






## A Lightweight Blockchain to Secure Data Communication in IoT Network on Healthcare System



Janardhana Dasarigatta Rangappa<sup>1\*</sup>, A.P. Manu<sup>2</sup>, Shivanna Kariyappa<sup>3</sup>, Suhas Kamshetty Chinnababu<sup>4</sup>,  
Gururaj Harinahalli Lokesh<sup>5</sup>, Francesco Flammini<sup>6</sup>

<sup>1</sup> Department of CSE, Nitte Meenakshi Institute of Technology, Bengaluru and Affiliated to Visvesvaraya Technological University, Belagavi 560064, India

<sup>2</sup> Department of CSE, PES Institute of Technology and Management, Shivamogga and Affiliated to Visvesvaraya Technological University, Belagavi 560064, India

<sup>3</sup> Department of CSE, Sahyadri College of Engineering and Management, Mangaluru 575007, India

<sup>4</sup> Department of CSE, Channabasaveshwara Institute of Technology, Gubbi 572216, India

<sup>5</sup> Department of Information Technology, Manipal Institute of Technology Bengaluru, Manipal Academy of Higher Education, Manipal 576104, India

<sup>6</sup> IDSIA USI-SUPSI, University of Applied Sciences and Arts of Southern Switzerland, Manno 6928, Switzerland

Corresponding Author Email: [jdr5414@gmail.com](mailto:jdr5414@gmail.com)

Copyright: ©2023 IETA. This article is published by IETA and is licensed under the CC BY 4.0 license (<http://creativecommons.org/licenses/by/4.0/>).

<https://doi.org/10.18280/ijssse.130604>

### ABSTRACT

**Received:** 4 October 2023

**Revised:** 14 November 2023

**Accepted:** 29 November 2023

**Available online:** 25 December 2023

#### Keywords:

*IoT, blockchain, distributed systems, blockchain based IoT, cloud storage, healthcare*

The burgeoning domain of the Internet of Things (IoT) encompasses a myriad of interconnected devices tasked with the automated collection of sensitive data. A paramount challenge within this realm is the establishment of stringent security protocols to safeguard sensitive information and thwart unauthorized access. Although various strategies have been conceived and implemented to fortify data protection in IoT environments, the unique resource limitations intrinsic to IoT devices necessitate further exploration. The criticality of efficient time and memory management for the optimization of IoT application performance cannot be overstated. This paper elucidates the efficacy of employing lightweight blockchain technology as a bulwark to secure numerous IoT applications. It introduces a symmetric cryptographic algorithm, known as Blowfish, tailored for the secure transmission of data within IoT networks. A novel key generation phase has been developed, demonstrating an adept utilization of time and memory resources on IoT devices for the encryption of transaction data. Furthermore, these transactions are recorded within a blockchain database, capitalizing on its inherent immutability. Comparative analysis reveals that the proposed scheme surpasses contemporary algorithms, including AES and 3DES, with regard to encryption time and memory overhead for key generation. This advancement heralds a significant stride in the quest to bolster IoT security.

## 1. INTRODUCTION

In the rapidly expanding landscape of the Internet of Things (IoT), myriad applications such as industrial automation and healthcare monitoring necessitate not only immediate responsiveness to sensor data but also robust data protection mechanisms to inform critical decision-making processes. Given the resource constraints characteristic of IoT devices, particularly in terms of memory utilization, optimizing performance in IoT applications is paramount. The efficiency of memory management mechanisms is directly correlated with the devices' capacity to store and process data, underscoring the importance of developing lightweight algorithms and data structures. The roles of time and memory management in enhancing the efficiency of IoT systems are critical, influenced by factors such as rapid response capabilities, energy efficiency, and system reliability. The

seamless operation of IoT devices and their applications hinges on the effective management of these resources. To address these challenges, the previous studies proposed a lightweight encryption method predicated on blockchain technology, which is distinguished by minimal resource usage and reduced encryption overheads [1-3]. This innovative method achieves a diminutive block size and a shorter key length, utilizing straightforward key generation to mitigate computational complexity.

In healthcare IoT systems, the stakes are particularly high as devices collect and transmit sensitive patient information, including diagnostic results, medical histories, and care plans. The unauthorized access to such healthcare information can result in significant privacy breaches. Thus, the protection of confidentiality and the prevention of unauthorized access become non-negotiable priorities [4]. Within healthcare IoT applications, the integrity of data is of utmost importance; any

alteration or manipulation of health records could precipitate errors in medical care [5, 6]. Ensuring the security of the communication network utilized by healthcare IoT systems is imperative to prevent both interception and unauthorized data access [7]. Secure communication protocols and robust encryption algorithms are fundamental in safeguarding patient information [8].

Numerous strategies have been introduced to fortify security and uphold privacy within IoT applications. The unique challenges of healthcare IoT systems, encompassing data encryption and secure communication, access control and authentication, data integrity, and privacy preservation, are manifold. This paper advocates for a sturdy key generation mechanism designed to encrypt transactions from IoT devices. The transactions from each device are aggregated and integrated into the blockchain network, culminating in enhanced security for healthcare data [9, 10].

The organization of this paper is as follows: Section 1 introduces the concept of IoT and the evolution of lightweight cryptographic techniques. A comprehensive review of the existing literature on IoT security and privacy issues, with a focus on healthcare applications, is articulated in Section 2. The design considerations for a lightweight blockchain mechanism, aimed at securing data communications, are delineated in Section 3. Section 4 elucidates the experimental results, while Section 5 discusses the experimental analysis in depth. The paper concludes with Section 6, synthesizing the findings and implications of the research.

## 2. RELATED WORK

In the sphere of the Internet of Things (IoT), particularly within healthcare systems, the integration with blockchain technology has garnered significant attention for its potential to enhance secure data communication. This section surveys extant research pertinent to these thematic areas that informs the implementation of the secure data communication system addressed in this paper. In the study delineated in reference [11], a novel system architecture was proposed wherein patient signals are recorded. For authentication purposes, a steganography technique was employed, with verification processes subsequently conducted in a cloud environment. The fusion of cloud and IoT technologies was pivotal in their approach, specifically tailored for the detection of migraine diseases within the human body—an application discussed by Pace et al. [12], where the authors explored workload balancing strategies on the server side of cloud systems. Further advancements in IoT healthcare applications are presented by Griggs et al. [13], where an edge-based architecture was introduced. The sensors incorporated into this design were noted for their low power consumption, both in radio communication and data processing. The performance of the proposed technique underwent rigorous evaluation in cloud-based systems. Discussion by Ali et al. [14] was particularly centered on the IBM Hyperledger platform, examining its application within secure data communication. According to their study, the proposed system interacts with a gateway to relay necessary information to healthcare users. Within this framework, blockchain technology was utilized to secure log information, although the data was stored within a local database.

The architecture proposed by Tuli et al. [15] introduces a decentralized communication strategy for nodes within

healthcare applications, embedding hidden services into its framework. This approach leverages the confluence of fog computing and edge computing methodologies, which are integrated with the Internet of Things (IoT) to support healthcare applications underpinned by deep learning models. The adoption of Artificial Intelligence (AI)-based strategies, as reported by Farahani et al. [16], is implemented across various layers of IoT infrastructure, embedding machine learning and deep learning models directly within hardware circuits. In the study of García-Valls et al. [17], the authors contribute to the discourse through a proposition of load balancing techniques, designed to equitably distribute computational loads across fog node servers to facilitate efficient execution. Security, particularly the facets of authentication and authorization, has been identified as paramount. A lightweight blockchain method, introduced by Wang et al. [18], addresses these crucial security concerns, providing a scalable framework for identity and access management in IoT devices. The protection of sensitive user data has been a focal point of recent research, with blockchain-based architectures being posited as a solution in the studies of Hossein et al. [19] and Al Omar et al. [20]. Such architectures, noted for their lightweight nature, have been extensively explored in the literature, with a predominant focus on the performance and energy efficiency of IoT nodes. The security enhancements offered by blockchain technology have been discussed in various schemes, emphasizing its role in safeguarding against data theft in healthcare applications. Given that IoT devices are typically resource-constrained, the support for lightweight architectures at edge and fog nodes is vital. The overarching goal of integrating blockchain-based architectures within IoT networks is to erect robust defenses against the exploitation of sensitive healthcare data. Blockchain has emerged as a technology of promise, offering a fortified bastion to preserve the security of data within these networks [21].

In the realm of healthcare data security, particularly within the context of cloud storage, Alshammari et al. [22] presents the development of a novel lightweight encryption algorithm. Utilizing a modified AES algorithm enhanced by a chaotic S-box, the method involves segmenting medical images into smaller units for encryption, subsequently reassembling them into a singular encrypted image. It is recognized that the encryption process for images is inherently more time-consuming when compared to textual data due to their larger size and complexity. Addressing the critical issue of power management within IoT networks, Mahdi et al. [23] introduce an enhanced Super Chacha stream cipher algorithm tailored for IoT data encryption. The proposed algorithm boasts a formidable key space, offering  $2^{512}$  potential keys, rendering brute force attacks computationally impractical. The encryption times reported range from 20 to 31 milliseconds, contingent upon the data's complexity. Further contributions to lightweight encryption strategies are found in the study of Usman et al. [24], where a novel algorithm capable of securing IoT devices using 64-bit block-size data is proposed. The efficiency of the algorithm is demonstrated through its minimal time and memory footprint, requiring only 2 milliseconds and 22 kilobytes, respectively, for implementation in a real-time IoT setting. In contrast, the RC5 algorithm, while also employed for encryption and decryption, demands substantially more time and memory, specifically 70 milliseconds and 72 kilobytes. A comprehensive analysis of various cryptographic algorithms is undertaken in the study of

Hasan et al. [25], where time complexity, payload size, and the resources required for encryption and decryption are meticulously compared. This study elucidates that symmetric cryptographic algorithms typically require fewer resources and less computational time relative to their asymmetric counterparts. Lastly, Panahi and Bayılmış [26] examines the Camellia algorithm's performance by evaluating the end-to-end delay across different payloads. The findings indicate that while the Camellia algorithm is robust, it incurs higher usage of read-only memory (ROM) and longer processing times when juxtaposed with other algorithms within the study.

### 3. DESIGN CONSIDERATION

The design and development of lightweight blockchain technology to secure data communication in IoT networks requires various factors to ensure scalability, security, and data privacy. To design a lightweight blockchain to handle many IoT devices and transactions, in addition to lessening the burden on IoT devices, we propose a lightweight cryptographic algorithm to secure data transmission and implement a robust key management process [27, 28].

#### 3.1 Design goals

In line with the design considerations mentioned above, a particular set of design goals is listed below for developing a lightweight blockchain to secure data communication in an IoT network. These goals help with the development process and make sure that the proposed work meets the needs for security, scalability, and strong key management in IoT networks.

Here are the design goals:

- Seamless authentication, privacy preservation, secure transmission of data, robustness against attacks, and flexible deployment, along with self-maintenance, can be achieved using blockchain technology. Every transaction of the IoT devices at a given instance of time is encrypted using an improved blowfish algorithm, and all the transactions are put together into a block. This block will be added to the blockchain to protect against data privacy issues due to its immutability.
- To collect specific needs of IoT to develop a blockchain-based IoT (B-IoT) for emphasising how blockchain can impact traditional cloud-centred IoT applications in healthcare. A lightweight blockchain is implemented to handle the resource constraints of the IoT network and ensure the security of healthcare data. This method avoids the transmission of data from the IoT node to the cloud centre storage network.
- To develop a lightweight blockchain-based IoT to reduce the processing overhead using elliptic curve cryptography (ECC) and hash functions that consume the smallest possible amount of energy. In the proposed methodology, the SHA1 algorithm is used to derive a shared secret key due to its one-way functionality. This lessens the burden on IoT resource utilisation.
- Develop a blockchain mechanism to ensure data encryption and digital signatures for patients' generated data, health data, and clinical record data to create a blockchain-based smart health care system.

Implementing the blockchain technique in the IoT network by collecting all the node transactions in a block for a given instance of time ensures scalability with a secure key management process.

#### 3.2 Proposed methodology

The step-by-step approaches are part of the proposed method for putting in place a lightweight blockchain to protect data communication in an IoT network. The following points are discussed to achieve the design methodology needed to implement the proposed work:

- The IoT devices interconnected with a network generate patient, health, and clinical record data.
- The data generated using IoT devices is encrypted and digitally signed with lightweight cryptography using symmetric cryptography.
- Develop the blockchain mechanism using a Merkle hash tree.
- Data decryption and access granting using lightweight cryptography using the elliptic curve method.
- End users such as care providers, patients, people involved in medical research, people involved in medical supply chain management, insurance companies, pharmaceutical companies, and other stakeholders can decrypt and access the medical data records.

The proposed architecture shown in Figure 1 collects the medical information's from the patients and clinical records of the patients through various Internet of Medical Things (IoMT). The collected data from the devices is encrypted using a lightweight cryptographic blowfish algorithm with a key size of 128 bits. A single transaction obtained by integrating a collected batch of data at regular intervals of time was used to construct the Merkle. The constructed Merkle root at the fog node is verified. If the number of unique Merkle roots is below 50%, then the constructed Merkle root is inconsistent. So, the entire transaction of the interval time data is discarded. Otherwise, the verification of the Merkle root is successful. Consistency verification of the Merkle root is done by using Eq. (1).

$$X = \begin{cases} \frac{n+1}{N+1} & \text{if } a = 1 \\ \frac{n}{N+1} & \text{otherwise} \end{cases} \quad (1)$$

where, 'X' is the success rate of the Merkle root verification at the fog node, 'n' is the number of successful verifications computed previously, and 'N' is the overall verification computed. 'a' is the current level of verification; if the value of 'a' is 1, then it represents the successful verification; otherwise, it is set to 0. Once the Merkle root verification is successful, it is sent to the construction block to add to the blockchain. After the successful creation of the block in the blockchain, the next step is to apply the decryption and access grant to the end users to access the medical data information for further analysis. In this paper, we are performing the creation of blocks and adding them to the blockchain. All the algorithms used, from the creation of keys to blocks in the blockchain, are discussed below.

To ensure robust key management for the encryption and decryption of the patient's healthcare data, a strong key derivation process is implemented in the proposed work. Two

shared secret keys are derived from the single master key of size 256-bit, a 16-bit polynomial number, and a 5-digit prime number. The reason to include a polynomial number, a prime number, and a 256-bit master key is to generate more randomness in the shared secret key by selecting 16 bytes of data from the hash value. These secret keys are used at the fog node of the IoT network to apply an encryption algorithm to the data generated by the devices. Once the data encryption process is completed, all the transactions on the device are integrated into one transaction. This transaction will be used to generate the Merkle root. These methods are proposed to defend against the various threats in the IoT network targeting medical devices to get access, and blockchain helps in preserving the privacy of healthcare data. Level 1 and Level 2 hashes are generated to integrate all the transactions on a particular device.

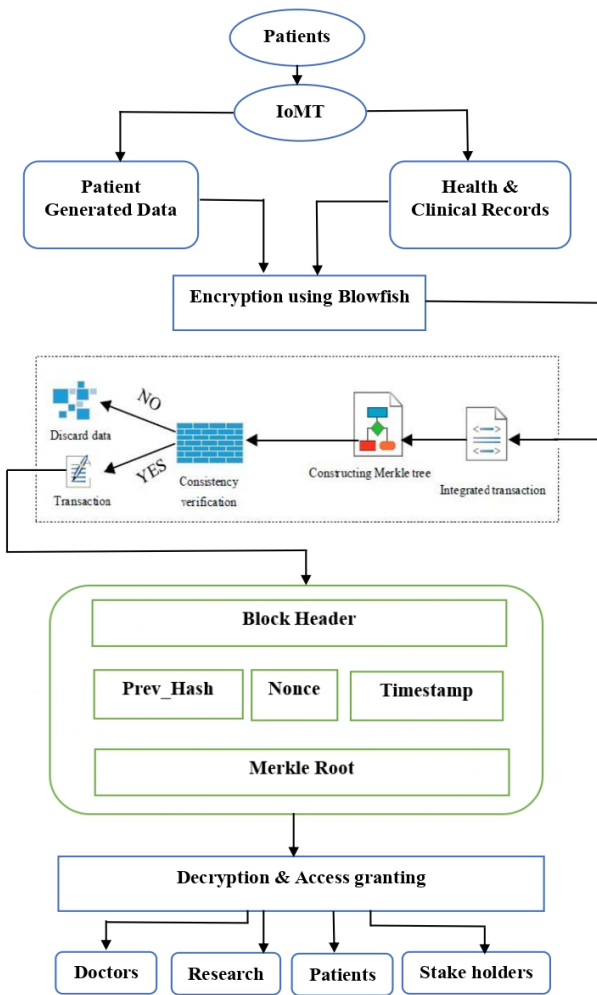


Figure 1. A lightweight blockchain architecture

### 3.3 Private key generation at fog node of IoT network

The proposed methodology is intended to achieve low-cost and lightweight cryptography while also ensuring data privacy and data origin authentication. In this context, two pairs of keys are generated, with shared key-1 derived from the polynomial<sup>1</sup> ( $f_1$ ), master secret key1 ( $MS_{k1}$ ) and prime number<sup>1</sup> ( $p_1$ ). We used the 128-bit Blowfish algorithm whenever a master was generated. During key generation, a polynomial with 16 bits and a prime number of five digits is used.

Algorithm<sup>1</sup> describes the various steps involved in the generation of shared key<sup>1</sup> ( $SH_{k1}$ ).

**Algorithm<sup>1</sup> Generate shared key<sup>1</sup> ( $SH_{k1}$ ) at Fog Node of IoT**  
**Input:**  $f_1, MS_{k1}, p_1$   
**Output:**  $SH_{k1}$

- 1: Invoke standard Blowfish algorithm such that  $MS_{k1} = \text{keygen}()$
- 2: Initialize  $MS_{k1}$  ← 256 bits
- 3: Compute hash<sup>1</sup> ←  $\text{SHA}_1(f_1 \parallel MS_{k1})$
- 4: Verify hash<sup>1</sup> is invertible  $\text{gcd}(\text{hash}^1, p_1) = 1$  and if condition fails then increment  $p_1$  and repeat step 4
- 5: Derive  $SH_{k1}$  ←  $\text{RAND}_{16}(\text{hash}^1)$  and record  $SH_{k1}$  for further processing

### 3.4 Computing shared secret key at fog node of IoT network

Algorithm<sup>2</sup> is described for generating shared key<sup>2</sup> ( $SH_{k2}$ ) at a fog computing node in an IoT network. This method enables the fog node to generate a master secret key<sup>1</sup> ( $MS_{k1}$ ) using the standard Blowfish algorithm. This master secret key<sup>1</sup> is combined with a polynomial<sup>2</sup> ( $f_2$ ) to generate hash<sup>2</sup> with 160 bits. To generate the shared key<sup>2</sup> ( $SH_{k2}$ ), use a  $\text{RAND}_{16}()$  function that generates the 128-bit output. The shared key<sup>2</sup> ( $SH_{k2}$ ) is distributed to user<sup>2</sup> for encryption and decryption operations.

**Algorithm<sup>2</sup> Generate shared key<sup>2</sup> ( $SH_{k2}$ ) at Fog Node of IoT**  
**Input:**  $f_2, MS_{k1}, p_2$   
**Output:**  $SH_{k2}$

- 1: Receives  $MS_{k1}$  from Algorithm<sup>1</sup>
- 2: Initialize  $MS_{k1}$  ← 256 bits
- 3: Compute hash<sup>2</sup> ←  $\text{SHA}_1(f_2 \parallel MS_{k1})$
- 4: Verify hash<sup>2</sup> is invertible  $\text{gcd}(\text{hash}^2, p_2) = 1$  and if condition fails then increment  $p_2$  and repeat step 4
- 5: Derive  $SH_{k2}$  ←  $\text{RAND}_{16}(\text{hash}^2)$  and record  $SH_{k2}$  for further processing

### 3.5 Level<sup>1</sup> hash code generation at fog node of IoT network

By ensuring data integrity and validation, the amount of memory required is significantly reduced. Algorithm<sup>3</sup> is created to generate hash codes for each transaction using the  $\text{SHA}_1$  algorithm. The end devices generate the same number of hash codes for further processing if they perform ten data transactions.

**Algorithm<sup>3</sup> Level<sup>1</sup> Hash code Generation at Fog Node of IoT Network**  
**Input:**  $TD^1 = TX_1, TX_2, TX_3, \dots, TX_n$   
**Output:**  $H(TX_1), H(TX_2), H(TX_3), \dots, H(TX_n)$

- 1: Device<sup>1</sup> generates the medical data transactions and such that  $TX_1, TX_2, TX_3, \dots, TX_n$
- 2: compute hash code of transactions i.e  
 $H(TX_1) \leftarrow \text{SHA}_1(TX_1)$   
 $H(TX_2) \leftarrow \text{SHA}_1(TX_2)$   
 $H(TX_3) \leftarrow \text{SHA}_1(TX_3)$   
 .....  
 $H(TX_n) \leftarrow \text{SHA}_1(TX_n)$
- 3: Record  $H(TX_1), H(TX_2), H(TX_3), \dots, H(TX_n)$  and for further processing

### 3.6 Level<sup>2</sup> hash code generation at fog node of IoT network

The Merkle tree is based on the binary tree approach, in which each node has two siblings, such as left and right nodes. In this context, Level<sup>2</sup> hash code generation takes a hash code from two different transactions as input and generates a hash code from two concatenated hashes that are already generated. Algorithm<sup>4</sup> depicts the generation of Level<sup>2</sup> hash codes at an IoT network Fog Node.

**Algorithm<sup>4</sup> Level<sup>2</sup> Hash code Generation at Fog Node of IoT Network**  
**Input:**  $H(TX_1), H(TX_2), H(TX_3) \dots H(TX_n)$   
**Output:**  $H_{12}, H_{34} \dots H_{ij}$   
 1: Receives  $H(TX_1), H(TX_2), H(TX_3) \dots H(TX_n)$  from Algorithm<sup>3</sup>  
 2: Compute hash code of transactions i.e  
 for  $i=1$  to  $n$  do  
 $H_{12} \leftarrow \text{SHA}_1(H(TX_1) \parallel H(TX_2))$   
 $H_{34} \leftarrow \text{SHA}_1(H(TX_3) \parallel H(TX_4))$   
 and ...  
 $H_{ij} \leftarrow \text{SHA}_1(H(TX_i) \parallel H(TX_j))$   
 3: Record  $H_{12}, H_{34} \dots H_{ij}$  for further processing

### 3.7 Building user<sup>1</sup> Merkle root at fog node of IoT network

Algorithm<sup>5</sup> is designed in such a way that the root node is built at the fog computing node. The root node is a collection of hash codes from all transactions involving a single-end device. In blockchain technology, the Merkle root is a basic component of the Merkle tree, and it is used as a data structure in the cryptographic system. The hashing algorithm played a significant role in creating the Merkle root. In the proposed work, each leaf in the Merkle tree performed hashing using a cryptographic hashing algorithm called SHA1 (Secure Hash Algorithm, 160-bit). The hashed leaves are then paired and then integrated into their hash values and hashed again. This process continues until there is only one hash value left, which becomes the Merkle root. Figure 2 depicts the process of creating the Merkle root using the SHA1 hashing algorithm.

**Algorithm<sup>5</sup> Construct user<sup>1</sup> Merkle root at Fog Node of IoT Network**  
**Input:** Level<sup>n</sup>  
**Output:** Merkle root<sup>1</sup>  
 1: Receives Level<sup>n</sup> output from previous algorithm. for example, receives  $H_{12}, H_{34}$  from Algorithm<sup>4</sup> if only four transactions  
 2: Compute merkle root of transactions i.e  
 Merkle root<sup>1</sup>  $\leftarrow \text{SHA}_1(\text{Level}^n)$   
 if only four transactions

3: Merkle root<sup>1</sup>  $\leftarrow \text{SHA}_1(H(TX_{12}) \parallel H(TX_{34}))$   
 Record Merkle root<sup>1</sup> for further processing

### 3.8 Generation of blockchain of user<sup>1</sup> at fog node of IoT network

Algorithm<sup>6</sup> has been designed in such a way that the entire medical IoT network can be implemented using Blockchain. Each IoT end device has been treated as a single block, with all transactions from that device hashed together to form a block. In an IoT network, the fog computing node oversees constructing each node's block. If the single transaction changes, the changes are reflected in the root hash of that transaction. To determine timing details, each block would contain the current root hash, the previous root hash, and nonce.

**Algorithm<sup>6</sup> Construction Blockchain Block<sup>1</sup> at Fog Node**  
**Input:** Merkle root<sup>1</sup>  
**Output:** Block<sup>1</sup>  
 1: Receives Merkle root<sup>1</sup> from Algorithm<sup>5</sup>  
 2: Compute blockchain of user<sup>1</sup> such that  
 Block<sup>1</sup>  $\leftarrow$  {previous-block, Merkle root<sup>1</sup>, t<sup>s</sup>, r<sup>n</sup>}  
 3: Repeat Algorithm<sup>3</sup> to Algorithm<sup>6</sup> for all users  
 4: Deploy blockchain on public IoT network

### 3.9 Implementation

The JSP web application, which is based on Java, is used to implement the suggested strategy for obtaining secret keys for encrypting the Internet of Things data using the blowfish algorithm. Amazon Web Services is one of the physical cloud environments where the method is being evaluated. The device configuration consists of 64-bit Amazon Linux 2/4.1.2 running Tomcat 8.5 with Corretto 11. Two cluster environments have been formed, and in each cluster, five nodes are deployed, and these nodes will work as two users' devices to record healthcare data. One node is dedicated to collecting the data from the cluster as a fog node or gateway. IoT node setup has been done using the IoTSim-Edge simulation tool. Each node sends data ranging in size from 1KB to 1MB during the regular interval of time. To implement the blockchain, add all the transactions of the cluster node as one transaction using the hashing algorithm and the testing carried out on the Hyperledger Fabric platform. Solo is the consensus algorithm used for validation and the creation of single blocks. Finally, in the proposed work, we are achieving the generation of a robust key for data encryption, and all the data encrypted transactions are grouped into one transaction, which is then added to the blockchain network.

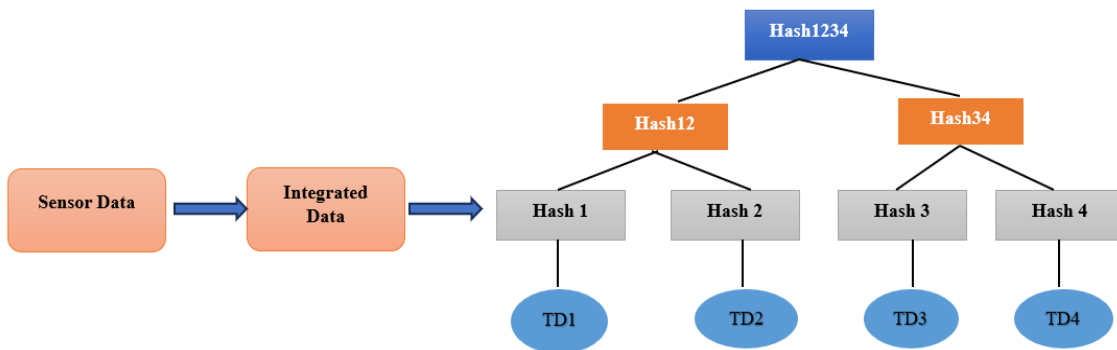


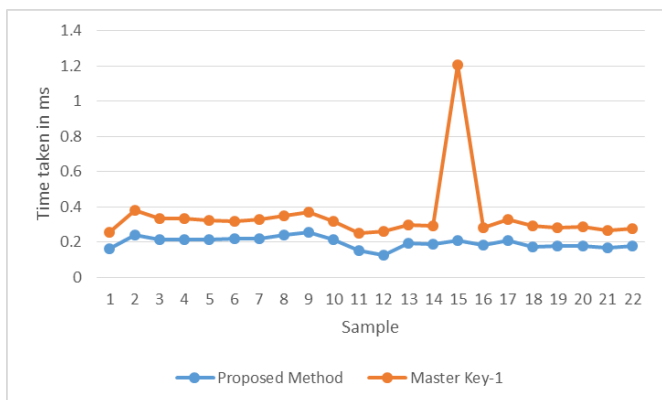
Figure 2. Construction of Merkle tree

#### 4. EXPERIMENTAL RESULTS

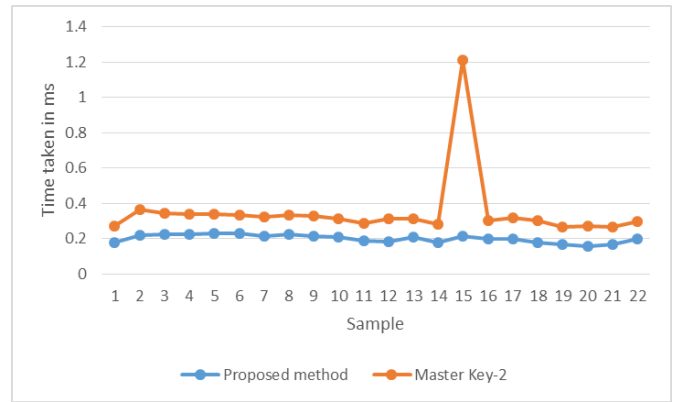
The proposed method is designed and implemented to reduce the processing overhead that causes any constrained IoT device to perform security operations. Using the block chain idea and the Merkle tree, network devices with limited memory, processing power, and bandwidth can encrypt, decrypt, and check the integrity of data. Table 1 displays the time required to derive existing master keys as well as two distinct keys derived from a single master key. The numerical values in Table 1 show that the processing time of the proposed method for generating two distinct keys is significantly less than that of an existing key generation system. This means that instead of generating multiple master keys, a single master key is enough to compute two or more derived keys. The experimental results include a comparison of the processing overhead of the existing algorithm and the proposed method, as shown in Figures 3 and 4. The graphical representation shows that the time required to derive two distinct keys (the proposed method) is significantly less than that required to derive master secret key<sup>1</sup>(MS<sub>k1</sub>) and master secret key<sup>2</sup>(MS<sub>k2</sub>).

**Table 1.** Time taken (ms) for key derivation (Existing vs. Proposed methodology)

Sample	MS <sub>k1</sub> (Blowfish)	MS <sub>k2</sub> (Blowfish)	SH <sub>k1</sub> and SH <sub>k2</sub> derived from MS <sub>k1</sub>
1	0.164	0.175	0.094
2	0.239	0.220	0.143
3	0.213	0.222	0.123
4	0.216	0.225	0.116
5	0.214	0.228	0.11
6	0.218	0.232	0.102
7	0.217	0.213	0.111
8	0.240	0.223	0.111
9	0.254	0.214	0.114
10	0.214	0.208	0.103
11	0.153	0.188	0.098
12	0.128	0.183	0.132
13	0.195	0.210	0.104
14	0.186	0.178	0.106
15	1.210	0.216	0.998
16	0.181	0.197	0.103
17	0.209	0.200	0.120
18	0.172	0.178	0.122
19	0.178	0.167	0.101
20	0.177	0.158	0.112
21	0.169	0.169	0.098
22	0.175	0.196	0.103
<b>Avg</b>	<b>0.196</b>	<b>0.200</b>	<b>0.151</b>



**Figure 3.** Time taken in (ms) MS<sub>k1</sub> vs. proposed method



**Figure 4.** Time taken in (ms) MS<sub>k2</sub> vs. proposed method

Memory utilisation for key derivation is another important parameter to consider when comparing the system's benefits. The memory consumption of key derivations is also given key weight in this research work. Table 2 displays the memory usage of existing algorithms and the proposed method in terms of bytes. Based on the values in Table 2 and Figure 5, we can conclude that the proposed method consumes less memory for deriving two different keys, which is nearly identical to the memory consumed by master secret key<sup>1</sup>(MS<sub>k1</sub>) and master secret key<sup>2</sup>(MS<sub>k2</sub>).

**Table 2.** Memory utilization in kb (Existing vs. Proposed methodology)

Sample	MS <sub>k1</sub> (Blowfish)	MS <sub>k2</sub> (Blowfish)	SH <sub>k1</sub> and SH <sub>k2</sub> derived from MS <sub>k1</sub>
1	8.78	8.79	9.05
2	8.85	8.77	8.79
3	8.82	8.84	8.80
4	8.87	8.80	8.82
5	8.81	8.82	8.83
6	9.12	8.81	8.83
7	9.09	9.05	9.06
8	8.87	8.80	8.82
9	8.86	8.83	8.83
10	8.83	9.07	8.85
11	8.81	8.83	8.85
12	8.84	8.84	9.09
13	9.07	8.84	8.81
14	8.85	8.87	8.84
15	9.08	8.85	9.05
16	8.87	8.89	8.86
17	8.84	8.85	8.82
18	8.86	8.88	8.85
19	9.08	9.09	9.06
20	8.89	8.85	8.84
21	9.09	9.10	9.07
22	8.87	8.89	8.86
<b>Avg</b>	<b>8.91</b>	<b>8.88</b>	<b>8.90</b>

The proposed method's experimental results include the amount of processing time it takes to generate a hash code from all transactions by a single user. In the current scenario, a hashing method such as SHA<sub>2</sub> is used to generate hash codes. This method generates 256-bit hash codes as an output, which increases the strength of block generation but adds processing and memory overhead. For block generation, we used the SHA<sub>1</sub> hash function in conjunction with the Merkle tree in the proposed method. This method generates 160-bit hash codes, which significantly reduces processing overhead and memory consumption. Furthermore, we use the Merkle tree to generate

blocks, with each block consisting of all transactions belonging to a single user. Tables 3 and 4 show the experimental values that were recorded. The nodes deployed in the user1 cluster environment produced the transaction data in bytes, as represented in Tables 3 and 4. These transaction data were further processed to generate a Level<sup>1</sup> hash code using the SHA1 algorithm. To produce the hash values for the transaction data, the amount of time consumed was comparatively less when compared to the SHA256 algorithm. The SHA256 algorithm consumes more computation resources on IoT devices. In Table 4, the Level<sup>2</sup> hash code is generated by concatenating the Level<sup>1</sup> hash values. This process continues until it reaches the single hash value, which is the Merkle root.

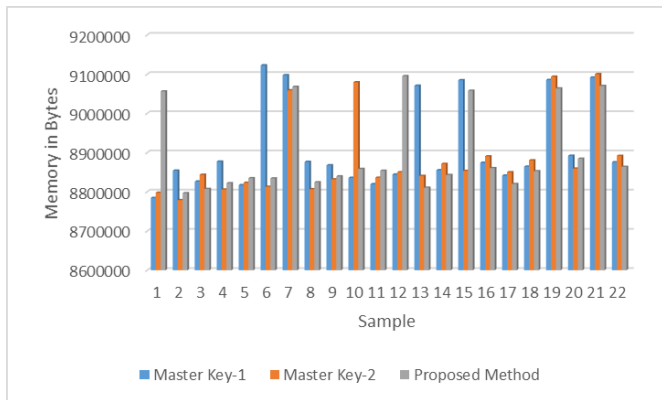


Figure 5. Memory utilization existing vs. proposed method

Table 3. Time taken in (ms) for Level<sup>1</sup> hash code with all transactions of single user

Sample	TX <sup>1</sup> (844 Bytes)	TX <sup>2</sup> (1260 Bytes)	TX <sup>3</sup> (1484 Bytes)	TX <sup>4</sup> (1807 Bytes)
1	0.071	0.072	0.071	0.068
2	0.074	0.075	0.072	0.060
3	0.073	0.090	0.071	0.071
4	0.076	0.072	0.074	0.074
5	0.074	0.070	0.072	0.072
6	0.066	0.062	0.099	0.048
7	0.082	0.070	0.080	0.082
8	0.078	0.080	0.081	0.073
9	0.067	0.068	0.074	0.071
10	0.082	0.076	0.080	0.093

Note: TX means Transactions

Table 4. Time taken in (ms) for Level<sup>2</sup> and Merkle root<sup>1</sup> hash code generation

Sample	Level <sup>2</sup>		Merkle Root <sup>1</sup>
	SHA <sub>1</sub> (H(TX <sub>1</sub> )    H(TX <sub>2</sub> ))	SHA <sub>1</sub> (H(TX <sub>3</sub> )    H(TX <sub>4</sub> ))	
1	0.067	0.059	0.057
2	0.064	0.093	0.062
3	0.064	0.067	0.064
4	0.067	0.059	0.061
5	0.056	0.055	0.052
6	0.066	0.069	0.072
7	0.087	0.061	0.065
8	0.067	0.058	0.085
9	0.070	0.060	0.067
10	0.059	0.055	0.056

We have recorded the processing time for encrypting and

decrypting all transactions belonging to the individual user with a shared key<sup>1</sup> while testing the proposed scheme on Amazon Web Services. The recorded values shown in Tables 5 and 6 show that there is no significant difference when increasing the transaction file size and that the recorded values are consistent.

Table 5. Time taken in (ms) for transaction encryption with SH<sub>k1</sub>

Sample	TX <sup>1</sup> (844 Bytes)	TX <sup>2</sup> (1260 Bytes)	TX <sup>3</sup> (1484 Bytes)	TX <sup>4</sup> (1807 Bytes)
1	0.210	0.668	0.251	0.226
2	0.183	0.627	0.190	0.193
3	0.177	0.501	0.191	0.209
4	0.171	0.560	0.234	0.201
5	0.161	0.518	0.181	0.186

Table 6. Time taken in (ms) for transaction decryption with SH<sub>k1</sub>

Sample	TX <sup>1</sup> (844 Bytes)	TX <sup>2</sup> (1260 Bytes)	TX <sup>3</sup> (1484 Bytes)	TX <sup>4</sup> (1807 Bytes)
1	0.174	0.215	0.177	0.212
2	0.153	0.163	0.172	0.178
3	0.144	0.176	0.159	0.187
4	0.167	0.175	0.179	0.159
5	0.136	0.141	0.150	0.178

The results shown in this section mainly focus on the time complexity and memory utilisation of the IoT devices to secure the data of the users in the various applications of the IoT. This work contributes more to the amount of time utilised for encrypting and decrypting the data using a robust key generation process and the amount of memory utilised for the said process. This work achieves better results without overburdening the computational resources of the devices used in the application. The results discussed in the proposed work help to implement in real-world applications like healthcare IoT devices, connected cars and transportation systems, industrial IoT (IIoT), and so on. The proposed scheme helps in protecting data privacy, and it is paramount to protect users from unauthorised access and misuse of data. Maintaining data integrity is more challenging in IoT networks. So, implementing a robust encryption process is essential, and in the proposed work, blockchain helps in maintaining data integrity. All the data generated by the devices is recorded in the blockchain, making it resistant to data tampering and providing a trustworthy record of the events.

The 128-bit shared secret keys obtained are used for conducting statistical analysis to prove the randomness of the generated key using the proposed methodology. The two tests are conducted to find the randomness in the generated key. They are the Chi-square test and the Kolmogorov-Smirnov test of normality. Both tests provide significant results regarding the randomness of the keys. This helps to ensure secure data communication in the IoT.

## 5. EXPERIMENTAL ANALYSIS

The findings of the experimental analysis in the proposed work provide valuable insight into the feasibility and practicality of integrating lightweight blockchain solutions to

enhance the security of data communication in IoT networks, ensuring that the benefits of blockchain technology align with the resource constraints essential in the IoT environment. The below-listed analysis was performed to prove the robustness of the key generation and the randomness of the key generation conducted in this work.

### 5.1 Security analysis

Because of current pandemic situations, this research work will be able to provide the best possible health care solutions because millions of people all over the world have lost their valuable lives. In this regard, this work can assist them in tracking their health-related activities so that patients can receive treatment and be cured as soon as possible. Since this project employs blockchain technology with a lightweight cryptographic algorithm, processing overhead and energy consumption are reduced, and all security-related operations can be completed under a single umbrella. Furthermore, all authenticated users can securely access health-related records, lowering costs and infrastructure.

### 5.2 Shannon entropy evaluation

In information security, the Shannon entropy is used to gauge how uncertain a random variable is. We use this method in our research to determine how many bits are in a byte. The strength of the key generated in this work rises as the number of bits increases. The Shannon entropy of current keys generated using the random number algorithm and the suggested way is displayed in Table 7. From the experimental values, we observed that the uncertainty is increased in the proposed method, which also increases the key robustness. The amount of time required to perform encryption and decryption is compared with the state-of-the-art algorithm as shown in Table 10. The proposed approach of the blowfish algorithm performs more efficiently while consuming less time and its functional comparison with other methods as shown in Table 11.

**Table 7.** Shannon entropy evaluation (Blowfish vs. Proposed methodology)

Sample	MS <sub>k1</sub>	MS <sub>k2</sub>	Proposed Method	
	(Blowfish)	(Blowfish)	Key <sub>1</sub>	Key <sub>2</sub>
1	3.149	2.005	2.774	3.500
2	2.500	1.622	2.656	3.156
3	3.274	2.647	2.953	3.156
4	2.292	2.507	2.953	3.250
5	3.374	3.287	3.000	3.203
6	2.772	2.873	3.031	3.078
7	2.005	2.250	2.875	3.031
8	2.500	3.500	3.203	3.031
9	3.149	3.875	2.899	3.572
10	2.133	2.772	2.906	2.778
<b>Avg</b>	<b>2.714</b>	<b>2.733</b>	<b>2.925</b>	<b>3.175</b>

### 5.3 Chi-square test

If the chi-square test says that the data being analysed is random, it means that the data is likely to have a random distribution, especially if it has a multinomial or categorical distribution. You can use the chi-square test to see if real data fits a theoretically expected distribution and to see if there are

any big differences between what you saw and what you thought it would be. A statistical method for finding discrepancies between observed and projected data is the Chi-Square test. This test can also be used to examine the categorical variables in the data to see if they correlate. As illustrated in Table 8, it is helpful to be able to identify an association between two categorical variables and a difference between them. Further, we successfully tested for the P-value (probability value) of Table 8, and the result is significant at  $p < 0.01$ ,  $p < 0.05$ , and  $p < 0.10$ .

**Table 8.** Randomness of the key using Chi-square test

Sample	Shared Secret Key <sup>1</sup>	Shared Secret Key <sup>2</sup>
1	1647391796	828454242
2	1701197106	926377267
3	879060326	862348390
4	859005491	942945333
5	945971556	808608097

### 5.4 Kolmogorov-Smirnov test of normality

The Kolmogorov-Smirnov test is a powerful tool for determining if two samples are significantly different from one another. Typically, it is employed to check the precision of random numbers. The uniformity of any random number generator is one of its most important properties, which may be assessed using the Kolmogorov-Smirnov test, as illustrated in Table 9. The distribution summary of Table 9 is depicted below.

Count: 10  
 Mean: 1286032377.4  
 Median: 1296806962.5  
 Standard Deviation: 413669139.318471  
 Skewness: -0.014671  
 Kurtosis: -2.5178  
 Result: The K-S test statistic (D) has a value of.32142. The probability is.20322. Your data isn't notably different from data that is typically distributed.

**Table 9.** Randomness of the key in Kolmogorov-Smirnov test

Sample	Shared Secret Key <sup>1</sup>
1	1647391796
2	1701197106
3	879060326
4	859005491
5	945971556
6	946222129
7	1684419123
8	1701078370
9	1650603313
10	845374564

### 5.5 Mathematical analysis

The proposed method has been mathematically validated so that experimental results can be widely accepted in practice. In the beginning, we demonstrated the memory consumption using Eq. (2).

$$U_m = \left( \frac{(T_m - F_m)}{T_m} \right) * 100 \quad (2)$$



**Table 10.** Performance comparison of proposed algorithm for encryption and decryption

Algorithm	Key Size (Bits)	Block Size (bits)	Time to Encrypt in ms	Time to Decrypt in ms
AES [21]	128-256	128	1-3	1-3
ChaCha [22]	256	64	2-3	2-3
RC5 [23]	32-2040	64	1-3	1-3
3DES [24]	168	64	1-2	1-2
Camellia [25]	128-256	128	1-2	1-2
<b>Proposed Method (Blowfish)</b>	<b>128-448</b>	<b>64</b>	<b>&lt;0.5</b>	<b>&lt;0.5</b>

**Table 11.** Functional comparison of proposed method vs other methods

Parameters	[1]	[20]	[10]	[7]	Proposed Method
Privacy policy	Yes	Yes	Yes	Yes	Yes
Computation cost for key derivation	High	High	High	High	Low
Storage overhead for key derivation	High	High	High	High	Low
Key robustness	Medium	Medium	Medium	Medium	High
Data integrity	Medium	Medium	Medium	Medium	High
Consistency of Blockchain	Medium	Medium	Medium	Medium	High

In Eq. (2), the utilised memory  $U_m$  is characterised by total memory  $T_m$  and free memory or available memory  $F_m$ . Further, as we have discussed in Algorithm<sup>1</sup> and Algorithm<sup>2</sup>, extracting secret keys such as  $SH_{k1}$  and  $SH_{k2}$  from the output hash code is illustrated in Eqs. (3) and (4). After getting the hash code  $H_L$  from the standard  $SHA_1$  algorithm, we must calculate how many characters are extracted from  $H_L$  and store the result in a variable called 'S'. From Eq. (3), we clearly say that the number of characters is  $S=16$ , where  $HL=40$ .

$$S = H_L / 2.5 \quad (3)$$

After calculating S, the next step is to use Eq. (4) to generate the actual secret keys. This step takes the hash code  $H_L$  as an input and retrieves a S character from  $H_L$  by looping from  $n=1$  to S using a random function. Table 9 shows a functional comparison of the proposed method with related research work.

$$S = \binom{S}{n} H_L, RANDO \quad (4)$$

## 6. CONCLUSION

Healthcare data more frequently collects sensitive information about patients. This includes health records, medication records, and sometimes real-time monitoring data. Ensuring the privacy of this information is more important to maintaining patient trust. These healthcare IoT devices are vulnerable to cybersecurity threats. Those who are employed in the healthcare sector should be aware of security measures to keep away from actions that lead to system vulnerabilities. We propose a mechanism to encrypt all the transactions of the user health records using a robust key generation system, and all the transactions are integrated into a single block by using the Merkle tree, which enhances the data security aspects and resilience to faults. The proposed work optimises the performance of the IoT devices and the block creation process. However, this work did not address the time required to read and write data in the blockchain or its optimisation techniques. In the future, the above limitations will be addressed by applying the proposed methodology to enhance the performance of the IoT network using a lightweight blockchain.

## REFERENCES

- Tripathi, G., Ahad, M.A., Paiva, S. (2020). S2HS-A blockchain based approach for smart healthcare system. *Healthcare*, 8(1): 100391. <https://doi.org/10.1016/j.hjdsi.2019.100391>
- Gul, M.J., Subramanian, B., Paul, A., Kim, J. (2021). Blockchain for public health care in smart society. *Microprocessors and Microsystems*, 80: 103524. <https://doi.org/10.1016/j.micpro.2020.103524>
- Islam, A., Shin, S.Y. (2020). A blockchain-based secure healthcare scheme with the assistance of unmanned aerial vehicle in Internet of Things. *Computers & Electrical Engineering*, 84: 106627. <https://doi.org/10.1016/j.compeleceng.2020.106627>
- Wang, J., Han, K., Alexandridis, A., Chen, Z., Zilic, Z., Pang, Y., Jeon, G., Piccialli, F. (2020). A blockchain-based eHealthcare system interoperating with WBANs. *Future Generation Computer Systems*, 110: 675-685. <https://doi.org/10.1016/j.future.2019.09.049>
- Shamshad, S., Mahmood, K., Kumari, S., Chen, C.M. (2020). A secure blockchain-based e-health records storage and sharing scheme. *Journal of Information Security and Applications*, 55: 102590. <https://doi.org/10.1016/j.jisa.2020.102590>
- Liang, Y. (2019). Identity verification and management of electronic health records with blockchain technology. In *2019 IEEE international conference on healthcare informatics (ICHI)*, Xi'an, China, pp. 1-3. <https://doi.org/10.1109/ICHI.2019.8904712>
- Pandey, P., Litoriya, R. (2020). Securing and authenticating healthcare records through blockchain technology. *Cryptologia*, 44(4): 341-356. <https://doi.org/10.1080/01611194.2019.1706060>
- Alonso, S.G., Arambarri, J., López-Coronado, M., de la Torre Díez, I. (2019). Proposing new blockchain challenges in eHealth. *Journal of Medical Systems*, 43: 1-7. <https://doi.org/10.1007/s10916-019-1195-7>
- Hasan, S.S., Sultan, N.H., Barbhuiya, F.A. (2019). Cloud data provenance using IPFS and blockchain technology. In *Proceedings of the Seventh International Workshop on Security in Cloud Computing*, pp. 5-12. <https://doi.org/10.1145/3327962.3331457>
- Dagher, G.G., Mohler, J., Milojkovic, M., Marella, P.B. (2018). Ancile: Privacy-preserving framework for access

- control and interoperability of electronic health records using blockchain technology. *Sustainable Cities and Society*, 39: 283-297. <https://doi.org/10.1016/j.scs.2018.02.014>
- [11] Hossain, M.S., Muhammad, G. (2016). Cloud-assisted industrial internet of things (IIoT)-enabled framework for health monitoring. *Computer Networks*, 101: 192-202. <https://doi.org/10.1016/j.comnet.2016.01.009>
- [12] Pace, P., Aloï, G., Gravina, R., Caliciuri, G., Fortino, G., Liotta, A. (2018). An edge-based architecture to support efficient applications for healthcare industry 4.0. *IEEE Transactions on Industrial Informatics*, 15(1): 481-489. <https://doi.org/10.1109/TII.2018.2843169>
- [13] Griggs, K.N., Ossipova, O., Kohlios, C.P., Baccarini, A.N., Howson, E.A., Hayajneh, T. (2018). Healthcare blockchain system using smart contracts for secure automated remote patient monitoring. *Journal of Medical Systems*, 42: 1-7. <https://doi.org/10.1007/s10916-018-0982-x>
- [14] Ali, M.S., Vecchio, M., Putra, G.D., Kanhere, S.S., Antonelli, F. (2020). A decentralized peer-to-peer remote health monitoring system. *Sensors*, 20(6): 1656. <https://doi.org/10.3390/s20061656>
- [15] Tuli, S., Basumatary, N., Gill, S.S., Kahani, M., Arya, R.C., Wander, G.S., Buyya, R. (2020). HealthFog: An ensemble deep learning based Smart Healthcare System for Automatic Diagnosis of Heart Diseases in integrated IoT and fog computing environments. *Future Generation Computer Systems*, 104: 187-200. <https://doi.org/10.1016/j.future.2019.10.043>
- [16] Farahani, B., Barzegari, M., Aliee, F.S., Shaik, K.A. (2020). Towards collaborative intelligent IoT eHealth: From device to fog, and cloud. *Microprocessors and Microsystems*, 72: 102938. <https://doi.org/10.1016/j.micpro.2019.102938>
- [17] García-Valls, M., Calva-Urrego, C., García-Fornes, A. (2020). Accelerating smart eHealth services execution at the fog computing infrastructure. *Future Generation Computer Systems*, 108: 882-893. <https://doi.org/10.1016/j.future.2018.07.001>
- [18] Wang, S., Li, H., Chen, J., Wang, J., Deng, Y. (2022). DAG blockchain-based lightweight authentication and authorization scheme for IoT devices. *Journal of Information Security and Applications*, 66: 103134. <https://doi.org/10.1016/j.jisa.2022.103134>
- [19] Hossein, K.M., Esmaeili, M.E., Dargahi, T., Khonsari, A., Conti, M. (2021). BCHealth: A novel blockchain-based privacy-preserving architecture for IoT healthcare applications. *Computer Communications*, 180: 31-47. <https://doi.org/10.1016/j.comcom.2021.08.011>
- [20] Al Omar, A., Bhuiyan, M.Z.A., Basu, A., Kiyomoto, S., Rahman, M.S. (2019). Privacy-friendly platform for healthcare data in cloud based on blockchain environment. *Future Generation Computer Systems*, 95: 511-521. <https://doi.org/10.1016/j.future.2018.12.044>
- [21] Priyanka, C.N., Ramachandran, N. (2023). Analysis on secured cryptography models with robust authentication and routing models in smart grid. *International Journal of Safety and Security Engineering*, 13(1): 69-79. <https://doi.org/10.18280/ijssse.130108>
- [22] Alshammari, B.M., Guesmi, R., Guesmi, T., Alsaif, H., Alzamil, A. (2021). Implementing a symmetric lightweight cryptosystem in highly constrained IoT devices by using a chaotic S-box. *Symmetry*, 13(1): 129. <https://doi.org/10.3390/sym13010129>
- [23] Mahdi, M.S., Hassan, N.F., Abdul-Majeed, G.H. (2021). An improved chacha algorithm for securing data on IoT devices. *SN Applied Sciences*, 3(4): 429. <https://doi.org/10.1007/s42452-021-04425-7>
- [24] Usman, M., Ahmed, I., Aslam, M.I., Khan, S., Shah, U.A. (2017). SIT: A lightweight encryption algorithm for secure internet of things. *arXiv preprint arXiv:1704.08688*. <https://doi.org/10.48550/arXiv.1704.08688>
- [25] Hasan, M.K., Shafiq, M., Islam, S., Pandey, B., Baker El-Ebiary, Y.A., Nafi, N.S., Rodriguez, R.C., Vargas, D.E. (2021). Lightweight cryptographic algorithms for guessing attack protection in complex internet of things applications. *Complexity*, 2021: 1-13. <https://doi.org/10.1155/2021/5540296>
- [26] Panahi, U., Bayılmış, C. (2023). Enabling secure data transmission for wireless sensor networks based IoT applications. *Ain Shams Engineering Journal*, 14(2): 101866. <https://doi.org/10.1016/j.asej.2022.101866>
- [27] Mohd, A.A., Kummarikunta, S., Naga, S.K.T., Buthukuri, V.R., Chintamaneni, P., Vatambeti, R. (2023). Design of mutual authentication method for deep learning-based hybrid cryptography to secure data in cloud computing. *International Journal of Safety and Security Engineering*, 13(5): 93-902. <https://doi.org/10.18280/ijssse.130513>
- [28] Vatambeti, R., Divya, N.S., Jalla, H.R., Gopalachari, M.V. (2022). Attack detection using a lightweight blockchain based elliptic curve digital signature algorithm in cyber systems. *International Journal of Safety and Security Engineering*, 12(6): 745-753. <https://doi.org/10.18280/ijssse.120611>

## NOMENCLATURE

MS <sub>k1</sub>	Master secret key <sup>1</sup>
MS <sub>k2</sub>	Master secret key <sup>2</sup>
SH <sub>k1</sub>	Shared key <sup>1</sup>
SH <sub>k2</sub>	Shared key <sup>2</sup>
f <sub>1</sub>	Polynomial <sup>1</sup>
f <sub>2</sub>	Polynomial <sup>2</sup>
p <sub>1</sub>	Prime number <sup>1</sup>
p <sub>2</sub>	Prime number <sup>2</sup>
Gcd	Greatest common divisor
RAND <sub>16</sub>	16-characters random extractor
hash <sup>1</sup>	Hash function output <sup>1</sup> of 160-bits
hash <sup>2</sup>	Hash function outout <sup>2</sup> of 160-bits
Keygen()	Key generator function
TD <sup>1</sup>	Transactions of device <sup>1</sup>
TX <sub>1</sub> ...TX <sub>n</sub>	User <sup>1</sup> transactions of device <sup>1</sup>
H(TX <sub>1</sub> )..H(TX <sub>n</sub> )	Hash codes of user <sup>1</sup> transactions
SHA <sub>1</sub>	Secure hash algorithm <sup>1</sup>
Mroot <sup>1</sup>	Merkle root of user <sup>1</sup>
Block <sup>1</sup>	Block of User <sup>1</sup>
r <sup>n</sup>	Nonce
t <sup>s</sup>	Time stamp
	Concatenation