Blockchain-Enabled Security Framework for Enhancing IoT Networks: A Two-Layer Approach

Hosny H. Abo Emira[1], Ahmed A. Elngar[2], Mohammed Kayed[3]
Faculty of Computer Science, Nahda University (NUB), Beni-Suef City, Egypt [1]
Faculty of Computers and Artificial Intelligence, Beni-Suef University, Beni-Suef City, Egypt [2, 3]

***Abstract*—The increasing proliferation of Internet of Things (IoT) nodes poses significant security challenges for their network's communication. Blockchain technology, with its decentralized and distributed nature, has the potential to address these security concerns within IoT networks. LEACH (Low Energy Adaptive Clustering Hierarchy) algorithm and blockchain technology enhance IoT network security, enabling energy-efficient data management and transaction integrity, enhancing network lifespan and protection. This paper presents a security model that combines the LEACH algorithm and blockchain technology to improve IoT networks' security. The LEACH algorithm forms clusters of IoT devices, with a designated cluster head (CH) responsible for data aggregation and forwarding. Our model incorporates blockchain technology's core principles and cryptographic foundations, providing additional security measures. It consists of two main layers: the LEACH clustering-based routing protocol, which forms clusters and layers, and a blockchain simulator module. The LEACH algorithm enhances energy consumption, enables efficient data management within clusters, and ensures the integrity, transparency, and immutability of transactions. Our model is implemented on a simulator, allowing for experimentation and modification to evaluate the performance and effectiveness of the security enhanced IoT network model. Our results demonstrate the effectiveness of the proposed enhanced LEACH algorithm compared to previous algorithms, in which the last node died after 1868 transactions. As well as the results of the pro-posed framework, which record 0.058% of the state rate and 2.75 Throughput. Simulation results are validated with respect to previous algorithms, and it obtained higher accuracy compared to them.**

***Keywords*—Internet of Things (IoT); Blockchain (BC); LEACH; clustering; authentication; security**

I. INTRODUCTION

Internet of Things (IoT) security faces numerous challenges in ensuring the protection of connected devices and the data they generate [1]. The sheer scale and heterogeneity of IoT systems, along with limited computational resources and varying security measures, make it challenging to implement standardized security protocols [2]. Challenges also arise from the massive volume of data generated, the lack of comprehensive security regulations, and the complexity of managing security updates [3]. To address these challenges, a holistic approach is required.

One potential solution to various IoT security challenges is blockchain technology. Blockchain's decentralized and immutable ledger ensures data integrity, transparency and prevents tampering [4]. Cryptographic algorithms enable secure authentication and identity management, limiting access to trusted devices. The distributed nature of blockchain eliminates single points of failure, enhancing system resilience [5]. Additionally, blockchain facilitates secure and private data sharing through encryption and selective disclosure, maintaining confidentiality. Smart contracts automate trust and enforce predefined rules, reducing the risk of errors or malicious actions [6]. By leveraging blockchain, IoT systems can benefit from enhanced security, data integrity, privacy, and resilience.

To further enhance IoT security, the LEACH (Low-Energy Adaptive Clustering Hierarchy) algorithm can be integrated with blockchain [7]. Originally designed for wireless sensor networks, LEACH focuses on energy optimization and efficient data management through clustering. By combining LEACH with blockchain, the security and trustworthiness of IoT networks can be enhanced [8]. The algorithm ensures an even distribution of energy load among devices, extending their lifespan and reducing the risk of vulnerabilities [9]. The blockchain provides a decentralized and immutable ledger for securely recording and verifying IoT transactions, ensuring data integrity and transparency. Together, the LEACH algorithm and blockchain technology offer a comprehensive approach to IoT security, addressing energy efficiency, data integrity, and trust, ultimately strengthening the overall security of IoT networks [10].

In this paper, we present a two-layer BC security model that aims to protect IoT networks while simplifying the implementation process. Our proposed model leverages the inherent features of blockchain technology, such as immutability, transparency, and decentralization, to enhance the security of IoT communication. By incorporating blockchain into the architecture, we establish a secure and tamper-proof environment for IoT devices. Additionally, we propose an enhanced LEACH algorithm, powered by fuzzy logic, for efficient data aggregation in IoT-enabled applications, with a specific focus on maximizing network lifetime. Our proposed algorithm optimizes the energy consumption and communication strategies of IoT devices, prolonging the overall lifespan of the network. Through extensive simulations, we demonstrate the effectiveness of our proposed enhanced LEACH algorithm compared to similar works, highlighting its superior performance in terms of network lifetime and energy efficiency.

To support the analysis and evaluation of diverse blockchain systems and their deployments, we introduce a

comprehensive framework and software tool. This framework enables the construction and simulation of discrete-event dynamic system models for blockchain systems. At the core of the framework lies the Base Model, which encompasses fundamental model constructs shared among various blockchain systems and is organized into abstract layers such as network and consensus. The Base Model provides adaptability and extensibility, allowing for the incorporation of specific system or deployment details. Furthermore, we provide a detailed description of the implementation of the simulator within the proposed framework, along with its application to the Ethereum blockchain. To validate the simulation results, we compare our findings to real-life systems and past research reported in previous studies to ensure their dependability and correctness.

In summary, the main contribution of this paper is threefold. First, we present a two-layer Blockchain Security model that safeguards IoT networks and simplifies their implementation. Second, we propose an enhanced LEACH algorithm that maximizes network lifetime through efficient data aggregation techniques. Finally, we utilized a comprehensive framework and software tool for constructing and simulating blockchain system models, facilitating analysis and evaluation. Collectively, these contributions provide a solid foundation for enhancing the security, reliability, and efficiency of IoT networks through the integration of blockchain technology.

The paper is organized as follows in the following sections: The second section discusses the related work. The third section examines the Proposed Methodology. The fourth section describes the model's implementation. The fifth section discusses the experimental results and validation. The last section summarizes the paper's contribution.

II. Related Work

Recently, there has been a growing interest in integrating blockchain technology into IoT. However, very few researchers were interested in how BCs can help in IoT device authentication requirements. In this section, we review the existing literature that focuses on integrating blockchain into IoT ecosystems, highlighting the scarcity of works that successfully address security requirements in this integration.

presents a summary of proposed solution, technology, advantages, and disadvantages of related works.

Yanhui, Liu, et al. [11] focused on the security of user identification and privacy in IoT. By protecting the user's identity and privacy, it becomes impossible for an attacker to link the acquired data with the actual identity of the user, thereby ensuring the safety of the user. To enhance the reliability of the system, the study employs the features of blockchain, which are immutable and incorruptible. The proposed approach records transaction details of user information using Hyperledger and conceals the actual identity of the user using the ring signature method. To generate the necessary parameters for the signature, the system employs a key generator to provide the system with public parameters and ring membership information. Finally, the study also includes an accountability mechanism to punish any attackers who seek to squander system resources by disclosing the user's identity and therefore denying them access to the system.

Ourad, et al. [12] suggests a blockchain-based approach for secure communication and authentication of IoT devices. The solution exploits the inherent characteristics of blockchain while integrating with pre-existing authentication techniques. The study also includes an accountability mechanism to punish any attackers who seek to squander system resources by disclosing the user's true identity and therefore denying them access to the system.

TABLE I. Summary of Related Work

Ref No.	Proposed Solution	Technology	Advantages	Disadvantages
[11]	Blockchain to protect user identity in IoT	Hyperledger, Ring Signature, Aggregated Signature	Identity privacy, Reliability	Waste system resources
[12]	Blockchain-based authentication for IoT devices	Blockchain, Authentication	Accountability, Tamper-proof logs	Resource constraints
[13]	Improved blockchain-based authentication protocol	Ethereum, Security, Anonymity	Secure access, Formal verification	Computational cost
[14]	IoT blockchain architecture for identity authentication	BCoT, Gateway, Device recognition	Distributed ledger, Feasibility	Device modification
[15, 16]	Multi-layer blockchain security model for 5G-enabled IoT	Hyperledger Fabric, Clustering, Evolutionary Computation	Security, Network authentication	Latency, Throughput
[17]	Secure data dissemination using AI and blockchain	AI-based intrusion detection system, blockchain, smart contracts, Interplanetary File System (IPFS)	Efficient and secure data transmission, resistance to attacks	Cost and complexity of implementing blockchain technology
[18]	Blockchain and DL integrated framework	Digital Twin (DT), Smart Contracts, LSTMSAE, MHSA-based BiGRU	Enhances communication security and data privacy	High computational power requirement, Potential for network congestion

Yavari, Mostafa, et al. [13] reveals that a unique authentication system for the administration of IoT device information based on blockchain is vulnerable to security risks such as secret disclosure, replay, traceability, and reuse attacks, all with a probability of success and a constant complexity of 1. The study also includes an improved blockchain-based authentication protocol (IBCbAP) that provides safe access management as well as anonymity. The JavaScript programming language and the Ethereum local blockchain were used to create the IBCbAP. Additionally, the study

verifies the security of IBCbAP through informal and formal analysis using the Scyther tool.

L Gong, et al. [14] proposes a mechanism and creates an IoT blockchain architecture for storing device identity information in a decentralized ledger. The research also suggests a Blockchain of Things (BCoT) Gateway, which streamlines the recording of authentication transactions in a blockchain network without requiring any changes to existing device hardware or applications. In addition, a new device recognition model that is well-suited for blockchain-based identity authentication is provided, utilizing a novel feature selection mechanism for device traffic flow. Finally, the study develops the BCoT Sentry framework as a reference model for the suggested strategy.

Honar Pajooh, Houshyar, et al. [15, 16] proposed a Multi-layer Blockchain Security model that can safeguard IoT networks and be implemented in a simple manner. The model uses clustering to make the multi-layer structure easier to manage. The IoT network is partitioned into K-unknown clusters using approaches that combine Simulated Annealing and Genetic Algorithms. The chosen cluster chiefs are in charge of local authentication and permission. The suggested concept is built on the open source Hyperledger Fabric Blockchain technology. Base stations securely communicate information with one another using a global blockchain architecture.

Kumar, Prabhat, et al. [17] presents a safe way of data distribution based on AI and blockchain technologies. The technology gathers data from healthcare sensors put around the patient's home and sends it to neighboring edge devices. The acquired data is filtered by an AI-based intrusion detection system positioned at the network's edge. The blockchain is then used to create a secure health monitoring network, in which regular or filtered transactions are forwarded to centralized cloud servers and approved via a smart contract-enabled consensus method. Validated transactions are saved on the cloud's distributed Interplanetary File System (IPFS), and the returned transaction hash is saved on the blockchain ledger at the edge devices, allowing for speedier data sharing.

Kumar, Prabhat, et al. [18] provide a decentralized data processing and learning framework in the Industrial Internet of Things (IIoT) network that combines blockchain technology and Deep Learning (DL). A unique Decision Tree (DT) paradigm is used in the framework to provide a virtual environment for modelling and reproducing IIoT security-critical processes. In the proposed blockchain-based data transmission architecture, smart contracts are employed to ensure data integrity and authenticity. Furthermore, the authors developed a DL approach that uses an Intrusion Detection System (IDS) to evaluate blockchain data. The DL scheme employs the Long Short-Term Memory-Sparse Autoencoder (LSTMSAE) technique to learn spatial-temporal representations, as well as the proposed Multi-Head Self-Attention (MHSA)-based Bidirectional Gated Recurrent Unit (BiGRU) algorithm to learn long-distance features and accurately detect attacks.

Al Ahmed, Mahmoud Tayseer, et al. [19] proposes the Authentication-Chains protocol, a decentralized, distributed blockchain-based authentication mechanism for IoT. The protocol clusters nodes and creates a unique authentication blockchain for each cluster. These cluster chains are linked by another blockchain. The proposed consensus mechanism is based on proof of identity verification to address the limited processing capabilities of IoT devices. The security performance of the protocol is analyzed and tested using cryptographic protocol verifier software, and a test bed based on the Raspberry Pi network is shown to validate the protocol's performance.

A significant study gap in the body of current literature is evident, with few studies devoted to examining the relationship between blockchain technology and IoT device authentication requirements. Although there is a rising interest in incorporating blockchain technology into IoT ecosystems, recent endeavours have primarily concentrated on security considerations or other use cases. It has been largely neglected to address the unique difficulty of guaranteeing strong device authentication inside this integration. The examined publications provide insightful information about several facets of blockchain's potential in the context of the Internet of Things, including how it might improve user privacy, secure communication, access control, and data distribution. Still, there isn't much of a focus on IoT device authentication standards.

III. Proposed Methodology

The proposed model aims to enhance the security of IoT networks by combining the LEACH algorithm and blockchain technology. In this section, we will introduce how The LEACH algorithm optimizes energy consumption and enables efficient data management within clusters, while blockchain technology ensures the integrity, transparency, and immutability of transactions. The LEACH algorithm is utilized to form clusters of IoT devices, with a designated cluster head (CH) responsible for data aggregation and forwarding within each cluster. By employing a randomized rotation of CHs, the LEACH algorithm ensures an even distribution of energy load among devices, thereby extending the network's lifespan. The primary objective of the randomized rotation is to achieve an even distribution of the energy load among devices in the network, which in turn helps to prolong the network's lifespan.

In our proposed model, we have integrated the core principles and cryptographic foundations of blockchain technology to provide additional security measures. Fig. 1 illustrates the system architecture, showcasing the structure of the model and the interaction between its components.

The proposed model consists of two main layers. Firstly, the LEACH clustering-based routing protocol forms clusters and layers within the IoT network. Unsupervised hybrid clustering algorithms are employed to create multiple clusters, with each cluster associated with a robust CH. Within each cluster, IoT devices and nodes undergo authentication and authorization processes, ensuring privacy and security.

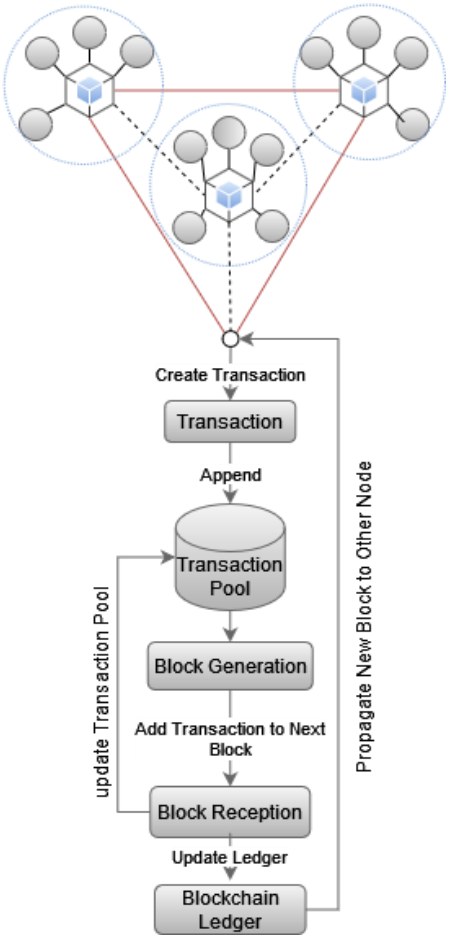

Fig. 1. Proposed model.

Secondly, the model incorporates a blockchain simulator module, which serves as the engine of the simulator. This module comprises four categories: event, scheduler, statistic, and main. It works on three steps: transaction creation and appending to the transaction pool, block generation where transactions are executed and added to the blockchain, and block reception where the new block becomes a permanent part of the blockchain ledger. The transaction pool is also updated accordingly.

By combining the LEACH algorithm and blockchain technology, our proposed model aims to provide enhanced security for IoT networks. The LEACH algorithm optimizes energy consumption and enables efficient data management within clusters, while blockchain technology ensures the integrity, transparency, and immutability of transactions. This integration enhances the overall security and trustworthiness of IoT systems.

A. Cluster Head Selection Algorithim

This level includes Internet of Things objects, nodes, and devices, as well as network elements in charge of communication, network processes, and protocols. Unsupervised hybrid clustering methods divide the IoT network into numerous clusters and layers. Each group is assigned a strong device known as the CH (Cluster Head). IoT devices and nodes are geographically dispersed. To ensure privacy and security inside each cluster, devices are authenticated and authorized to access the network through the use of local authorization and authentication services.

Fig. 2 shows the clustering approach for the IoT network. The clustering is done with the utilization of the LEACH algorithm. LEACH is a clustering-based routing protocol specifically designed for wireless sensor networks with limited energy resources. LEACH helps in reducing the latency and overhead in IoT systems by minimizing communication distances among IoT objects and selected cluster heads. With clustering, fewer nodes require long-distance transmissions to the base station (BS) nodes, resulting in reduced total power consumption and improved network coverage.

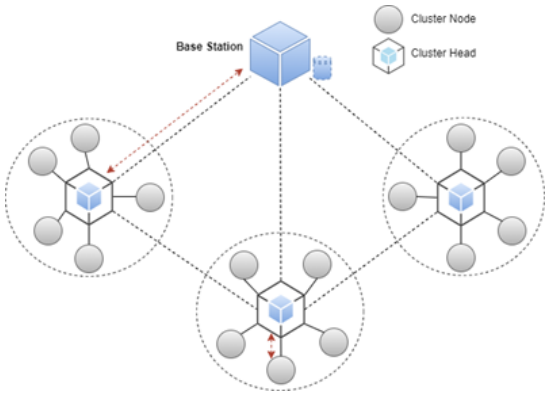

Fig. 2. IoT network clustering scheme.

Our technique, which employs the LEACH clustering algorithm, also aids in maximizing the application efficiency of BC (Blockchain) technology by minimizing deployment complexity. The CH nodes control the whole network, which is separated into non-overlapping clusters, while other cluster members interface with the CH nodes for data transfer.

The goal is to accomplish clustering by deploying the LEACH algorithm throughout the network. In the suggested approach, critical network characteristics such as distance, network coverage, energy, and load are handled as node clustering factors.

Our technique presents LEACH-based clustering, which avoids uniform distribution of nodes and clusters in order to model the heterogeneous character of the IoT network. The overall number of clusters and the number of nodes in each cluster are not predetermined. As a result, the overall network's lifespan rises, but energy dissipation within CH nodes becomes more uniform.

B. Blockchain Simulator

In this section, we present the Base Model that forms the foundation of the Blockchain Simulator. The simulator's purpose is to replicate many sorts of blockchain systems while also allowing for the installation of specific application enhancements as required. We begin by explaining the Blockchain Simulator's design principles and aims, emphasizing generality, flexibility, and accessibility. Following that, we go over the architecture layer by layer, beginning at the network layer and ending with the Consensus Layer. We

define the primary functional components (entities) inside each layer and characterize the tasks or activities they carry out.

1) Principles of design: The Base Model's design is governed by the Blockchain Simulator's key aims, which are as follows:

a) Generality: The Blockchain Simulator will be relevant to a broad range of blockchain systems, setups, and architectural queries.

b) Flexibility: It should be easy for designers or analysts to manipulate the Blockchain Simulator to explore different aspects of blockchain systems.

c) Accessibility: Regardless of the goals, the Blockchain Simulator should remain user-friendly, both for conducting simulation studies and for extending its capabilities.

Designing a tool like the Blockchain Simulator involves striking a balance between generality and extensibility, while simultaneously ensuring simplicity. The Base Model plays a crucial role in achieving this balance, as it determines the level of generality supported by the model class and the ease with which new models can be built. The Base Model is also converted into software modules, it affects the ease with which the Blockchain Simulator may be extended to create more thorough representations of certain blockchain processes.

The Blockchain Simulator caters to the essential components of all blockchains (nodes, transactions, blocks, and incentives). The Base Model defines the level of generality in the model class supported by the Blockchain Simulator, as well as the ease with which new models can be constructed. By representing these building blocks in software modules, the Base Model also influences the extensibility of the Blockchain Simulator, enabling the provision of more detailed models for specific blockchain processes.

2) Network layer: The Blockchain Simulator's Network Layer is made up of two components: The underlying Broadcast protocol and Node. The Node entity oversees keeping system state variables like the transactions pool and the blockchain ledger up to date. The Broadcast protocol specifies how data items such as Blocks and Transactions are broadcast over the network.

The Node object includes both the Blockchain ledger and the Transactions pool entities. These entities are kept and updated in real time by each node. Nodes are represented as objects, each having its own unique ID, balance, local ledger, and transactions pool. When fresh transactions and blocks arrive, the transactions pool and local ledger are represented as expandable array lists. All blockchain implementations have these traits. These traits, however, may be broadened by introducing additional ones.

The Broadcast protocol entity is used for information entity dissemination, which may be properly simulated by taking network configurations, node geographical distribution, and node connection into consideration. It can also be abstracted by considering only a time delay for information propagation among nodes. The simulator is simplified by concealing unneeded features by abstracting the broadcast protocol, resulting in a decrease in network setup settings such as the broadcast protocol, geographical distribution of nodes, and the number of connections per node. The simulator improves efficiency and usability by making the propagation delay the only parameter that may be changed.

3) Consensus layer: The purpose of the Consensus Layer is to describe the rules that nodes must follow in order to agree on the state of the block-chain. This layer consists of four components: a transaction, a block, a transactions pool, and a blockchain ledger. The Block entity is reliant on the Blockchain ledger entity, which is dependent on the Transaction entity. As a result, the blockchain ledger is composed of blocks, which are composed of transactions. Because every produced transaction is placed in the Transactions pool, it is dependent on the Transaction entity. These four entities are managed by the Node entity.

Within the Consensus Layer, the entities perform a range of activities or actions. One example of such activity is the creation of blocks and transactions. Fig. 3 depicts the flow of these operations, which occur continuously since transactions and blocks are regularly brought to the network. Fig. 3 depicts the process for the consensus operations within the Blockchain Simulator's Base Model.

a) Transaction: Transactions are essential components of all blockchain platforms and play an important role in updating the state of the blockchain. A new transaction affects the transaction pool by being added to the network when it is introduced.

We employ two modelling approaches for transactions: full and light. The full technique tracks each individual transaction in the system, allowing for the analysis of transaction latency. While this technique closely resembles real-world blockchain transactions, it demands substantial computing resources and simulation time due to tracking each transaction separately. On the other hand, the light technique focuses on studying the throughput of blockchain systems, disregarding transaction confirmation time within the system. Irrespective of the chosen modelling technique, we represent transactions as objects with various attributes for instance transaction ID, timestamp, contents, size, submitter, and recipient. These attributes are generally shared across most blockchains.

In the full modelling technique, using an array list abstraction, we establish an independent transactions pool for each node in the whole modelling method. Each participating node in the network receives a transaction when it is generated by a node. When a transaction is received, the receiving node adds it to its own pool. This method includes three different activities: transaction creation, transaction propagation, and transaction adding.

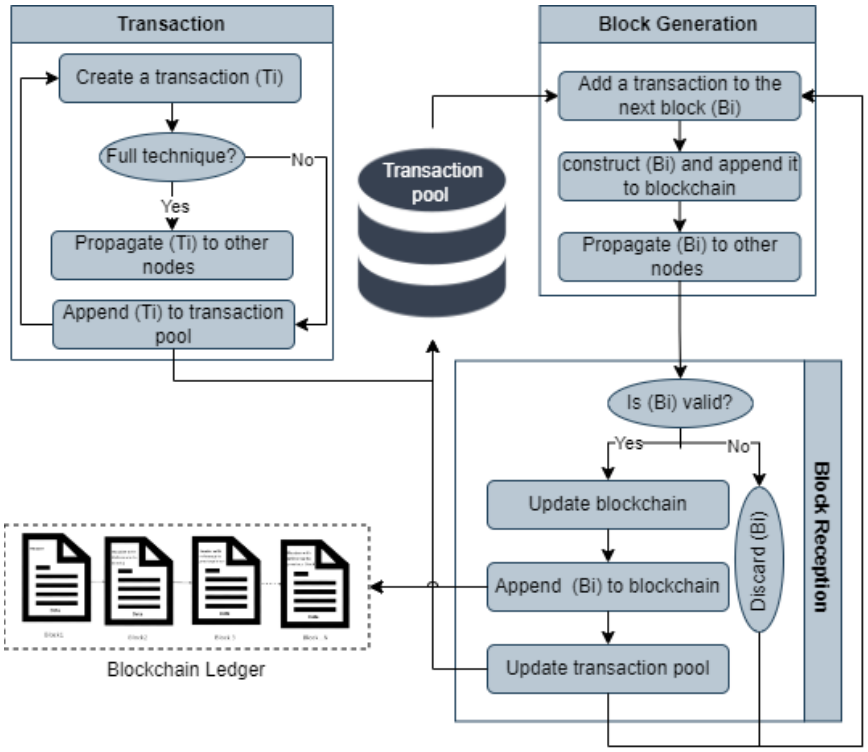

Fig. 3. Workflow of the consensus operations within the blockchain simulator's base model.

In the light modelling technique, we employ a single shared transactions pool among all network nodes. This method offers a simplified approach to modelling transactions by excluding the propagation process and continuous pool updates by nodes. Consequently, the light technique enhances simulation efficiency and speed. However, because individual transactions are not recorded, it cannot give insights into transaction slowness. Nonetheless, it is useful for collecting throughput metrics in blockchain systems. Then Before the mining process, we construct a number of transactions (N) and then append them to the common pool. Miners may utilize this pool to choose numerous transactions to include in their next block. In most cases, the number of transactions (N) should be sufficient for one or two blocks. When a miner creates a successful block, the pool is reset, and a new batch of transactions is added for the following block.

Both modelling methods can be employed in the Blockchain Simulator, granting users the flexibility to choose the method that aligns best with their specific needs. For instance, if the primary focus is on throughput analysis, the full technique may not be necessary due to its significant extension of simulation runtime.

b) Block: Blocks are crucial elements of blockchain systems, which are made up of transactions. When a new block is added, it updates the transactions pool as well as the blockchain ledger. The block's transactions are removed from the pool, and the freshly created block is added to the ledger to update it. Blocks are represented as objects in our models, with different features like depth, block ID, previous block ID, date, size, miner ID, and transactions. The block ID acts as a unique identifier, whereas the block depth reveals its location inside the node's blockchain. The miner ID identifies the node that created the block. As its content, each block provides a list of transactions. These characteristics are prevalent across several blockchain systems.

In the consensus layer, we represent blocks using two basic processes: block creation and block receiving. Block construction refers to the actions taken by a miner to generate and attach a block to the blockchain ledger. These activities include completing the transactions included inside the block, constructing the block, adding it to the local blockchain, and disseminating the block to other nodes in the network. Block reception, on the other side, is concerned with how nodes update their blockchain ledgers in response to new blocks. When a valid block arrives, a recipient node performs three actions: it updates the local blockchain, appends the block to the local blockchain, and updates the transactions pool.

A node validates the authenticity of a new block when it receives it. A block is deemed valid if it was appropriately produced and includes correctly performed transactions. Our attention is drawn to block depth as a measure of validity. To be considered as legitimate, the received block must be deeper than the previous block in the ledger. Any block with a low depth that deviates from the existing blockchain is rejected and deleted. If the received block meets the depth requirement, the recipient node takes three steps: it updates its local blockchain if necessary to align with the received block, it adds the block to its local copy of the blockchain, and it refreshes the transactions pool by removing transactions already executed in the block.

c) Transactions pool and blockchain ledger: The Blockchain Ledger and Transactions Pool are key components of blockchain systems that reflect their status. The transactions pool is updated anytime a new transaction or block enters the network, but the blockchain ledger is only updated when a block is received. Because each node has its own copy of the pool and ledger, The blockchain network's nodes oversee maintaining and updating both.

Due to network propagation delays, nodes may briefly retain distinct viewpoints of the blockchain ledger when forks occur. To remedy this, the consensus layer creates rules for nodes to follow to resolve forks and obtain consensus. Popular blockchain systems such as Ethereum and Bitcoin use the longest-chain rule. This rule states that nodes must update their ledgers anytime. They are given a block that adds to a chain that is longer than their own. This method guarantees that nodes retain a synchronized view of the blockchain ledger, encouraging participant consensus.

IV. Proposed Model Implementation

This section contains, the Network Clustering algorithm is introduced, which aims to optimize parameters by iteratively updating them based on an objective function. The algorithm starts with an initial parameter set and iterates through a specified number of iterations. It utilizes a stochastic search process, where the mean guides the search while a random vector introduces randomness. The algorithm aims to find the parameter set that minimizes the objective function. Additionally, the proposed simulator implementation using Python is discussed, including modules such as the Simulator Module, Configuration Module, and the Ethereum module, which is divided into the Network Module and Consensus Module. The simulator enables the customization of parameters and allows users to analyze the performance and behavior of blockchain systems.

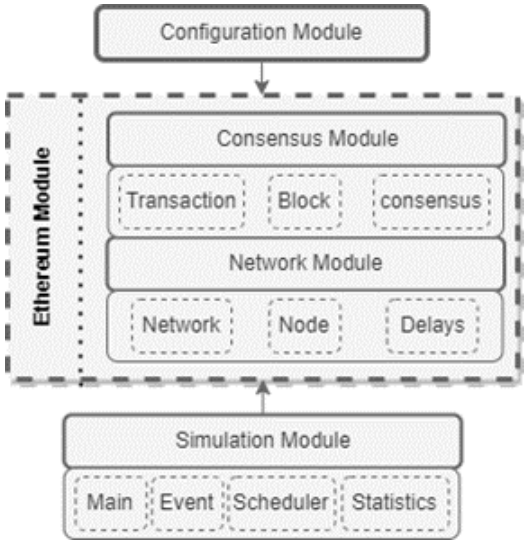

Fig. 4. Proposed implementation modules.

A. Network Clustering

The algorithm represents an optimization process as shown in Algorithm 1 that aims to find the best parameters for a given objective function E. It starts with an initial parameter set to the value P0 and a standard deviation S. The number of iterations K determines how many times the algorithm will repeat.

The algorithm iterates from i = 0 to i $\leqslant$ K, updating the parameters Pi and the mean Mi at each iteration. At each iteration, a random vector Ni is created with a normal distribution, using the mean Mi and standard deviation S.

If the objective function value E(Pi + Ni) is lower than E(Pi), indicating an improvement, the parameters are updated as follows:

- Pi+1 = Pi + Ni, meaning the new parameters are obtained by adding the random vector Ni to the current parameters Pi.
- Mi+1 = α Mi + β Ni, where α and β are constants used to adjust the influence of the mean and the random vector on the update. They can be tuned to control the step size and exploration-exploitation balance.

If the objective function value E(Pi · Ni) is lower than E(Pi), indicating another type of improvement, the parameters are updated as follows:

- Pi+1 = Pi · Ni, meaning the new parameters are obtained by element-wise multiplication between the current parameters Pi and the random vector Ni.
- Mi+1 = γ Mi · Ni, where γ is a constant used to control the influence of the mean and the random vector on the update.

If neither of the above conditions is satisfied, indicating no improvement, the parameters remain the same:

- Pi+1 = Pi
- Mi+1 = δ Mi, where δ is a constant used to control the influence of the mean on the update.

After the iterations are completed, the optimized parameters PK are returned.

Algorithm 1. Optimization of Function Parameters

Algorithm 1: Optimization of Function Parameters

Given: **P0** for the initial parameters, **S** for the standard deviation, and **K** for the number of iterations
*Returns: PK, **the optimised parameters***
Initialize Mi = 0.
for i = 0; i $\leqslant$ K; i = i + 1 repeat
Make a normal distribution random vector **Ni** with a mean **Mi** and standard deviation **S**.
 if E(Pi + Ni) < E(Pi) **then**
  Pi+1 = Pi + Ni
  Mi+1 = α Mi + β Ni
 else if E(Pi Ni) < E(Pi) **then**
  Pi+1 = Pi Ni
  Mi+1 = γ Mi Ni
 else
  Pi+1 = Pi
  Mi+1 = δ Mi
 end
end

This algorithm employs a stochastic search process that explores the parameter space by iteratively updating the parameters based on the objective function's evaluations. The mean Mi helps guide the search process, while the random vector Ni introduces randomness and exploration. The algorithm aims to find the parameter set that minimizes the objective function E.

B. Blockchain Implementation

The proposed simulator implementation is presented by using Python 3.9 with Intel Core i7-2670QM CPU @ 2.20 GHz processor and 8 GB memory. Fig. 4 depicts the key modules. The Simulator Module implements the simulator's engine and is divided into four categories: event, scheduler, statistic, and main. The configuration module enhances the simulation module by allowing the user to customize the simulation model and experiments. The Ethereum module is used to put the suggested Blockchain Simulation into action. It is separated into the following layers: Network and Consensus Modules.

1) Simulation module: The event class specifies the structure of simulation events. A block-level event contains four attributes: node ID, time, type , and block. Provides event preparation at two extraction levels, Examines Blocks (Bi) as Event (E), and Examines Transactions (T) as Event (E). The attribute type specifies how the event should be handled, specifically whether the current event should generate a new block or receive a previously created block. The node ID and timestamp are allocated to the node responsible for handling the event and its time management. The block attribute holds the essential information required for the occurrence of the block.

The scheduler class handles future event scheduling and recording in the queue. The array list queue acts as a storage method for handling future events inside the simulation. It undergoes continuous updates during the simulation process, accommodating the insertion of new events and the removal of existing ones. To illustrate, when an event triggers the creation of a block, the scheduler class acts by scheduling deny reception events for other nodes. These deny reception events are designed to prevent other nodes from receiving the newly created block. By scheduling such events, the simulation ensures that the block remains exclusive to the node that created it, thereby controlling its distribution within the network. It also organizes a new block generation event, in which a miner is chosen to propose and build a new block on top of the previous block.

The Main class creates the environment and then instructs the Scheduler class to schedule some basic events in order to start the simulator. The initial setup involves generating transactions and creating the first block, also known as the genesis block. Following that, the simulation advances by executing events sequentially, one after the other, until either the queue is empty, or the preset simulation time limit is achieved. Throughout this process, the Statistics class plays a crucial role. It collects and maintains the results generated during the simulation, allowing for the calculation of various statistics related to the final output. This includes analyzing block-related data, such as the number of blocks included in the ledger. By utilizing the Statistics class, the simulation can provide valuable insights and metrics that help evaluate the performance and behavior of the system being simulated.

2) Configuration module: This module's goal is to serve as the primary user interface, allowing users to customize parameters relating to nodes, blocks, transactions, consensus, and simulation setups. It allows for the modification of the simulation environment and improves user engagement. Table II shows the input settings that were set before starting the simulator. The duration between blocks, the number of nodes, the number of transactions to be created each second, as well as other characteristics, may all be specified. Furthermore, the simulator disables transactions that are not interesting. This can only be accomplished by setting the option has Trans to "False" and not altering the simulator code. Furthermore, it simulates the transactions by employing an appropriate approach.

TABLE II. INPUT PARAMETERS CONFIGURED BEFORE RUNNING THE SIMULATOR

Type	Parameter	Description
Blocks	B interval	The average time to produce a block in seconds
	B size	Block size in Megabyte (MB)
	B delay	Block propagation delay time in seconds
	Tn	The frequency with which transactions can be created
	T delay	Transactions propagation delay in seconds
	T size	Transaction size in MB
Nodes	Nn	The total number of network nodes
Simulation	Sim time	Duration of the simulation
	Runs	Number of simulations runs

3) Ethereum in proposed simulator: This section discusses the development of simulation classes that reflect the Ethereum Model utilizing the previously described two levels.

Network Module: This module is implemented in two classes: the node class and the network class. The simulator uses node class to define the structure of nodes. Every node is assigned a distinct ID and has an associated balance, implemented as an individual object for each node. The local BC and the transactions pool are modelled by assigning two array lists. When the full transaction approach is used, each node just keeps a transactions pool. Otherwise, shared a common pool by all the nodes. To propagate both blocks and transactions among nodes, the network latency is implemented using network class. In the configuration module, the user of the simulator can configure the time delay in which it is implemented as the latency. As a result, the broadcast protocol, which governs how information entities (such as Blocks and Transactions) are distributed across networks, may be implemented.

Consensus Module: is divided into three classes: Transactions, Block, and Consensus. Every transaction is represented by an object with the following properties: ID, submitter ID, receiver ID, timestamp, value, and size. There are numerous actions to be performed by the entities inside this layer, which may be explained as follows:

- Transactions class:

Create transaction (Ti),

Propagate Ti to other Nodes (Ni) Ti ⟶ Ni,

Append Ti to Transaction Pool (TP) Ni ⟶ TP: [Ti].

- The second class is Block Generation that have three actions:

Execute transactions to the next Bi TP ⟶ Bi: [Ti],

Construct Bi and append it to local BC (LBC)

Bi ⟶ LBC,

Propagate Bi to all Ni LBC ⟶ Ni: [Bi].

- After generating new Bi and stored in LBC the class named block reception check validation of block if it not valid will return it to Block Generation class if it valid:

Update LBC Ni ⟶ LBC,

Append Bi to local BC (LBC) Ni ⟶ LBC: [Bi],

Update TP Ni ⟶ TP: [Bi].

In the setup module, the end user can select transaction sizes as fixed integers or random values derived from a wide range of distributions, including the accelerating distribution. Transaction modelling approaches are also used in this class. Each block is represented as an object that has information like depth, ID, previous ID, timestamp, size, miner ID, and transactions. This class also handles block creation and reception. The consensus class is responsible for implementing the consensus algorithm as well as the process of selecting leaders for the generation and insertion of new blocks into the ledger.

V. Experimental Results and Discussion

This section contains the network environment is simulated to evaluate the performance of the proposed clustering algorithm. The simulation consists of one hundred random generated nodes distributed in a 2-D network. The clustering techniques are applied in MATLAB 2018a, which offers a dependable environment and makes it simple to compare the outcomes. The simulation parameters include simulation area dimensions, number of nodes, initial energy of each node, and packet size. The simulation results demonstrate the effectiveness of the proposed clustering model and the algorithm's ability to reduce total network energy. Fig. 6 shows the operational nodes per transmission, operational nodes per round, and energy consumed per transmission, respectively. Additionally, we compare the performance of different protocols in terms of the number of nodes, energy consumption, first node dies, and last node dies. The results highlight the superiority of the proposed Enhanced LEACH protocol in terms of network lifespan. Furthermore, we compare the network lifetime for different algorithms, with Enhanced LEACH showing significant improvement compared to other protocols. The results validate the efficiency of Enhanced LEACH in maximizing network longevity and optimizing energy utilization in wireless sensor networks.

The section also looks at how different consensus and network characteristics affect the integrity, efficiency, and mining of blockchain systems. The simulator is compared with Ethereum, a common public blockchain, in terms of generating blocks, stale rate, and throughput. The results show that the proposed simulator outperforms other simulators, achieving higher values for all three metrics. Additionally, validation runs are conducted using data gathered from Ethereum to ensure the accuracy and reliability of the simulator. Overall, the simulation results and validation show that the suggested clustering methods and simulator are useful and efficient in assessing blockchain systems. These findings provide valuable insights for optimizing energy consumption, network longevity, and mining decentralization in wireless sensor networks and blockchain technology.

TABLE III. Algorithm Simulation Parameter

Parameters	*Value*
Simulation area dimensions	100 × 100 m
Sink node coordinates	center and corner
Number of nodes	100
Node placement	Random
Initial energy of each node	0.5 Joule
Packet size	8 Bytes

A. Clustering Results and Validation

The performance evaluation of the suggested clustering algorithm was conducted through simulation in a network environment. This environment consisted of a 2-D network with one hundred nodes randomly generated and distributed. MATLAB 2018a was chosen for its reliability in handling clustering algorithms and its ease of simulating various algorithms, enabling a comprehensive comparison of the results. Table III displays the Algorithm parameters used in this instance.

The obtained simulation results demonstrate the usefulness of the suggested clustering model as well as the algorithm's efficiency in reducing overall network energy. Fig. 5 (a) show that the operational nodes per transmission. Fig. 5 (b) illustrates operational nodes per Round. And Fig. 5 (c) shows Energy consumed per transmission.

The provided data in Table VI represents different protocols and their corresponding simulation results in terms of the number of nodes, energy consumption, and the lifespan of the network. The protocols under consideration are Simple LEACH, I-LEACH, Modified LEACH, and proposed Enhanced LEACH.

These protocols are related to wireless sensor networks and aim to optimize energy consumption and network lifetime. Each protocol employs different techniques to manage node energy and prolong the network's operation.

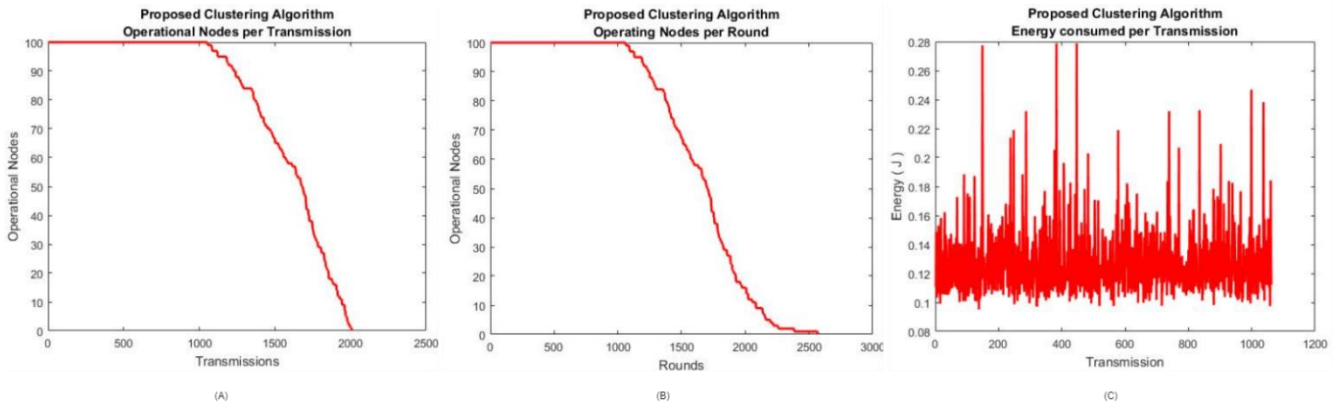

Fig. 5. Performance of the proposed clustering algorithm. (a): Operational nodes per transmission. (b): Operational nodes per Round. (c): Energy consumed per transmission.

It's important to note that the exact number of nodes and energy values may vary depending on the specific implementation and simulation setup. The "First Node Dies" and "Last Node Dies" metrics indicate the sequence in which nodes exhaust their energy, with a higher number indicating a longer network lifespan.

To determine the best protocol, we need to consider the criteria of maximizing network longevity and minimizing energy consumption. As shown in Fig. 6. Comparison of Protocols in Wireless Sensor Networks, it can be observed that the "Enhanced LEACH" protocol offers the longest network lifespan with the last node dying at 1868 units. This indicates better overall network longevity compared to the other protocols listed. However, the specific energy levels and number of nodes are not provided for a direct comparison.

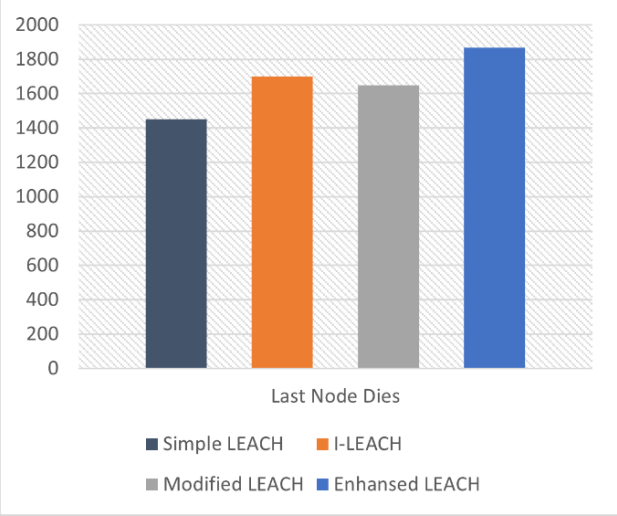

Fig. 6. Comparison of protocols in wireless sensor networks.

The graph in Fig. 7 depicts the network lifetime in terms of the number of living and dead nodes for various algorithms over a given number of rounds. The x-axis shows the number of rounds, which ranges from 0 to 2000, while the y-axis reflects the number of dead nodes, which ranges from 0 to 100.

In the case of Simple LEACH, Modified LEACH, and I-LEACH algorithms, the number of active nodes decreases rapidly with an increase in rounds. After approximately 1650 rounds, all nodes become inactive, resulting in a network with no active nodes.

However, the proposed Enhanced LEACH algorithm shows a significant improvement in terms of network lifetime. Even after 1650 rounds, some nodes managed to remain active until around 1850 rounds. This indicates that Enhanced LEACH extends the network lifetime compared to the other algorithms.

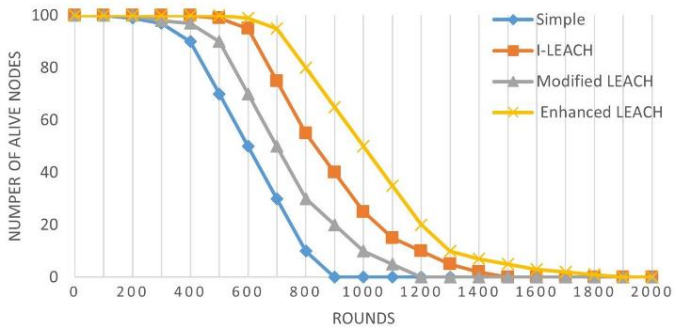

Fig. 7. Lifetime metrics for different protocols.

TABLE IV. COMPARISON OF THE PERFORMANCE OF DIFFERENT PROTOCOLS

Protocols	*Nodes*	*Energy*	*First Node Dies*	*Last Node Dies*
Simple LEACH	100	0.5	780	1450
I-LEACH			1050	1700
Modified LEACH			955	1650
Enhanced LEACH			1100	1868
Simple LEACH	100	0.5	780	1450
I-LEACH			1050	1700

The assignment of different power levels for different forms of communication inside the network is a crucial feature contributing to Enhanced LEACH's enhanced network lifespan. By optimizing power usage and communication strategies, Enhanced LEACH allows for more efficient utilization of energy, resulting in a longer-lasting network.

Overall, the results demonstrate that the Enhanced LEACH algorithm offers improved network lifetime and maximizes the

utilization of sensor nodes' energy resources, making it a promising solution for prolonging the operational time of wireless sensor networks.

B. Blockchain Simulation Results and Validition

The focus of this section was to examine how the state block rate influences mining decentralization and transaction latency, as well as Ethereum's approach to enhancing block mining decentralization. To compare the results, the simulator's findings were contrasted with those of Ethereum, which is one of the most prominent public blockchains. The comparison included proven invariants, such as block creation frequency and the relationship between a miner's hashing share and their likelihood of winning the Proof of Work competition. Additionally, to validate the simulator, sample public data from Ethereum was also utilized.

We conducted a simulation to assess the impact of different consensus and network settings on the reliability, efficiency, and mining aspects of blockchain systems. The purpose was to demonstrate the capabilities of the simulator, including its performance in terms of run time. The simulation employed validation measures like those used in Ethereum but encompassing a wider range of parameter values.

The Table V presents a comparison of different simulators used in the context of blockchain technology. The simulators are evaluated based on three performance metrics: "B included" (daily number of blocks added to the main blockchain), "Stale Rate" (Uncle rate is the proportion of blocks every day that are not in the main chain), and "Throughput" (number of transactions processed per second).

Based on these findings, the suggested simulator outperforms the other simulators in all three categories. It obtains a higher "B included" number, suggesting that it includes more blocks in the main blockchain every day. The "Stale Rate" is also significantly higher, suggesting that there are somewhat more blocks that are not on the main chain. Furthermore, the "Throughput" figure is larger, suggesting that more transactions are performed per second.

To achieve the outcomes Table VI displays the data collected from Ethereum that was utilized as input to the validation runs. That is, Table VII values were utilized for the appropriate input parameters. The data for Ethereum comes from etherscan.io3. Collected for the purpose of this experiment, fit a frequency distribution with the little data collected.

TABLE V. COMPARISON OF DIFFERENT SIMULATORS

Simulator	*B included*	*Stale Rate*	*Throughput*
BlockSim [20, 21]	143 ± 5	0.049% ± 0.069%	2.66 ± 0.09
BlockPerf [22]	146 ± 4	0.025% ± 0.051%	2.69 ± 0.09
SimChain [23]	140 ± 6	0.032% ± 0.058%	2.71 ± 0.07
NetSim [24]	145 ± 3	0.042% ± 0.072%	2.68 ± 0.08
ChainSim [25]	144 ± 4	0.051% ± 0.074%	2.67 ± 0.09
Proposed Simulator	150 ± 7	0.058% ± 0.081%	2.75 ± 0.1

TABLE VI. DATA GATHERED FROM ETHEREUM

Parameters	*Ethereum*
B interval	12.42s
B delay	2.3s
B size	7,997,148 Gas
T size	Distribution

TABLE VII. INPUT PARAMETERS FOR THE SIMULATOR

Parameters	*Value*
Total Number of Devices	50
Total Number of Blocks	100
Blocks in a Chain	50
TX List Size Limit	5
Total Number of Transactions	1000
Average Transaction Latency	0.005
Transaction Throughput	50
Simulation Duration (secs)	20

VI. CONCLUSION

This paper makes significant contributions to the field by tackling the pressing security challenges arising from the exponential growth of IoT nodes. It presents a two-layered blockchain security model, harnessing the inherent decentralization and distribution of blockchain technology to fortify IoT network security. Moreover, the introduction of an enhanced LEACH algorithm, empowered by fuzzy logic, is a noteworthy innovation aimed at optimizing data aggregation and extending the operational lifespan of network nodes. The paper goes even further by introducing a framework that capitalizes on the blockchain's distributed nature to authenticate IoT nodes, structured as a discrete event system model, with a particular focus on network and consensus stages. Through meticulous simulation results, the proposed enhanced LEACH algorithm demonstrates its effectiveness, outperforming previous works and significantly prolonging the network nodes' longevity. In sum, this paper's contributions encompass a two-layered blockchain security model, an enhanced LEACH algorithm, and a blockchain-based authentication framework, all of which exhibit great promise in addressing IoT network security concerns. Future research avenues may delve into exploring the scalability and practical implementation of these proposed models in real-world IoT environments, further solidifying their significance in the domain.

REFERENCES

[1] Tawalbeh, L.a., et al., IoT Privacy and security: Challenges and solutions. Applied Sciences, 2020. 10(12): p. 4102.

[2] Al-Hadhrami, Y. and F.K. Hussain, DDoS attacks in IoT networks: a comprehensive systematic literature review. World Wide Web, 2021. 24(3): p. 971-1001.

[3] Stoyanova, M., et al., A survey on the internet of things (IoT) forensics: challenges, approaches, and open issues. IEEE Communications Surveys & Tutorials, 2020. 22(2): p. 1191-1221.

[4] Tao, X., et al., Confidentiality-minded framework for blockchain-based BIM design collaboration. Automation in Construction, 2022. 136: p. 104172.

[5] Leng, J., et al., Secure blockchain middleware for decentralized iiot towards industry 5.0: A review of architecture, enablers, challenges, and directions. Machines, 2022. 10(10): p. 858.

[6] Hewa, T., M. Ylianttila, and M. Liyanage, Survey on blockchain based smart contracts: Applications, opportunities and challenges. Journal of network and computer applications, 2021. 177: p. 102857.

[7] Bhatia, S., et al., Performance analysis of energy efficient improved LEACH protocol in IoT networks. IET Communications, 2022.

[8] Amjad, S., et al., Blockchain based authentication and cluster head selection using DDR-LEACH in internet of sensor things. Sensors, 2022. 22(5): p. 1972.

[9] Ramya, R., et al. Energy efficient enhanced LEACH protocol for IoT based applications in wireless sensor networks. in 2022 International Conference on Inventive Computation Technologies (ICICT). 2022. IEEE.

[10] Islam, M.J., et al., Blockchain-SDN-based energy-aware and distributed secure architecture for IoT in smart cities. IEEE Internet of Things Journal, 2021. 9(5): p. 3850-3864.

[11] Yanhui, L., et al., Research on identity authentication system of Internet of Things based on blockchain technology. Journal of King Saud University-Computer and Information Sciences, 2022. 34(10): p. 10365-10377.

[12] Ourad, A.Z., B. Belgacem, and K. Salah. Using blockchain for IOT access control and authentication management. in Internet of Things–ICIOT 2018: Third International Conference, Held as Part of the Services Conference Federation, SCF 2018, Seattle, WA, USA, June 25-30, 2018, Proceedings 3. 2018. Springer.

[13] Yavari, M., et al., An improved blockchain-based authentication protocol for iot network management. Security and Communication Networks, 2020. 2020: p. 1-16.

[14] Gong, L., D.M. Alghazzawi, and L. Cheng, BCoT sentry: A blockchain-based identity authentication framework for IoT devices. Information, 2021. 12(5): p. 203.

[15] Honar Pajooh, H., Blockchain for secured IoT and D2D applications over 5G cellular networks: a thesis by publications presented in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer and Electronics Engineering, Massey University, Albany, New Zealand. 2021, Massey University.

[16] Honar Pajooh, H., et al., Multi-layer blockchain-based security architecture for internet of things. Sensors, 2021. 21(3): p. 772.

[17] Kumar, P., et al. A Secure Data Dissemination Scheme for IoT-Based e-Health Systems using AI and Blockchain. in GLOBECOM 2022-2022 IEEE Global Communications Conference. 2022. IEEE.

[18] Kumar, P., et al., Blockchain and Deep Learning for Secure Communication in Digital Twin Empowered Industrial IoT Network. IEEE Transactions on Network Science and Engineering, 2022.

[19] Al Ahmed, M.T., et al., Authentication-Chains: Blockchain-Inspired Lightweight Authentication Protocol for IoT Networks. Electronics, 2023. 12(4): p. 867.

[20] Alharby, M. and A. van Moorsel, Blocksim: An extensible simulation tool for blockchain systems. Frontiers in Blockchain, 2020. 3: p. 28.

[21] Alharby, M., Models and simulation of blockchain systems. 2020, Newcastle University.

[22] Polge, J., et al., BlockPerf: A hybrid blockchain emulator/simulator framework. IEEE Access, 2021. 9: p. 107858-107872.

[23] Ibba, N., T. Wiipongwii, and T.-T.R. Chung, SimChain: Simulator for Supply Chain Decision Making with Blockchain. 2021.

[24] Priyeash, B. and J. Thyagarajan. Performance Evaluation and Comparison Analysis of AODV and RPL Using NetSim in Low Power, Lossy Networks. in International Conference on Futuristic Communication and Network Technologies. 2020. Springer.

[25] Wang, B., et al. Chainsim: A p2p blockchain simulation framework. in Blockchain Technology and Application: Third CCF China Blockchain Conference, CBCC 2020, Jinan, China, December 18-20, 2020, Revised Selected Papers 3. 2021. Springer.