Shatakshi Kokate
Urmila Shrawankar

OPTCHAIN: AN ADVANCED OPTIMIZATION METHOD FOR ENHANCING IOT DATA SECURITY VIA BLOCKCHAIN

Abstract *The increased use of IoT devices in various domains generates abundant data traffic. Securing this data during its transfer and storage is essential. Blockchain is now a trending technology to provide security to the data; however, it is observed that blockchain performs poorly while managing large volume data. To mitigate this issue, an advanced Optchain method to reduce the data size before submitting it to the blockchain network is discussed in this paper. This Optchain method optimizes IoT-generated data using data-classification and compression techniques. The classification of data as relevant or irrelevant is based on predefined thresholds of critical healthcare parameters. Subsequently, the Optchain method employs the Z-standard algorithm for compressing only the relevant data, ensuring efficient storage and faster blockchain transactions. Simulation results using the iFogSim simulator and Ethereum blockchain demonstrated improved storage costs and computational times compared to traditional methods.*

Keywords data classification, data compression, IoT, blockchain, security, random forest, Z-standard

Citation Computer Science 26(4) 2025: 105–130

1. Introduction

Strong security measures are required to safeguard sensitive data during transit and storage due to the exponential growth in the data generation caused by the widespread use of IoT devices. Blockchain technology has a decentralized and secures nature and has emerged as a viable solution to address these security concerns [12]. With the use of several computers and distributed ledger technology, blockchain creates a record of transactions that cannot be changed. This technology ensures data integrity, transparency, and security without the need for a central authority [16].

However, when applied to IoT data management, blockchain technology faces significant challenges – particularly with scalability, storage limitations, latency, and high energy consumption. These challenges arise due to the inherent characteristics of IoT environments, where massive amounts of data are continuously generated by various sensors and devices. Traditional blockchain architectures (designed for smaller, more static data streams) not able to efficiently manage the volume and velocity of IoT data. For example, as data accumulates, the storage requirements on the blockchain grow exponentially, leading to performance bottlenecks and increased energy consumption due to the need for constant consensus and data replication across nodes [5]. Additionally, the latency introduced by blockchain verification processes can hinder real-time IoT applications that demand quick data processing and decision-making [22]. The challenge of scalability in blockchain arises from its decentralized nature, where each node in the network must store and validate every transaction. As the number of transactions increases, the size of the blockchain grows, making it difficult for the nodes to keep up with the storage and processing demands. This is particularly problematic in public blockchain networks, where achieving a consensus across many nodes adds further delays and computational overhead. In the context of IoT data management, scalability becomes a critical issue because IoT systems generate massive amounts of data from various sensors in real time. Traditional blockchain architectures are not optimized to handle such large-scale high-frequency data inputs efficiently. As a result, the network experiences slowdowns, leading to delays in data validation and storage; this can disrupt real-time IoT applications that require quick decision-making and responsiveness.

Given these challenges, there is a need for innovative solutions that not only secure IoT data using blockchain but also address the scalability and storage inefficiencies that arise from large-scale IoT deployments. This paper presents Optchain – an advanced optimization method designed to tackle these issues. The core idea behind Optchain is to reduce the data size before submitting it to the blockchain, thereby improving the processing efficiency, storage management, and transaction speed.

A key motivation for the use of OptChain in the healthcare domain lies in the need to efficiently handle and secure data generated by IoT devices used in critical medical applications. As healthcare IoT devices generate continuous streams of sensitive patient data, it requires real-time management. These data streams are essential for timely medical decisions, but they are also prone to storage and processing bottlenecks when using conventional blockchain architectures. OptChain addresses these challenges by not only reducing the data size but also maintaining the integrity and confidentiality required for healthcare data, thus enhancing the applicability of blockchain technology in healthcare environments. In the case of remote monitoring, for example, a diabetic patient's glucose levels from continuous monitoring sensors can be securely shared with healthcare providers in real-time, ensuring rapid responses while maintaining privacy and security.

Optchain achieves this through a two-step approach. First, IoT data is classified as relevant or irrelevant using a Random Forest machine-learning algorithm, ensuring that only critical data is retained for blockchain processing. Second, the relevant data undergoes compression using the Zstandard algorithm, which effectively reduces the size of the IoT-generated data, further optimizing the storage and transmission. This dual approach mitigates the scalability and storage limitations typically associated with blockchain in IoT environments.

The methodology involves gathering IoT data from various sensors and devices in a healthcare environment, classifying the data using machine-learning models, Zstandard algorithms for compressing the relevant data, and submitting the compressed data to the blockchain network for secure storage and management. Data classification and compression are performed at the primary fog node, while the blockchain is stored on backup fog nodes. This paper integrates fog computing into the proposed system to address the limitations of processing large volumes of IoT data directly on blockchain networks. Fog computing acts as an intermediary layer between IoT devices and the blockchain, enabling data preprocessing, classification, and compression closer to the data source. By offloading these operations to fog nodes, the system reduces its latency, improves its real-time performance, and minimizes the computational burden on blockchain networks. Evaluation metrics such as data-size reduction, storage cost, computational time, throughput, transmission time, latency, and energy consumption were considered to evaluate the performance of the Optchain method. This methodology was implemented to demonstrate significant improvements in managing IoT data with blockchain, ensuring enhanced security, efficiency, and cost-effectiveness.

The problem statement focuses on a solution to optimize the burgeoning amount of IoT data in a secure manner. The primary objectives of this study are as follows:

- develop classification mechanism to identify relevant IoT data;
- implement compression technique to reduce data size before blockchain storage;
- evaluate performance of proposed method using healthcare IoT data in simulated environment.

2. Literature review

In recent years, the integration of blockchain technology with IoT systems has garnered significant attention from researchers aiming to enhance data security, privacy, and efficiency.

2.1. Blockchain technology

In [12], the authors proposed a system to ensure secure IoT data transmission using blockchain and also used hybrid consensus mechanisms that combined the features of Delegated Proof of Stake (DPoS) and Practical Byzantine Fault Tolerance (PBFT), further enhancing the scalability and reliability of IoT-blockchain systems. This hybrid model supports scalable IoT systems, providing privacy, security, and data efficiency across diverse industries.

In [16], Puneet et al. addressed medical information security and privacy using a lightweight cryptographic system with blockchain. In healthcare systems, data security and connectivity are crucial for effective patient data management, where blockchain improves secure data storage and transmission. The proposed work integrated the Intuitionistic Derivative Symmetrical Encryption (IDSE) algorithm with blockchain using a Differential Hashing Pattern (DHP) for key generation to enhance the security, reduce the data size, and improve the transmission speed.

In [5], the authors integrated a blockchain protocol with the Multi-Objective Squirrel Search Optimization Algorithm (MOSSA) to enhance security and scalability in healthcare data management. By optimizing blockchain parameters such as block size, transaction size, and channels, MOSSA improves throughput, efficiency, and reduces delay and computational overhead. The experimental results demonstrate significant performance gains compared to existing methods, making the proposed solution highly effective for securing healthcare systems.

In [22], the authors introduced the concept of BlockCloud, which leverages blockchain as a cloud service to enhance security and efficiency in smart systems. Known for its scalability and cost-effectiveness, cloud computing faces several security concerns. By integrating blockchain technology, this paper addressed these risks by ensuring data integrity through cryptographic methods and electronic wallets. The paper also provided a detailed overview of blockchain applications in cloud computing, highlighting its potential to solve key challenges.

In [14], the authors conducted a survey on blockchain applications, challenges, and opportunities across various domains. Their research emphasized the diverse applications of blockchain technology beyond cryptocurrencies, including its use in securing IoT data. They discussed technical and performance-related challenges of integrating blockchain with IoT such as transaction speed and scalability, which were the critical issues that the proposed method aimed to mitigate.

In [17], the authors explored the integration of blockchain technology with IoT systems, identifying numerous challenges and opportunities. While revolutionary, blockchain must be applied cautiously to IoT systems due to potential risks like high costs and operational challenges. This paper analyzed key points where blockchain could enhance IoT applications and examined existing platforms, emphasizing the importance of addressing challenges such as scalability, security, and privacy to ensure their successful integration.

2.2. Data classification

In [10], the authors proposed a system to enhance data integrity and reduce computational load on cloud servers in IoT environments. By employing data-classification techniques such as k-nearest neighbor and Complement Naive Bayes, the proposed system ensured that only relevant data was transmitted to the cloud, thereby improving the efficiency and security in data transmission.

In [26], the authors conducted a performance analysis and comparison of machine-learning and deep-learning algorithms for classifying IoT data. The authors evaluated various algorithms to determine their effectiveness and efficiency in handling the large and complex data sets generated by IoT devices. Their findings highlighted the strengths and limitations of different approaches.

In [19], the authors examined IoT and big-data categorization in the healthcare industry, focusing on the significance of effective data-classification methods in managing large volumes of healthcare data. The authors investigated how the large-scale data generated by IoT devices can be effectively classified and analyzed to improve healthcare outcomes. They discussed various classification techniques and their applications in managing and interpreting healthcare data.

In [21], ANN and Naive Bayes classification algorithms were subjected to a performance analysis, which shed light on their suitability for various data-classification tasks. This analysis aided in placing the performance of several machine-learning algorithms for the Internet of Things data in context.

In [18], the authors focused on the classification of glucose data for monitoring diabetic patients. The authors presented a method to enhance the accuracy and reliability of glucose level predictions using advanced classification techniques. This approach aimed to improve patient monitoring and management by providing more precise and timely data analysis.

In [13], the author utilized the random forest algorithm for the classification of big IoT, demonstrating its suitability for handling large data sets and optimal feature selection. By applying the random forest algorithm, the authors aimed to efficiently manage and analyze the vast amounts of data generated by IoT devices.

In [9], the authors offered an empirical evaluation of supervised machine-learning algorithms for IoT data, including information about how well different algorithms performed. This paper conducted a comprehensive analysis of five supervised machine-learning algorithms: KNN, Naive Bayes, Decision Tree, Random Forest, and Logistic Regression-on IoT data sets. The Decision Tree algorithm outperformed the others with 97% accuracy, while Random Forest and KNN showed similar results; Naive Bayes and Logistic Regression performed the worst.

Table 1 provides a concise overview of each classification algorithm, highlighting their respective advantages and disadvantages. Random Forest reduces overfitting and provides robust performance by aggregating predictions from individual trees. Also, Random Forest offers a well-balanced combination of accuracy, robustness, and interpretability, making it a strong contender for healthcare IoT data-classification tasks.

Table 1
Comparison of data-classification algorithms

Sr. No.	Algorithn	Advantages	Limitations	Support IoT
1	Decision Tree [8] [9] [11] [13]	Easy to interpret	Prone to overfitting	Yes
2	Random Forest [8] [9] [12] [13]	Reduces overfitting	Computationally expensive	Yes
3	K-Nearest Neigbor [8] [9] [11] [13]	Simple and intuitive	Sensitive to noisy or irrelevant features	Yes
4	Logistic Regression [8] [13]	Provides interpretable results	Limited to linear decision boundaries	No
5	Naive Bayes [8] [9] [10] [11] [13]	Fast training and prediction	Assumes independence among features	No
6	Gaussian Naive Bayes [12]	Speedy execution time	Less delay	No
7	Support Vector Machine [9] [11]	Effective for high-dimensional data	Requires tuning of hyper parameters	Yes
8	Artificial Neural Network [8] [11] [13]	Suitable for complex, non-linear data	Requires large amounts of data and computation	No
9	Convolutional Neural Network [9] [13]	Better execution time	Requires large amounts of labeled data	No

2.3. Data compression

In [7], the authors utilized the Zstandard compression technique to significantly reduce the data sent from IoT devices to the fog server, achieving a compression ratio of 70% with minimal resource consumption. Raspberry Pi 4 and sensors like DS18B20 and MAX30102 were used to gather data, and Python was employed for programming, making the system highly efficient for healthcare applications in terms of compression ratio, throughput, and latency. For future improvements, combining data compression with cryptographic techniques is suggested to enhance security and transmission speed.

In the digital era, medical records are stored electronically, and the growing population significantly increases the need for storage space. Since lossy compression cannot be applied due to the need for full data recovery, [25] proposed a fast and efficient lossless data-compression technique using LZW. The method leveraged data redundancy to compress records and ensured quick and complete recovery, being implemented using Python and HADOOP for storing compressed data.

In [27], the authors carried out a comparison of lossless text-compression techniques and presented a Tamil compression method. By introducing a novel approach tailored to Tamil script characteristics, the authors contributed to advancing language-specific compression methodologies. Their research aimed to improve data-storage efficiency and transmission speeds for Tamil language applications.

In [24], the authors introduced the Tiny Anomaly Compressor (TAC) – a novel data-compression algorithm that leveraged the TinyML approach to enable real-time data analysis on small devices without needing predefined mathematical models. TAC achieved a high compression rate of 98.33% and outperformed other algorithms like Swing Door Trending (SDT) and Discrete Cosine Transform (DCT) in terms of compression error and signal-to-noise ratio.

In [8], the authors addressed the energy-consumption challenge in wireless sensor networks by exploring data-compression techniques to improve network lifetime. The proposed adaptive lossless data-compression algorithm (ALDC) reduced data-transmission size significantly, achieving energy savings between 67.8 and 73.2%. A further optimized algorithm encoding residue samples demonstrated even better energy efficiency, with a 76.8% energy saving and zero redundancy.

In [20], the study focused on optimizing data transmission over constrained networks. The Internet of Things (IoT) plays a crucial role in modern autonomous systems but faces challenges due to limited resources, making data compression essential. For sensitive data transmitted over the low bandwidths IoT networks like narrowband IoT (NBIoT) and LTE-M, lossless compression techniques are preferred to maintain data integrity. This paper discussed the need for such techniques in these networks and highlighted the challenges and recent compression methods used in low-power wide-area networks.

In [3], the authors conducted a comparative study of various compression techniques used in IoT environments. With the rapid growth of connected devices generating large amounts of data, resource limitations like memory, processing power, and battery life have become critical. Applying data-compression techniques can help reduce energy consumption, storage needs, and transmission costs. This paper surveyed and compared popular IoT compression techniques, analyzing their attributes (such as lossless or lossy compression, limitations, and implementation locations).

In [1], the authors investigated edge-computing methods for lossless compression in critical IoT applications. Data compression at smart Edge/Fog-based gateways is essential for reducing transmission latency and increasing network bandwidth in time-sensitive IoT applications like healthcare. This paper analyzed various lossless data-compression algorithms run on Edge/Fog gateways, evaluating their performance in terms of latency and compression rate. It provided insights to help select the appropriate compression algorithm for time-critical IoT use cases.

In [2], the authors introduced a lightweight lossless-compression method for multi-sensor systems, focusing on N-dimensional data with multi-sensor systems. The research addressed the need for efficient data handling in sensor networks by proposing methods that reduced data size while preserving data integrity. Table 2 provides a concise overview of each compression algorithm, highlighting their respective advantages and disadvantages. Depending on the specific requirements of the compression task (such as compression ratio, speed, and resource constraints), the proposed system chooses a Z-standard algorithm for data-compression needs.

Table 2

Comparison of data-compression algorithms

Sr. No.	Algorithm	Advantages	Limitations
1	Z-standard [14]	High compression ratios, Memory efficiency	Limited Adoption in some environments
2	Lempel-Ziv-Welch (LZW) [15] [16][18] [20] [21]	High compression ratios	Requires dictionary initialization
3	Huffman [16] [18] [19]	Assigning shorter codes to more frequent symbols, reducing overall file size	Requires predefined frequency table; inefficient for small or dynamically changing data sets
4	Deflate (ZIP) [15] [18]	Widely supported (e.g., ZIP file format)	Slower compression compared to LZ77 alone
5	Run-Length Encoding (RLE) [18] [19] [20]	Simple and efficient for run-length data	Limited effectiveness for non-repetitive data
6	Burrows-Wheeler Transform (BWT) [16] [20]	Effective for compressing text data	Requires additional encoding for data storage
7	Arithmetic Coding [17] [19]	Handles non-uniform probability distributions	High computational complexity
8	Lempel-Ziv-Markov chain algorithm (LZMA) [20] [21]	High compression ratios	Higher computational complexity than LZ77
9	Prediction by Partial Matching (PPM) [15] [16] [20]	Adapts to changing data patterns	Slower compression compared to simpler methods

2.4. Fog computing

In [11], the authors explored the integration of fog computing into the cloud to secure data, highlighting the advantages of using fog computing for data processing and security. The authors discussed how fog computing was useful in securing the data generated by IoT devices and delivered to the cloud.

In [6], the authors proposed a reinforcement learning-based technique for cutting the latency in cloud-fog gaming. Their research emphasized the importance of reducing latency in cloud-fog environments for improving transaction speeds and processing efficiency in blockchain networks.

In [23], the authors presented a model integrating fog computing and blockchain for identification and authentication in healthcare IoT systems. Their approach enhanced security and efficiency by leveraging fog nodes for local processing and blockchain for secure immutable data storage. This combined model addressed key challenges in healthcare IoT such as data integrity and real-time authentication, providing a robust solution for secure healthcare data management.

In [15], the authors discussed fog computing architectures, applications, and security issues, providing a detailed overview of fog computing's role in enhancing IoT system performance and security. While fog computing offers many advantages, this article highlighted the security, privacy, and safety challenges due to its distributed architecture and proposed solutions to address these concerns for further development.

Table 3 shows an analysis of existing study in terms of the techniques and methodologies used as well as their strengths and weaknesses.

Table 3
Analysis of existing study

Sr. No.	Citation	Technique used	Methodology	Merits	Limitation
1	[2]	Intuitionist Derivative Symmetrical Encryption (IDSE), Blockchain	Integration of IDSE with blockchain using Differential Hashing Pattern (DHP) for key generation	Enhanced security, reduced data size, improved transmission speed in healthcare systems	Not tested in large-scale real-world scenarios
2	[3]	Multi-Objective Squirrel Search Optimization Algorithm (MOSSA)	Optimization of blockchain parameters (block size, transaction size) using MOSSA for healthcare data management	Improved scalability, throughput, and efficiency in securing healthcare data	High complexity of optimization process

Table 3 cont.

Sr. No.	Citation	Technique used	Methodology	Merits	Limitation
3	[9]	J48 Classification algorithm	Applied IoT healthcare data classification using machine-learning techniques for better patient monitoring	Improves IoT-based healthcare decision-making	Security and data privacy risk
4	[17]	Tiny Anomaly Compressor (TAC), TinyML	Local machine-learning-based data compression on constrained IoT devices without predefined mathematical models	High compression rates (up to 98.33%) with low error, enabling local analysis and real-time performance on constrained devices	Data granularity may be lost
5	[18]	Adaptive Lossless Data Compression (ALDC)	MATLAB coding and simulation of residue encoding to reduce data	Demonstrated energy savings of up to 73.2%, efficient data compression, and improved network lifetime	Does not scale well in more complex IoT networks

The review studies reveal several limitations, such as data overload, lack of optimization, security and privacy concerns, and insufficient data granularity. To address these challenges, the proposed Optchain method has been developed, offering a more efficient solution by optimizing data management while ensuring security and maintaining data relevance.

3. System architecture

Figure 1 depicts the system architecture of the proposed Optchain method for securing IoT data via blockchain. The system architecture comprises several interconnected components to facilitate the secure handling and processing of IoT data. With fog computing with a blockchain layer, the process initiates with data generated from IoT, which are then channeled into a data classifier. The fog computing is used in this architecture to process data at the local level. The proposed system leverages fog nodes to enhance the performance of the blockchain-IoT ecosystem. IoT devices send raw data to nearby fog nodes, which handle data classification (relevant vs. irrelevant) and apply Zstandard compression. Only the compressed relevant data is forwarded to the blockchain network for secure storage. This layered approach ensures low latency, reduces network traffic, and enhances scalability by offloading intensive data-processing tasks to fog nodes, making the system more efficient for real-time IoT applications.

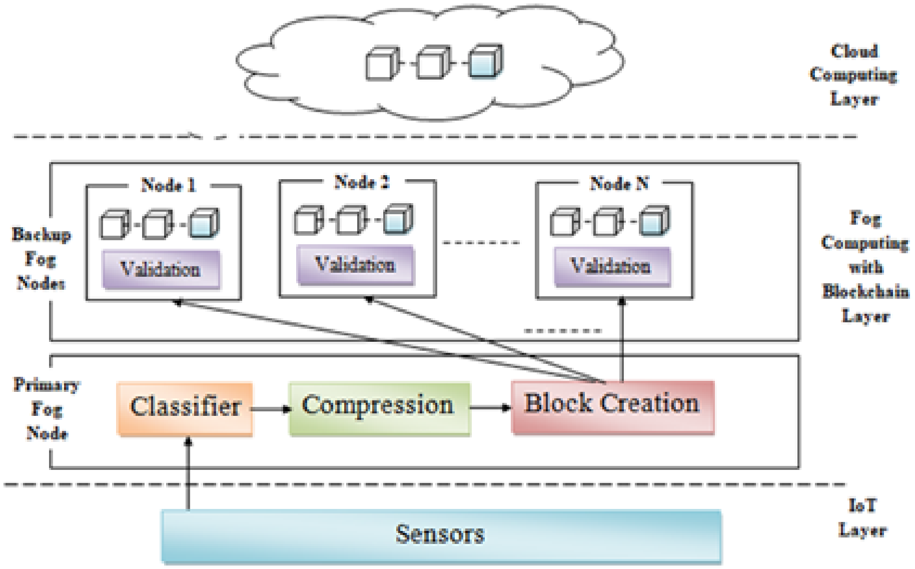


Figure 1. Proposed system architecture

The classifier employs the Random Forest algorithm to effectively segregate relevant data from irrelevant data, ensuring efficient data management. Once identified, the relevant data undergoes further processing through a data compressor. Utilizing the Zstandard algorithm, the data is compressed to optimize the storage and transmission efficiency while maintaining its integrity. Subsequently, the compressed sensitive data is encapsulated into a block – ready for integration into the existing blockchain ledger. Before its incorporation, the newly formed block undergoes rigorous validation to ensure its accuracy, consistency, and compliance with predefined protocols. Upon successful validation, the block is seamlessly appended to the blockchain ledger, thus contributing to the immutable record of transactions and events within the system. For long-term storage, the blockchain will be stored on the cloud.

A detailed description of each component of the system architecture is as follows:

1. IoT Layer (Sensors):
   - The system starts with IoT devices (e.g., healthcare sensors) that gather data such as patient vitals.
   - These sensors continuously collect raw data and send it to the primary fog node.

2. Primary Fog Nodes:
   - The primary fog nodes classify data collected from the sensors. It separates relevant data from irrelevant data, reducing the amount of unnecessary information to be processed.
   - After classification, the Compression module further reduces the size of the relevant data. Compression is essential for optimizing storage and transmission before data is sent to the blockchain network.
   - The compressed data is passed to the Block Creation module, which organizes the data into blocks for inclusion in the blockchain network.
3. Backup Fog Nodes:
   - Once the block is created, it is distributed across multiple fog nodes (Node 1, Node 2, ..., Node N) – each performing a validation process to verify the integrity and authenticity of the data.
   - These fog nodes are part of a decentralized network that performs consensus operations to ensure that the stored data is valid and immutable. Backup fog nodes ensure fault tolerance and redundancy.
4. Cloud-Computing Layer:
   - After validation in the fog-computing layer, the verified data blocks are sent to the cloud for long-term storage, deeper analysis, or additional processing.

This comprehensive architecture ensures the secure, efficient, and transparent handling of sensitive data from IoT devices, providing a robust foundation for data management and integrity within the ecosystem. Also, this architecture bridges the gap between resource-constrained IoT devices and blockchain networks, ensuring faster transactions, improved data management, and seamless system performance.

4. Implementation

The proposed method is implemented by using the following steps:

4.1. IoT data collection and transmission

The healthcare data set used in this research was sourced from Kaggle [4] and was comprised of patient-monitoring data generated by IoT healthcare devices, including temperature sensors, pulse oximeters, blood pressure monitors, ECGs, EMGs, and glucometers. It featured various attributes such as device type, operational metrics, security events, and potential threats. The data set description is https://www.kaggle.com/datasets/faisalmalik/iot-healthcare-security-dataset?resource=download.

4.2. Classification phase (using Random Forest algorithm)

The classification process involves categorizing the IoT data using the Random Forest (RF) algorithm, distinguishing between relevant and irrelevant data. Subsequently, the relevant data is selected as input for the data-compression model. The random forest operates as an ensemble-learning algorithm, relying on a fundamental principle: the construction of small decision trees with minimal features is computationally economical. Many small and weak decision trees are generated in parallel and combined by majority voting or averaging to produce a robust learner. Random forests have been shown to be among the most accurate learning algorithms available empirically.

Algorithm 1 Random Forest

```
Precondition: A training set S:=(x1,y1),...,(xn,yn), features F, and numbers of trees in forest B
**Function** RandomForest(S,F)
$H \leftarrow \theta$
for $i \in 1, ..., B$ do
$S(i) \leftarrow$ A bootstrap sample from S
$hi \leftarrow$ RandomizedTreeLearn (S(i), F)
$H \leftarrow H \cup hi$
end for
return H
**end Function**
**Function** RandomizedTreeLearn(S,F)
At each node:
$f \leftarrow$ very small subset of F
Split on best feature in F
return The learned tree
Output code for P
```

The provided pseudo-code in Algorithm 1 outlines the Random Forest algorithm. The random forest algorithm consists of two functions: RandomForest, and RandomizedTreeLearn. In the RandomForest function, a set of decision trees H is built by repeatedly creating bootstrap samples by taking the data from s (which is a training set). The Randomized Tree Learn function is used for learning a randomized decision tree for each sample. By choosing a small subset of features f at each node and splitting on the best features in f, the RandomizedTreeLearn function constructs a decision tree.

Consider an example of a pulse oximeter – an IoT healthcare device that measures blood oxygen levels (SpO2) and heart rate (BPM). Along with these primary metrics, the device may collect additional data such as device temperature, battery status, signal strength, time stamps, and error flags; it also sometimes generates null values or redundant values. However, not all of these features are equally important for clinical decision-making. The Random Forest model evaluates the importance of each feature based on how well it contributes to predicting health outcomes, such as detecting low oxygen levels or abnormal heart rates. For example, SpO2 and heart rate are ranked as high-priority features due to their direct relevance to patient monitoring, while signal strength and frequent time stamps receive lower scores and are excluded to reduce the data size. By keeping only relevant features like SpO2, heart rate, and occasional error flags, the system ensures that essential healthcare information is retained without unnecessary overhead. This selective feature reduction optimizes storage, improves processing efficiency, and allows for faster real-time analysis – critical for time-sensitive healthcare applications.

4.3. Compression phase (using Zstandard algorithm)

The Zstandard algorithm is employed in the compression process to reduce the size of the input IoT data, producing compressed data (CD). This compressed data is then transmitted to generate a block and is subsequently appended to the existing blockchain.

Algorithm 2 Zstandard algorithm

```
Initialize table with single character strings
P=first inout character
WHILE not end of input stream
    C=next input character
    IF P+C is in the string table
        P=P+C
    ELSE
        output the code for P
        add P+C to the string table
        P=C
END WHILE
Output code for P
```

Algorithm 2 shows the Zstandard algorithm, which starts by initializing a table containing single-character strings. It then reads the input stream character by character, building up sequences, and adding them to the string table as it encounters new combinations. If a combination is already present in the table, it continues to build on the current sequence. When a new combination is encountered, it outputs the code for the previous sequence, adds the new combination to the table, and resets the current sequence to the latest character. This procedure keeps going until the input stream ends and the final code for the last sequence is outputted.

To understand how the Zstandard algorithm functions, let us consider a simple example involving healthcare IoT data. Imagine a heart-rate-monitoring device that continuously records heart-rate measurements and transmits them to the healthcare system for real-time analysis and storage.

The raw data stream is shown as follows:

Raw Data (Heart Rate Readings):
78, 78, 80, 80, 80, 81, 82, 82, 82, 82, 83, 83, 83, 83, 83.

Since IoT data often contains repetitive patterns, Zstandard leverages these repetitions to efficiently compress the data. The Zstandard algorithm compresses data by building a dictionary of repeating patterns and encoding the data using shorter codes for frequently occurring values.

Step 1: Dictionary Construction
The algorithm first scans the data to identify frequent patterns or sequences. In this example, repeated sequences like (78, 78), (80, 80, 80), and (83, 83, 83, 83, 83) are detected.

Step 2: Encoding Using Shorter Codes
Zstandard assigns shorter codes to these repeating patterns to replace the original data:

- 78, 78 → Code A;
- 80, 80, 80 → Code B;
- 83, 83, 83, 83, 83 → Code C.

After encoding, the data stream might look something like this:
Compressed Data:
A, B, 81, 82, 82, C.

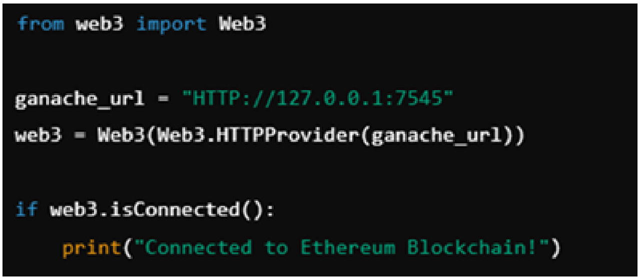
Step 3: Decompression (Reconstruction)
When the data needs to be retrieved, Zstandard decompresses the codes back into their original values using the dictionary.
Decompressed Data:
78, 78, 80, 80, 80, 81, 82, 82, 82, 82, 83, 83, 83, 83, 83

5. Experimental setup

The problem of handling and safeguarding the enormous volumes of data generated by IoT devices in the healthcare industry was tackled in this experiment. The iFogSim simulator was used in the experimental setup to simulate an IoT environment for healthcare, where a variety of sensors produced continuous data streams. Sorting this data according to predetermined healthcare parameters into groups that were useful and not useful was the first stage. The pertinent data was subjected to a compression method after classification in order to greatly minimize its size.

Key performance parameters were monitored during the experiment, such as the percentage decrease in data size, processing effectiveness, transaction speed, and blockchain-transaction cost. The anticipated results encompassed a significant decrease in the data dimensions and enhanced the processing effectiveness as a result of the diminished data volume, expedited blockchain transactions, and decreased transaction expenses. This configuration showed the potential for improved performance and cost-effectiveness in managing healthcare data as well as the viability of utilizing blockchain technology to secure IoT data.

The integration of an Ethereum blockchain ensured the security of the compressed data. Using Ganache, a local Ethereum network and a connection to a test network were established. Solidity was used to create smart contracts that managed data submission and retrieval while maintaining data security and integrity. The web3.py module was used in the Python scripts to enable communication between the Ethereum blockchain and the iFogSim simulation. This module enabled the simulation to send transaction requests to the blockchain. The Python code was used to create a Web3 object and connect to the Ganache instance via HTTP (as shown in Figure 2).



```
from web3 import Web3

ganache_url = "HTTP://127.0.0.1:7545"
web3 = Web3(Web3.HTTPProvider(ganache_url))

if web3.isConnected():
    print("Connected to Ethereum Blockchain!")
```

Figure 2. Code of Web3 object

Once connected, the web3.py module allowed the simulation to deploy smart contracts and send IoT-data transactions. For instance, the simulation triggered smart-contract functions like storeData() and retrieveData() through web3.py API calls, ensuring secure and efficient data management.

In terms of the iFogSim configuration, the simulation was set up with various fog nodes – each simulating different healthcare facilities. These nodes handled data streams from sensors deployed in a hospital environment. The healthcare parameters used for the data classification included thresholds for temperature (37^0C), blood pressure (120/80 mmHg), heart rate (60-100 BPM), glucose level (90-130 mg/dL), and oxygen saturation (95-100%). Each sensor's data was monitored in real-time, processed at the fog nodes, and then sent to the blockchain for secure storage.

Moreover, specific simulation scenarios included testing the system's response to varying data loads, simulating critical healthcare events, and analyzing the latency and throughput of the blockchain-based data management. By adjusting the number of sensors, the scalability of the system was tested, providing insights into how well the system could manage increasing amounts of healthcare data.

6. Evaluation metrics

The parameters that were considered for the measurement of results were as follows:

- **Compression Ratio (CR)**
  This is used to measure the amount of data that is compressed; it is the ratio of the size of the compressed data to the uncompressed data.

$$CR = \frac{\textit{Size of Uncompressed Data}}{\textit{Size of Compressed Data}}. \tag{1}$$

- **Compression Factor (CF)**
  This is given by the following formula:

$$CF = \frac{1}{\textit{Compression Ratio}}. \tag{2}$$

- **Saving percentage**
  This is given by the following formula:

$$\frac{\textit{Uncompressed Data-Compressed Data}}{\textit{Uncompressed Data}}. \tag{3}$$

- **Storage cost**
  Storage cost refers to the expense associated with storing the data. In the context of blockchain and IoT systems, the storage cost is crucial, as it can significantly impact the scalability and economic viability of the system (especially when dealing with large volumes of data).
- **Computational Time**
  Computational time is the amount of time required to perform a computational task or process a set of operations.
- **Throughput**
  Throughput is the rate at which a system can process and transmit data over a given period.
- **Compression Ratio:**
  This measures the effectiveness of the data compression by comparing the size of the original data with the compressed data; a higher ratio indicates better compression, reducing storage and transmission overhead.
- **Transmission Time:**
  This refers to the time required to transmit data from one point to another across a network; optimizing the transmission time ensures faster communication and reduces delays.
- **Latency:**
  Latency measures the delay between a request and the corresponding response; lower latency ensures real-time data processing, which is crucial for healthcare IoT applications.
- **Energy Consumption:**
  This indicates the total energy used by devices or systems during the data processing and transmission. Reducing the energy consumption improves the sustainability and efficiency of IoT systems.

7. Results

The results of this study showed that the suggested approach was more effective than traditional compression techniques such as Run Length Encoding (RLE), Shannon-fano, Huffman Coding, and Lempel-Ziv-Welch (LZW) in the context of reducing the data size. These methods were tested on healthcare IoT data sets, which included time-sensitive and sensitive patient data gathered from various sensors. The experiments were conducted using the iFogSim simulator for data generation and processing as well as the Ethereum blockchain for securing the compressed data. The performance metrics were evaluated and normalized to provide a comprehensive comparison, highlighting the advantages and limitations of each method. The experiments were carried out for varying sizes of healthcare IoT data (21,527, 43,527, and 68,574 KB).

Table 4

Comparison of compression techniques and proposed method for different data sizes

Data size	Parameters	RLE	Shannon-Fano	Huffman Coding	LZW	Optchain Method
21,527 KB	Compressed Data Size	10,978	8179	4736	3874	3014
	Compression Ratio	0.51	0.38	0.22	0.18	0.14
	Compression Factor	1.96	2.63	4.54	5.56	7.14
	Saving Percentage	49%	62%	78%	82%	86%
43,527 KB	Compressed Data Size	34,973	26,058	15,086	12,343	5457
	Compression Ratio	0.51	0.38	0.22	0.18	0.13
	Compression Factor	1.96	2.63	4.54	5.56	1.98
	Saving Percentage	49%	62%	78%	82%	87%
68,574 KB	Compressed Data Size	35,173	25,258	14,324	11,587	9150
	Compression Ratio	0.51	0.38	0.22	0.18	0.13
	Compression Factor	1.96	2.63	4.54	5.56	7.49
	Saving Percentage	49%	62%	78%	82%	87%

Table 4 shows that the output values of the parameters for the different data sizes varied in the proposed system, as the incoming data may consist of different amounts of relevant and irrelevant data. The proposed method first categorized the data and then applied the compression to the relevant data only. In contrast, the values remained the same for the other techniques, as they solely applied compression to all incoming data. These outcomes demonstrated how well the suggested strategy worked to get the greatest possible decreases in the data sizes, compression ratios, and saving percentages across the various data sets.

The results obtained after the implementation of the proposed Optchain method were compared with those of the existing Blockchain-MOSSOA and FC-with-Blockchain systems. This evaluation aimed to demonstrate the improvements in storage cost, computational time, and throughput in an IoT-based healthcare environment.

Figure 3 presents a line graph that illustrates the storage costs for Blockchain-MOSSOA, FC-with-Blockchain, and the proposed Optchain method across the different numbers of IoT devices. These values depict the storage costs in terms of memory usage for each approach when managing the different numbers of IoT devices. The proposed Optchain method consistently demonstrated lower storage costs compared to Blockchain-MOSSOA and FC-with-Blockchain, highlighting its efficiency in processing IoT data. On average, the proposed Optchain method reduced storage costs by approximately 27.41% when compared to Blockchain-MOSSOA and by 20.00% when compared to FC-with-Blockchain.

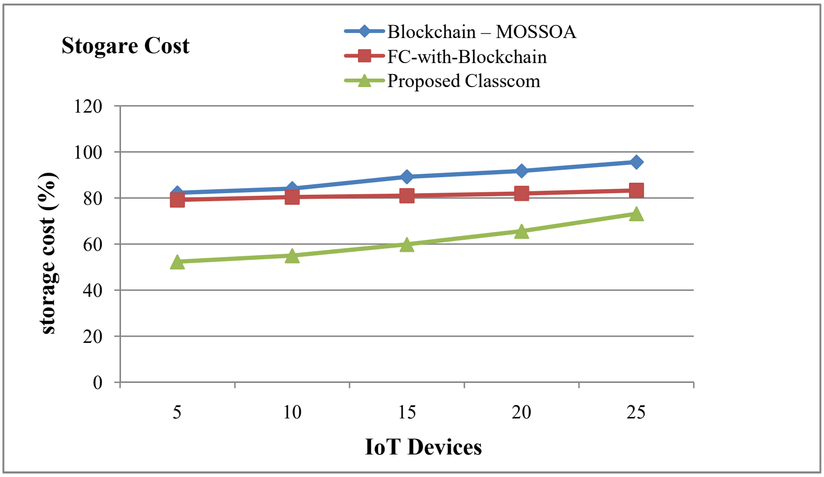


Figure 3. Storage cost comparison of proposed method with existing methods

Figure 4 presents a line graph illustrating the computational time for Blockchain-MOSSOA, FC-with-Blockchain, and the proposed Optchain method across the different numbers of IoT devices. The performance of the computing time for each method is shown by these values. The proposed Optchain method consistently demonstrated lower computational costs as compared to Blockchain-MOSSOA and FC-with-Blockchain, highlighting its efficiency in processing IoT data. On average, the proposed Optchain method reduced computational time by approximately 39.05% when compared to Blockchain-MOSSOA and by 29.89% when compared to FC-with-Blockchain.

Figure 5 illustrates the throughput comparison for Blockchain-MOSSOA, FC-with-Blockchain, and the proposed Optchain method across the varying numbers of IoT devices. The hroughput (measured in transactions per second) indicates the system efficiency in handling data. The proposed Optchain method consistently exhibited higher throughput than both Blockchain-MOSSOA and FC-with-Blockchain for all of the device counts.

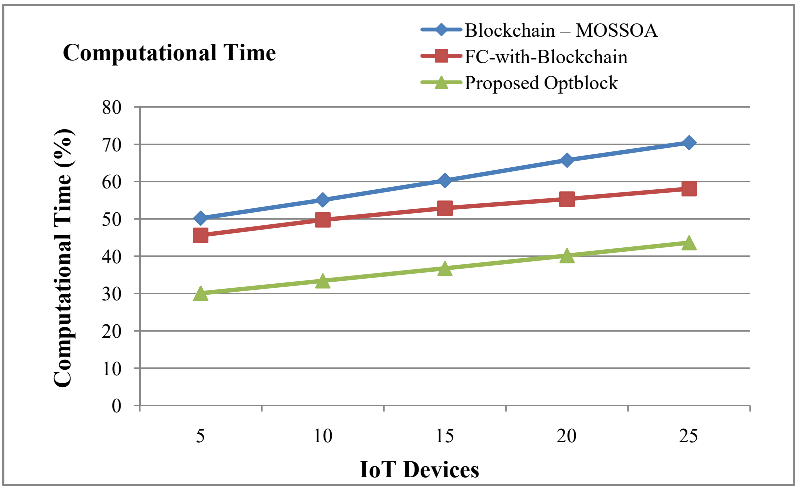


Figure 4. Computational time comparison of proposed method with existing methods

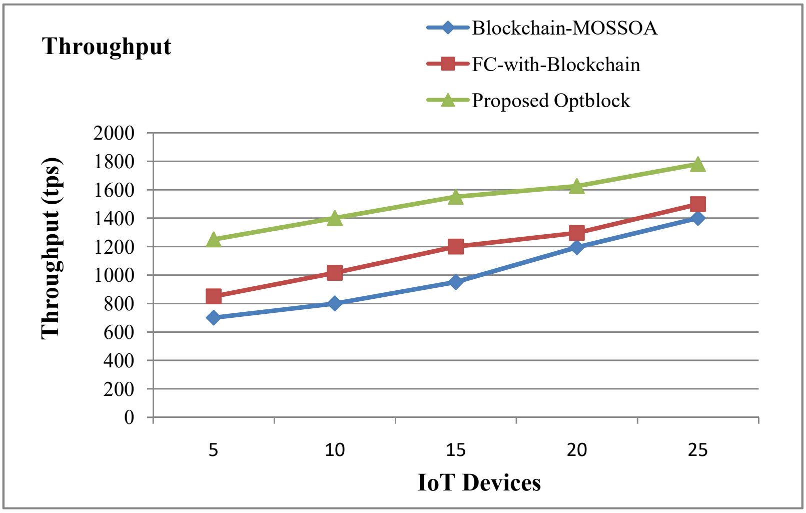


Figure 5. Throughput comparison of proposed method with existing methods

To provide a more detailed analysis, we included quantitative comparisons among the proposed Optchain method and traditional compression techniques such as Huffman coding, Run-Length Encoding (RLE), and Zstandard. These comparisons were based on key performance metrics: compression ratio, transmission time, latency, and energy consumption. The data collected from the healthcare sensors contained a mixture of both low- and high-frequency data, making it an ideal scenario for evaluating the efficiency of the compression methods.

Scenario 1: High-Frequency Data Transmission

For high-frequency data transmission where the amount of data generated was large, Optchain demonstrated a significant reduction in the transmission time and energy consumption as compared to traditional techniques like Huffman coding and Run-Length Encoding (RLE). The performance improvement in such scenarios was primarily attributed to Optchain's ability to intelligently classify relevant and irrelevant data before applying compression. This resulted in reduced data size, faster transmission, and less computational overhead. The high-frequency data set included continuous readings from multiple sensors.

Table 5

Comparative analysis of compression techniques for high-frequency data transmission

Method	Compression ratio [%]	Transmission time [ms]	Latency [ms]	Energy consumption [J]
Proposed Optchain	70	45	15	3.2
Huffman	60	68	20	4.5
Run-Length Encoding	55	75	25	5.1
Zstandard	65	58	18	3.8

Table 5 shows that Optchain outperformed the existing methods by achieving a 70% compression ratio with a 45 ms transmission time, and it significantly reduced energy consumption of 3.2 J; this was approximately 28% lower than the next best method (Zstandard). The key factor in Optchain's superior performance was its ability to classify irrelevant data before applying the compression, which minimized the total data size that needed to be transmitted.

Scenario 2: Low-Frequency Sensitive Data

In contrast, for low-frequency but highly sensitive data such as heart rate monitoring, the proposed method provided an additional layer of security through its blockchain-based framework while maintaining a competitive compression ratio. While capable of reducing the data size, the traditional techniques often did not address security concerns as efficiently as Optchain. This makes Optchain highly suitable for healthcare environments where data integrity and security are critical.

For low-frequency sensitive data such as heart-rate readings, the security of the data transmission and integrity are critical. Here, both compression and security performance were evaluated.

Table 6 shows that Optchain provided a competitive 68% compression ratio while maintaining a transmission time of 50 ms. Importantly, Optchain (with a blockchain-based security mechanism) added an additional layer of data integrity, which was absent in the traditional methods. Despite this added security, Optchain still achieved an energy consumption of 3.5 J, outperforming Huffman and RLE by reducing the energy consumption by 25–30%.

Table 6
Comparative analysis of compression techniques for low-frequency sensitive data

Method	Compression ratio [%]	Transmission time [ms]	Security	Energy consumption [J]
Optchain	68	50	Blockchain Integrated	3.5
Huffman	58	72	None	4.7
Run-Length Encoding	52	80	None	5.3
Zstandard	63	65	None	4.0

These scenarios demonstrated that Optchain significantly outperformed the traditional methods in both high-frequency data transmission and low-frequency security-sensitive contexts. These insights validated its applicability in various healthcare IoT environments – particularly where security and efficient data handling are essential.

8. Discussion

The research presented in this paper proposed an advanced method for managing IoT data within blockchain networks, achieving significant improvements over existing methodologies – particularly in terms of scalability, storage, and processing efficiency. While previous studies have attempted to address these challenges by optimizing blockchain protocols or integrating cloud and fog computing, many solutions have still struggled with the issue of efficiently reducing the massive volume of IoT data.

In real-world IoT environments, the proposed method holds strong potential for practical application. IoT systems generate vast amounts of data, and managing this in a blockchain network often becomes cumbersome. The proposed method has the ability to classify data into relevant and irrelevant categories before compression offers an efficient solution – particularly for environments like healthcare, smart cities, and industrial IoT systems (where data volume and sensitivity are paramount). The method can easily be applied in these practical IoT environments by incorporating it into existing IoT-fog-blockchain architectures, ensuring that only pertinent data is processed and stored; this leads to faster and more efficient transactions as well as reduced storage needs.

A key comparative advantage of this approach is its use of the Zstandard compression algorithm for relevant data. Unlike other existing blockchain-IoT systems (which often try to store all data on-chain), the proposed method smartly reduces the data size before storage, significantly enhancing the blockchain's performance in handling IoT data. The results showed that the proposed method outperformed traditional methods such as standard compression algorithms like Huffman and Real length Encoding in terms of compression ratio and processing speed – especially when tested on healthcare IoT data sets.

During the implementation, however, some limitations were encountered. Even though the classification and compression process drastically reduced the data size, it introduced some complexity in the classification phase; this required fine-tuning for different IoT use cases. Additionally, the initial processing overhead for classifying the data may have affected real-time applications in some resource-constrained IoT environments. Nonetheless, these challenges do not detract from the proposed method's overall applicability and scalability, making it a robust solution for IoT-based blockchain systems.

9. Conclusion

In conclusion, many IoT applications are under increasing pressure to bolster security measures and protect data in the face of growing cyber security threats. While blockchain technology offers a promising solution, its effectiveness in managing the vast volume of healthcare IoT data is constrained by inherent limitations. To address this challenge, an advanced method for combining data-classification and compression techniques has been proposed. Through the systematic classification and compression of IoT data, this method effectively reduces data sizes and minimizes network delays during data transmission. The findings presented in this research underscore the substantial improvements achieved in terms of the reductions in data sizes, processing efficiency, transaction speeds, and transaction costs. Notably, the observed saving percentages of approximately 86 to 87% highlighted the significant reduction in data size, affirming the efficacy of the proposed method. These results suggested that leveraging machine-learning-based classification and advanced compression techniques can greatly enhance network performance and scalability in IoT applications. Additionally, the proposed method effectively addresses the challenge regarding handling large amounts of data faced by blockchain networks. Moving forward, further research and development efforts should focus on optimizing the proposed system for diverse IoT deployment scenarios and exploring additional avenues for improving data-processing efficiency and resource utilization. Overall, this study contributes valuable insights into addressing the inherent challenges of IoT data management and lays the foundation for future advancements in this rapidly evolving field. Implementing the proposed method in real-world IoT environments requires integrating it with the existing IoT infrastructure, ensuring its compatibility with diverse sensor types and communication protocols. Additionally, deploying the blockchain and compression techniques at fog nodes can help manage data locally, reducing latency and improving the overall efficiency of the data processing and security. This method can be further applied to various domains such as smart cities, agriculture, industrial IoT, and environmental monitoring, where the efficient management, processing, and secure transmission of large volumes of IoT data are critical for improving operational performance, scalability, and decision-making.

References

[1] Gia T.N., Qingqing L., Queralta J.P., Tenhunen H., Zou Z., Westerlund T.: Lossless compression techniques in edge computing for mission-critical applications in the IoT. In: *2019 Twelfth International Conference on Mobile Computing and Ubiquitous Network (ICMU)*, pp. 1–2, IEEE, 2019. doi: 10.23919/icmu48249.2019.9006647.

[2] Giorgi G.: Lightweight Lossless Compression for N-Dimensional Data in Multi-Sensor Systems, *IEEE Sensors Journal*, vol. 19(19), pp. 8895–8903, 2019. doi: 10.1109/jsen.2019.2922666.

[3] Hamdan S., Awaian A., Almajali S.: Compression techniques used in iot: A comparitive study. In: *2019 2nd International Conference on new Trends in Computing Sciences (ICTCS)*, pp. 1–5, IEEE, 2019. doi: 10.1109/ictcs.2019.8923112.

[4] IoT Healthcare Security Dataset, https://www.kaggle.com/datasets/faisalmalik/iot-healthcare-security-dataset?resource=download.

[5] Jaishankar B., Vishwakarma S., Mohan P., Pundir A.K.S., Patel I., Arulkumar N.: Blockchain for securing healthcare data using squirrel search optimization algorithm, *Intelligent Automation & Soft Computing*, vol. 32(3), pp. 1815–1829, 2022. doi: 10.32604/iasc.2022.021822.

[6] Jameii S.M., Khanzadi K.: A Latency Reduction Method for Cloud-fog Gaming based on Reinforcement Learning, *International Journal of Engineering*, vol. 35(9), pp. 1674–1681, 2022. doi: 10.5829/ije.2022.35.09c.01.

[7] Kahdim A.N., Manaa M.E.: Design an efficient internet of things data compression for healthcare applications, *Bulletin of Electrical Engineering and Informatics*, vol. 11(3), pp. 1678–1686, 2022. doi: 10.11591/eei.v11i3.3758.

[8] Ketshabetswe K.L., Zungeru A.M., Mtengi B., Lebekwe C.K., Prabaharan S.R.S.: Data compression algorithms for wireless sensor networks: A review and comparison, *IEEE Access*, vol. 9, pp. 136872–136891, 2021. doi: 10.1109/access.2021.3116311.

[9] Khadse V., Mahalle P.N., Biraris S.V.: An empirical comparison of supervised machine learning algorithms for internet of things data. In: *2018 fourth international conference on computing communication control and automation (ICCUBEA)*, pp. 1–6, IEEE, 2018. doi: 10.1109/iccubea.2018.8697476.

[10] Kokate S., Shrawankar U.: An efficient approach for secured data transmission between IoT and Cloud, *Research Reports on Computer Science*, pp. 35–44, 2023. doi: 10.37256/rrcs.2320232628.

[11] Kokate S., Shrawankar U.: Integration of the cloud with fog computing to secure data transmission between iot and cloud. In: *Integration of Cloud Computing with Emerging Technologies*, pp. 83–92, CRC Press, 2023. doi: 10.1201/9781003341437-9.

[12] Kokate S., Shrawankar U.: A Trustworthy IoT to Cloud Data Transmission Frameworks, *Cureus Journal Of Computer Science*, vol. 1(1), 2024. doi: 10.7759/s44389-024-00274-8.

[13] Lakshmanaprabu S.K., Shankar K., Ilayaraja M., Nasir A.W., Vijayakumar V., Chilamkurti N.: Random forest for big data classification in the internet of things using optimal features, *International Journal of Machine Learning and Cybernetics*, vol. 10(10), pp. 2609–2618, 2019. doi: 10.1007/s13042-018-00916-z.

[14] Monrat A.A., Schelén O., Andersson K.: A survey of blockchain from the perspectives of applications, challenges, and opportunities, *IEEE Access*, vol. 7, pp. 117134–117151, 2019. doi: 10.1109/access.2019.2936094.

[15] Neware R., Shrawankar U.: Fog computing architecture, applications and security issues, *International Journal of Fog Computing (IJFC)*, vol. 3(1), pp. 75–105, 2020. doi: 10.4018/ijfc.2020010105.

[16] Puneeth R., Parthasarathy G.: Security and data privacy of medical information in blockchain using lightweight cryptographic system, *International Journal of Engineering*, vol. 36(5), pp. 925–933, 2023. doi: 10.5829/ije.2023.36.05b.09.

[17] Reyna A., Martín C., Chen J., Soler E., Díaz M.: On blockchain and its integration with IoT. Challenges and opportunities, *Future Generation Computer Systems*, vol. 88, pp. 173–190, 2018. doi: 10.1016/j.future.2018.05.046.

[18] Rghioui A., Lloret J., Oumnad A.: Big data classification and internet of things in healthcare, *International Journal of E-Health and Medical Communications (IJEHMC)*, vol. 11(2), pp. 20–37, 2020. doi: 10.4018/978-1-6684-3662-2.ch071.

[19] Rghioui A., Lloret J., Parra L., Sendra S., Oumnad A.: Glucose data classification for diabetic patient monitoring, *Applied Sciences*, vol. 9(20), 4459, 2019. doi: 10.3390/app9204459.

[20] Routray S.K., Javali A., Sahoo A., Semunigus W., Pappa M.: Lossless compression techniques for low bandwidth io ts. In: *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*, pp. 177–181, IEEE, 2020. doi: 10.1109/i-smac49090.2020.9243457.

[21] Saritas M.M., Yasar A.: Performance analysis of ANN and Naive Bayes Classification Algorithm for Data Classification, *International Journal of Intelligent Systems and Applications in Engineering*, vol. 7(2), pp. 88–91, 2019. https://ijisae.org/index.php/IJISAE/article/view/934/585.

[22] Shrawankar U., Shrawankar C.: BlockCloud: Blockchain as a Cloud Service. In: *Blockchain for Smart Systems*, pp. 53–63, Chapman and Hall/CRC, 2022. doi: 10.1201/9781003203933-5.

[23] Shukla S., Thakur S., Hussain S., Breslin J.G., Jameel S.M.: Identification and authentication in healthcare internet-of-things using integrated fog computing based blockchain model, *Internet of Things*, vol. 15, 100422, 2021. doi: 10.1016/j.iot.2021.100422.

[24] Signoretti G., Silva M., Andrade P., Silva I., Sisinni E., Ferrari P.: An evolving tinyml compression algorithm for iot environments based on data eccentricity, *Sensors*, vol. 21(12), 4153, 2021. doi: 10.3390/s21124153.

[25] Sridhar A.P., Lakshmi P.V.: An efficient lossless medical data compression using lzw compression for optimal cloud data storage, *Annals of the Romanian Society for Cell Biology*, vol. 25(6), pp. 17144–17160, 2021. http://annalsofrscb.ro/index.php/journal/article/view/9004.

[26] Vakili M., Ghamsari M., Rezaei M.: Performance analysis and comparison of machine and deep learning algorithms for IoT data classification, *arXiv preprint arXiv:200109636*, 2020. doi: 10.48550/arXiv.2001.09636.

[27] Vijayalakshmi B., Sasirekha N.: Comparative Analysis of Lossless Text Compression Methods with Novel Tamil Compression Technique, *International Journal of Research in Engineering and Science (IJRES)*, vol. 9(7), pp. 38–44, 2021. https://www.ijres.org/papers/Volume-9/Issue-7/Series-13/H09073844.pdf.

Affiliations

Shatakshi Kokate

G. H. Raisoni College of Engineering Nagpur, Department of Computer Science and Engineering, India, ORCID ID: https://orcid.org/0000-0002-3257-8099

Urmila Shrawankar

Shri. Ramdeobaba College of Engineering and Management, Department of Computer Science and Engineering, Nagpur, India, ORCID ID: https://orcid.org/0000-0003-4523-9501

Received: 09.07.2024
Revised: 22.09.2024
Accepted: 19.08.2025