior search compared to a traditional data structure like a linked list. With its multi-level structure and forward pointers, the Skip List enables efficient traversal and navigation through the blockchain, significantly enhancing the performance of the LIoT framework. Pan Digital leader election algorithm combined with the Skip List data structure revolutionizes the storage, retrieval, and consensus aspects of the LIoT framework, paving the way for scalable and reliable blockchain solutions in resource-constrained environments. Embedding LIoT in docker enhances portability by encapsulating it within a container, including dependencies. Docker facilitates seamless container communication, fostering efficient collaboration and streamlined deployment in diverse environments. In addition to the features mentioned above, future work for the LIoT framework includes the integration of proof-of-work and smart contract concepts.

## FUNDING INFORMATION

This work is funded by the AICTE Research Promotion Scheme (RPS), File No.8-135/FDC/RPS/POLICY-1/2021-2022, dated 18th Feb 2022. The authors thank the funding agency for their continuous support in completing this work.

**AUTHOR CONTRIBUTIONS STATEMENT:**

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Suresh Jaganathan	✓	✓		✓	✓	✓	✓			✓		✓	✓	✓
Karthika Veeramani		✓	✓	✓		✓		✓	✓		✓			

C : Conceptualization

M : Methodology

So : Software

Va : Validation

Fo : Formal analysis

I : Investigation

R : Resources

D : Data Curation

O : Writing - Original Draft

E : Writing - Review & Editing

Vi : Visualization

Su : Supervision

P : Project administration

Fu : Funding acquisition

**CONFLICT OF INTEREST STATEMENT**

Authors state no conflict of interest.

**DATA AVAILABILITY**

Data availability is not applicable to this paper as no new data were created or analyzed in this study.




**REFERENCES**

- [1] V. Karthika and S. Jaganathan, "A quick synopsis of blockchain technology," *International Journal of Blockchains and Cryptocurrencies*, vol. 1, no. 1, 2019, doi: 10.1504/IJBC.2019.101852.
- [2] Y. Chae, "Blockchain technology in education: a comprehensive review in transparency," in *2023 Congress in Computer Science, Computer Engineering, & Applied Computing (CSCE)*, 2023, pp. 850–857, doi: 10.1109/CSCE60160.2023.00144.
- [3] S. U. Abas, F. Duran, and A. Tekerek, "A Raspberry Pi based blockchain application on IoT security," *Expert Systems with Applications*, vol. 229, p. 120486, 2023, doi: 10.1016/j.eswa.2023.120486.
- [4] A. Al Sadawi, M. S. Hassan, and M. Ndiaye, "A survey on the integration of blockchain with IoT to enhance performance and eliminate challenges," *IEEE Access*, vol. 9, pp. 54478–54497, 2021, doi: 10.1109/ACCESS.2021.3070555.
- [5] C. Xu, K. Wang, G. Xu, P. Li, S. Guo, and J. Luo, "Making big data open in collaborative edges: a blockchain-based framework with reduced resource requirements," in *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1–6, doi: 10.1109/ICC.2018.8422561.
- [6] A. M. Langer, "Blockchain analysis and design," in *Analysis and Design of Next-Generation Software Architectures*, Cham: Springer International Publishing, 2020, pp. 149–164.
- [7] F. Alderazi, "Security of internet of things: a review of challenges with integrating blockchain with IoT," in *2022 2nd International Conference on Computing and Information Technology (ICCIT)*, 2022, pp. 154–160, doi: 10.1109/ICCIT52419.2022.9711650.
- [8] A. Meneghetti, M. Sala, and D. Taufer, "A survey on PoW-based consensus," *Annals of Emerging Technologies in Computing*, vol. 4, no. 1, pp. 8–18, 2020, doi: 10.33166/AETiC.2020.01.002.
- [9] C. Lepore, M. Ceria, A. Visconti, U. P. Rao, K. A. Shah, and L. Zanolini, "A survey on blockchain consensus with a performance comparison of PoW, PoS and pure PoS," *Mathematics*, vol. 8, no. 10, pp. 1–26, 2020, doi: 10.3390/math8101782.
- [10] T. Combe, A. Martin, and R. Di Pietro, "To docker or not to docker: a security perspective," *IEEE Cloud Computing*, vol. 3, no. 5, pp. 54–62, 2016, doi: 10.1109/MCC.2016.100.
- [11] K. Wang, Y. Wang, Y. Sun, S. Guo, and J. Wu, "Green industrial internet of things architecture: an energy-efficient perspective," *IEEE Communications Magazine*, vol. 54, no. 11, pp. 48–54, 2016, doi: 10.1109/MCOM.2016.1600399CM.
- [12] D. Pavithran, K. Shaalan, J. N. Al-Karaki, and A. Gawanmeh, "Towards building a blockchain framework for IoT," *Cluster Computing*, vol. 23, no. 3, pp. 2089–2103, 2020, doi: 10.1007/s10586-020-03059-5.
- [13] E. Bandara, D. Tosh, P. Foytik, S. Shetty, N. Ranasinghe, and K. De Zoysa, "Tikiri—towards a lightweight blockchain for IoT," *Future Generation Computer Systems*, vol. 119, pp. 154–165, 2021, doi: 10.1016/j.future.2021.02.006.
- [14] E. Bandara *et al.*, "Mystiko—blockchain meets big data," in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 3024–3032, doi: 10.1109/BigData.2018.8622341.
- [15] E. Bandara, X. Liang, P. Foytik, S. Shetty, N. Ranasinghe, and K. De Zoysa, "Rahasak—scalable blockchain architecture for enterprise applications," *Journal of Systems Architecture*, vol. 116, 2021, doi: 10.1016/j.sysarc.2021.102061.
- [16] Y. Liu, K. Wang, Y. Lin, and W. Xu, "LightChain: a lightweight blockchain system for the industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3571–3581, 2019, doi: 10.1109/TII.2019.2904049.
- [17] D. Hanggoro and R. F. Sari, "A review of lightweight blockchain technology implementation to the internet of things," in *2019 IEEE R10 Humanitarian Technology Conference (R10-HTC)(47129)*, 2019, pp. 275–280, doi: 10.1109/R10-HTC47129.2019.9042431.
- [18] Y. Hassanzadeh-Nazarabadi, A. Küpçü, and Ö. Özkasap, "LightChain: scalable DHT-based blockchain," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 10, pp. 2582–2593, 2021, doi: 10.1109/TPDS.2021.3071176.
- [19] B. Seok, J. Park, and J. H. Park, "A lightweight hash-based blockchain architecture for industrial IoT," *Applied Sciences*, vol. 9, no. 18, 2019, doi: 10.3390/app9183740.
- [20] M. Zamani, M. Movahedi, and M. Raykova, "Rapidchain: scaling blockchain via full sharding," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 931–948, doi: 10.1145/3243734.3243853.




- [21] A. R. Shahid, N. Pissinou, C. Staier, and R. Kwan, "Sensor-chain: a lightweight scalable blockchain framework for internet of things," in *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2019, pp. 1154–1161, doi: 10.1109/iThings/GreenCom/CPSCom/SmartData.2019.00195.
- [22] D. Na and S. Park, "Fusion chain: a decentralized lightweight blockchain for iot security and privacy," *Electronics*, vol. 10, no. 4, pp. 1–18, 2021, doi: 10.3390/electronics10040391.
- [23] M. Maroufi, R. Abdolee, B. M. Tazekand, and S. A. Mortezaei, "Lightweight blockchain-based architecture for 5G enabled IoT," *IEEE Access*, vol. 11, pp. 60223–60239, 2023, doi: 10.1109/ACCESS.2023.3284471.
- [24] B. B. Rad, H. J. Bhatti, and M. Ahmadi, "An introduction to Docker and analysis of its performance," *IJCSNS International Journal of Computer Science and Network Security*, vol. 17, no. 3, pp. 228–235, 2017.
- [25] D. Pavithran and K. Shaalan, "An optimal consensus node selection process for IoT blockchain," in *2019 Sixth HCT Information Technology Trends (ITT)*, 2019, pp. 115–119, doi: 10.1109/ITT48889.2019.9075069.
- [26] R. C. Lunardi, M. Alharby, H. C. Nunes, A. F. Zorzo, C. Dong, and A. Van Moorsel, "Context-based consensus for appendable-block blockchains," in *2020 IEEE International Conference on Blockchain (Blockchain)*, 2020, pp. 401–408, doi: 10.1109/Blockchain50366.2020.00058.
- [27] G. Adu-boateng and M. N. Anyanwu, "Skip list: implementation, optimization and web search," *International Journal of Experimental Algorithms*, vol. 5, no. 1, pp. 7–13, 2015.
- [28] J. Zhang *et al.*, "S3: a scalable in-memory skip-list index for key-value store," *Proceedings of the VLDB Endowment*, vol. 12, no. 12, pp. 2183–2194, 2018, doi: 10.14778/3352063.3352134.

## BIOGRAPHIES OF AUTHORS



**Suresh Jaganathan**    is Associate Professor in the Department of Computer Science and Engineering, Sri Sivasubramaniya Nadar College of Engineering, has more than 26 years of teaching experience. He received his Ph.D. in computer science from Jawaharlal Nehru Technological University, Hyderabad, M.E. software engineering from Anna University and B.E. in computer science and engineering from Madurai Kamarajar University, Madurai. He has more than 30 publications in referred international journals and conferences. Apart from this, to his credit, he has two patents in image processing and has written a book on "Cloud computing: a practical approach for learning and implementation", published by Pearson Publications. He is an active reviewer in reputed journals (Elsevier - Journal of Networks and Computer Applications, Computer in Biology and Medicine) and co-investigator for the SSN-NIVIDA GPU Education Centre. His areas of interest are distributed computing, deep learning, data analytics, machine learning, and blockchain technology. He can be contacted at email: sureshj@ssn.edu.in.



**Karthika Veeramani**    is Assistant Professor in the School of Computer Science and Engineering at Vellore Institute of Technology, Chennai. She received her M.E. in computer science and engineering from Anna University and her B.Tech. in information technology from the University College of Engineering, BIT Campus, Tiruchirappalli. Before her current role, she gained two years of industrial experience as a programmer analyst at Cognizant Technology Solutions, Chennai. She holds an Indian patent on regularized discriminant analysis. She has published papers at reputed international conferences and journals and has published one book chapter in IGI Global. Her research areas include big data analytics, machine learning, and blockchain technology. She can be contacted at email: karthika.v@vit.ac.in.