FMDB *Transactions on Sustainable Critical Infrastructures*

A Novel Lightweight Cryptographic Model for Secure Communication in 5G-Enabled IoT Networks

K. Shivanna[1,*], D. R. Janardhana[2], Rajashekar Kunabeva[3], Prashanth Kambli[4], Sayyed Khawar Abbas[5]

[1]Department of Computer Science and Engineering, Dayananda Sagar Academy of Technology and Management, Bengaluru, Karnataka, India.
[2]Department of Computer Science and Engineering, Nitte Meenakshi Institute of Technology, Bengaluru, Karnataka, India.
[3]Department of Electronics and Communication Engineering, PES Institute of Technology and Management, Shivamogga, Karnataka, India.
[4]Department of Information Science and Engineering, M. S. Ramaiah Institute of Technology, Bengaluru, Karnataka, India.
[5]Department of Information Systems, Corvinus University of Budapest, Budapest, Pest County, Hungary.
shivannak4phd@gmail.com[1], jdr5414@gmail.com[2], dr.rkece2022@gmail.com[3], prash.kambli@msrit.edu[4], sayyedkhawar.abbas@uni-corvinus.hu[5]

Abstract: The Internet of Things (IoT) has experienced rapid expansion, changing how researchers use and view technology. The security and privacy of data transmitted within IoT networks have become crucial concerns as IoT devices become increasingly widespread. This study describes a method for improving the security characteristics of IoT communication protocols by leveraging a lightweight cryptographic mechanism in a 5G-enabled IoT device with limited power and memory, commonly referred to as a resource-constrained device. Traditional cryptographic algorithms designed for high-performance systems can be impractical for these devices. A lightweight cryptographic technique offers robust security while optimising resource usage. In this work, researchers investigate the uses of hash functions and light-weight block cyphers in the context of IoT communication protocols. In this work, researchers propose integrating lightweight cryptography into IoT communication protocols to address common security challenges, including data confidentiality, integrity, and authentication. Researchers also examine the impact of lightweight cryptography on key management processes, such as key generation and distribution, in resource-constrained IoT environments, and present experimental results and performance metrics demonstrating its effectiveness in enhancing the security of IoT communication protocols without imposing undue burdens on IoT devices.

Keywords: IoT Security; Communication Protocol; Lightweight Cryptography; Data Confidentiality; Data Integrity; Internet of Things (IoT); Resource-constrained Devices.

Received on: 06/09/2025, **Revised on:** 25/10/2025, **Accepted on:** 28/12/2025, **Published on:** 31/03/2026

Journal Homepage: https://www.fmdbpub.com/user/journals/details/FTSCIS

DOI: https://doi.org/10.69888/FTSCIS.2026.000713

Cite as: K. Shivanna, D. R. Janardhana, R. Kunabeva, P. Kambli, and S. K. Abbas, "A Novel Lightweight Cryptographic Model for Secure Communication in 5G-Enabled IoT Networks," *FMDB Transactions on Sustainable Critical Infrastructures*, vol. 1, no. 1, pp. 58–72, 2026.

1. Introduction

[*]Corresponding author.

The Internet of Things (IoT) concerns the connectivity of physical devices over the internet. The IoT can connect everything with surveillance [16]. As millions of devices come online in the Internet of Things, huge amounts of data are generated every moment. The majority of IoT devices satisfy resource constraints, including low power, low memory, low bandwidth, and battery- oriented design. It can be difficult to share large amounts of data within a given period [17]. Through a variety of communication protocols, including Bluetooth, Zigbee, and Wi-Fi, Internet of Things (IoT) devices connect to the cloud. These procedures, however, are susceptible to security breaches that could lead to data loss, unauthorised access, or device malfunction. Lightweight cryptographic approaches can enhance the security of IoT communication protocols. The term" lightweight cryptography" describes cryptographic methods created for devices with limited resources, such as Internet of Things (IoT) devices [18]. These methods reduce computational cost and memory requirements while preserving an appropriate level of security by using smaller key and block sizes and simpler algorithms [19]. The following are some of the techniques that can be used to enhance the security features of IoT communication protocols:

- **Symmetric Key Encryption:** To protect IoT communication protocols, symmetric key encryption techniques such as the Advanced Encryption Standard (AES) may be used. However, the key size used in these algorithms may be too large for resource- constrained devices. Lightweight cryptographic techniques such as PRESENT and SIMON can serve as alternatives to AES.
- **Hash Functions:** SHA-256 and similar hash functions can be used to verify the integrity of data sent between IoT devices. For devices with limited resources, these functions could be computationally expensive. Alternatives to BLAKE2 and Gimli are lightweight hash functions.
- **Authentication:** To authenticate IoT devices, protocols such as Transport Layer Security (TLS) can be utilised. However, these protocols may require significant computational resources, leading to high power consumption and reduced battery life. Lightweight authentication protocols such as TinyECC and HMQV can be used as alternatives.
- **Key Management:** Key management is crucial in securing IoT communication protocols. However, traditional key management protocols such as Public Key Infrastructure (PKI) may be too complex and resource-intensive for IoT devices. Lightweight key management protocols such as TESLA and LKH can be used as alternatives.

In this context, the reliability and security of IoT devices can be enhanced by strengthening IoT communication protocols through lightweight cryptographic mechanisms [20]. These methods can reduce memory requirements and computational costs while maintaining an appropriate level of security. Companies should consider implementing these strategies to secure their IoT infrastructure and safeguard sensitive data [21]. The IoT communication protocol requires addressing security-related issues, which are discussed below:

1.1. Data Confidentiality

A crucial component of security in the Internet of Things (IoT) ecosystem. IoT devices generate and collect large amounts of data, some of which may be private and sensitive. Data confidentiality, therefore, means preventing unauthorised access, use, or publication of the data. Data confidentiality in the IoT can be achieved via a variety of techniques, including encryption, authentication, and access control. Even if the data is intercepted during transit, encryption prevents unauthorised parties from accessing it. Access control restricts the types and amounts of data that various users or devices can access, while authentication guarantees that only authorised devices and users can access the data. Overall, IoT data confidentiality is essential to guaranteeing the security and privacy of sensitive data produced by IoT devices. Combining encryption, authentication, access control, and physical security measures can help achieve this.

1.2. Data Integrity

A crucial component of security in the Internet of Things (IoT) ecosystem is data confidentiality. IoT devices generate and collect large amounts of data, some of which may be private and sensitive. Data confidentiality, therefore, means preventing unauthorised access, use, or publication of the data. Data confidentiality in the IoT can be achieved via a variety of techniques, including encryption, authentication, and access control. Even if the data is intercepted during transit, encryption prevents unauthorised parties from accessing it. Access control restricts the types and amounts of data that various users or devices can access, while authentication guarantees that only authorised devices and users can access the data. Overall, IoT data confidentiality is essential to guaranteeing the security and privacy of sensitive data produced by IoT devices. Combining encryption, authentication, access control, and physical security measures can help achieve this.

1.3. Data Accuracy

The accuracy of data generated by IoT devices is crucial for making informed decisions. However, IoT devices may experience hardware or software failures, leading to incorrect data. Furthermore, some IoT devices may not have built-in quality assurance measures to ensure data accuracy.

1.4. Data Reliability

The accuracy, completeness, and consistency of data generated by Internet of Things (IoT) devices. In the context of IoT, data reliability is critical for making informed decisions and ensuring the smooth operation of IoT systems.

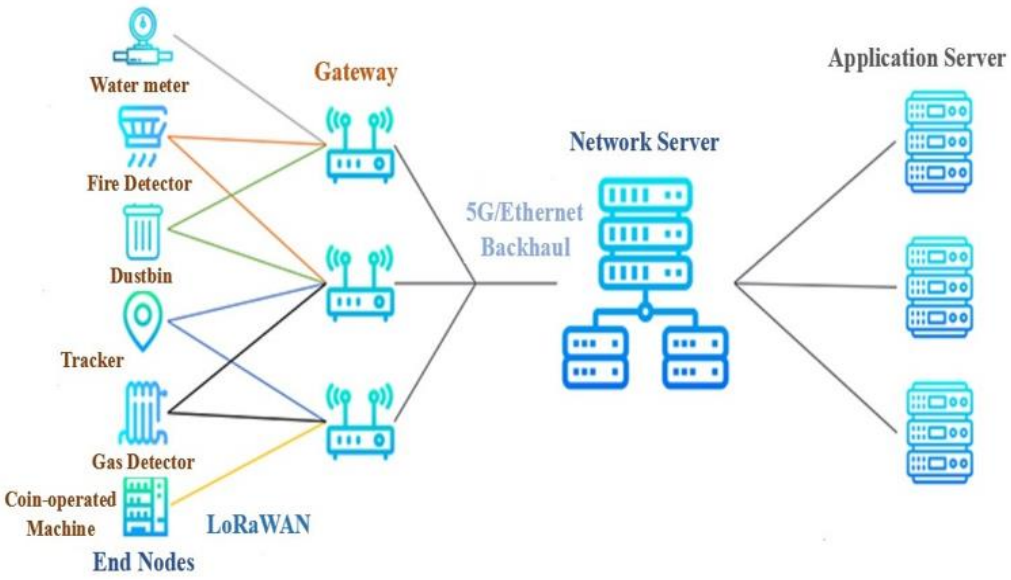

Figure 1: Architecture of the LoRaWAN communication protocol

Devices connecting to the network under resource constraints often use the Long-Range Wide Area Network (LoRaWAN) protocol. The LoRaWAN network's architecture is shown in Figure 1, which also shows how it is connected to five different edge nodes, such as a water meter, a tracker, a gas detector, a fire detector, and a coin-operated machine, through a gateway that sends data gathered by the nodes to the network server. The network server collects the aggregated data from the gateway and sends it to the application server. Additionally, the application server displays the data it has processed and received from the network server. To connect the network server and application server, Ethernet backhaul or 5G is needed. The major contribution of the proposed work is outlined below:

- Augmenting the security features of the LoRaWAN communication protocol, such as the key derivation process and the Blowfish algorithm, is used for encryption and decryption purposes.
- The pseudo-random number generator (PRNG) algorithm is used to generate the 256-bit key.
- Then a 256-bit random number is used to generate the application key, and a set of 3 128-bit random numbers.
- Each random set generated in the previous step is used to generate application session and authentication session keys.
- Data origin authentication and confidentiality verification are performed between the node and the application server.

1.5. Nomenclature

Table 1 provides the notation and its meaning in the proposed security framework. It defines the cryptographic aspects such as application keys, random number sets, random prime numbers, network and authentication session keys, and Secure Hash Algorithm (SHA). Table 1 also shows the application, network and node-generated signatures, the transaction identifiers (T1-T5) and the variables used to store the results. It also defines basic cryptographic concepts, such as message (M), ciphertext (C), encryption (E), and decryption (D), and serves as a good guide to the notation used throughout the study.

Table 1: Notations and meanings

Notations	Descriptions
$AppKey^1$	application key^1
$RanSet^1$	random number set^1
$RanSet^2$	random number set^2

$RanSet^3$	random number set^3
$RandPrm^1$	random prime $number^1$
$RandPrm^2$	random prime $number^2$
$RandPrm^3$	random prime $number^3$
$NwkSKey^1$	network session key^1
$AppSKey^1$ session	application key^1
$AuthSKey^1$	authentication session key^1
SHA_1	Secure Hash Algorithm ^{1}
$result^1$ to $result^9$	variables for storing results
$AppS_{sig}$	signature generated at the application server
NwS_{sig}	signature generated at the network server
$node_{sig}$	signature generated at node
T^1 T^5	Transactions from 1 to 5
M	message
C	ciphertext
E	encryption
D	decryption

2. Related Work

The Internet of Things (IoT) has seen a dramatic rise across fields such as sensing, healthcare, transportation, and industry, generating a large volume of data. This data travels from one place to another through an unreliable network. Enhancing the safe transmission of data requires implementing security and privacy measures, which are more crucial in today's digital era. On resource-constrained devices, conventional algorithms are not suitable for protecting user data. So, it requires more lightweight cryptographic algorithms to secure data transmission and improve data confidentiality, data integrity, authentication, authorisation, and key management. In Alassaf et al. [1], the authors discussed the SIMON block cypher method to improve its performance and make it suitable for IoT networks. AES and ECC algorithms were determined to be the best choice for IoT devices after a thorough analysis of lightweight cryptography as a solution to the security challenges of resource-constrained devices [2]. In Gunathilake et al. [3], the authors discussed the recent trends and advances around lightweight cryptographic techniques. They categorised algorithms into symmetric, asymmetric, and hash algorithms. Even though they subdivided lightweight cryptographic techniques based on the different applications into two types, they are ultra-lightweight and ubiquitous lightweight. In the current generation, high-speed data transmission networks are deployed, and devices are connecting to them smarter than ever. Lightweight cryptography will increase the robustness, the long-distance transfer of encrypted data, and the acceptable level of security [4].

AES, DES, 3DES, RSA, and Blowfish algorithms were compared based on their time complexity, size, encryption strength, and decryption speed in Gunathilake et al. [5]. For the IoT applications, they conducted a simulated experiment on a real-time deep learning network. The authors discussed in-depth research on cutting-edge, lightweight encryption techniques for Internet of Things devices in Singh et al. [6]. In various IoT scenarios, many issues arise, including device power consumption, battery life, memory usage, performance costs, and network security in the field of information and communication technology (ICT). In this paper, they go into detail on state-of-the-art lightweight cryptographic primitives, including high-performance systems, low-resource devices, and stream cyphers for IoT environments. For 5G network device-to-device communication using lightweight cryptography techniques discussed in Seok et al. [7]. To cover IoT devices with limited resources, they offer solutions utilising elliptic curve cryptography (ECC) and lightweight authenticated encryption with associated data (AEAD) cyphers. Different layers of the IoT architecture are used for sensing, and data analysis is performed in accordance with IoT security [8]. An IoT network composed of various devices, including edge nodes, fog nodes, gateways, and a cloud server. IoT communication occurs from the edge node to the cloud server, where securing data throughout the communication channel is crucial. Analysis of cryptographic schemes for Fog-to-Things communication, discussed in Diro et al. [9] and in this paper, shows that ECC will be a better replacement for embedded systems. An access control policy is to be incorporated into the IoT network over encrypted data using hybrid cryptographic techniques.

A simulation-based experiment was conducted using attribute-based encryption and lightweight cryptography. The time required for encryption and decryption was drastically reduced. To protect memory in IoT devices, hash-based techniques are combined with one-time passwords, enabling detection of malicious attacks. Designing the energy-saving technique for the constraint network is a more tedious task. In Khashan et al. [12], they develop an automatic encryption system for WSN communication that is both secure and energy efficient. Based on the resources available on the sensor nodes, this approach selects the encryption parameters. In Cirani et al. [13], application scenarios, including secure data aggregation and service authorisation, as well as security facets such as key distribution and security bootstrapping, are also covered. This study

discusses the security risks and needs associated with IoT cryptography, technology, and trends [14]. The study also compares existing IoT security solutions and the difficulties they pose. In IoT, the most widely used application-layer communication protocols are CoAP and MQTT. In Bali et al. [15], the authors identify vulnerabilities in the conventional secure MQTT protocol and propose a solution that enables security features using a lightweight authentication mechanism, implemented in the Cooja simulator. Various surveys have been conducted on resource-constrained IoT devices. In this work, they examine available algorithms with respect to implementation costs, hardware and software performance, and attack resistance, providing a comprehensive view of the field. Additionally, they discussed the need for and direction of future research in lightweight cryptography to achieve the best possible balance among price, performance, and security.

Zigbee and IPv6-based low-power WAN are the two most widely used communication technologies in the IoT space, and both include security features such as authentication, confidentiality, and integrity. The LoRa Wide Area Network (LoRaWAN) security framework uses pre-shared cryptographic keys when a key update is required. As a result, this paper assesses security flaws in LoRaWAN's key management system and proposes various alternative approaches. It is determined that the use of a technique based on the recently specified Ephemeral Diffie-Hellman Over COSE (EDHOC) is a practical solution, given its flexibility for updating session keys, its low computational cost, and the few message exchanges required. Today, blockchain technology has emerged as a promising approach to data security [22]. This study focuses on the role of developing blockchain technology in an environmentally friendly IoT ecosystem, outlines the critical elements to be considered in creating a green IoT ecosystem, and explores how blockchain technology helps the IoT ecosystem become more environmentally friendly [23]. These resource-constrained devices are not suitable for security approaches that rely primarily on encryption, as they cannot perform complex encryption and decryption quickly enough to transfer data securely in real time. In Han and Wang [24], privacy preservation and energy-efficient algorithms for securing data in cloud computing were proposed, and the use of double encryption methods can be an effective strategy to enhance the security and privacy of data stored in the cloud. Double encryption involves encrypting data twice using two different encryption techniques or keys, adding an extra layer of protection.

3. Methodology

The primary goal is to enhance the security of IoT communication protocols. This includes ensuring data confidentiality and integrity, and authentication, while minimising vulnerabilities and attack surfaces. Lightweight cryptographic techniques should be selected and optimised to minimise computational and memory overhead on resource-constrained IoT devices. This goal ensures that security can be achieved without excessive resource consumption. An effective key management system should securely generate, distribute, and manage cryptographic keys for IoT devices. Key management is crucial for maintaining the confidentiality and integrity of data. The security solution should introduce minimal additional communication overhead to conserve bandwidth, especially in scenarios with limited network resources (Figure 2).

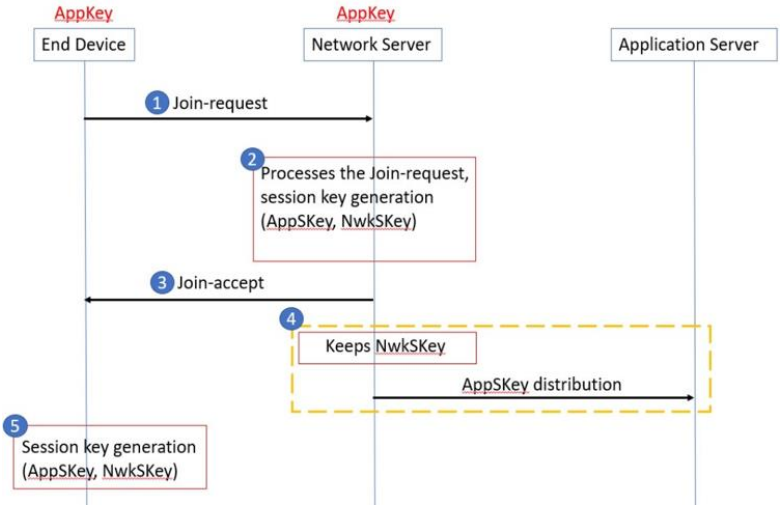

Figure 2: LoRaWAN key derivation function (KDF)

3.1. Procedure 1: Application Key (AppKey[1]) Generation

The proposed research work begins with the application of a key generation process. Application key generation in an IoT (Internet of Things) communication protocol is the process of creating cryptographic keys used to secure application-level communication between IoT devices and an application server. Figure 2 depicts the Key Derivation Function (KDF), which is

applied to the shared secret keys to derive the application key. The KDF ensures that the key is derived in a secure and deterministic manner. The secure generation and use of application keys in our IoT communication protocol is discussed in the procedure:

- The edge node generates the application key ($AppKey^1$) of size 256 bits using the standard Pseudo Random Number Generator (PRNG) algorithm.
- $AppKey^1$ is a sequence of characters from which researchers randomly select a subset.
- Apply random functions 3 times to generate three random sets, such as $RanSet^1$, $RanSet^2$, and $RanSet^3$.
- $AppKey^1$ can be used to derive a network session key, an application session key, and an authentication session key, as discussed later in this paper. The step-by-step procedure to derive $AppKey^1$ is illustrated in Algorithm 1.

Algorithm 1: Application Key ($AppKey^1$) Generation at Edge Node

Input: Key Generator Algorithm (PRNG)

Output: $AppKey^1$

1: Generate key using PRNG algorithm
2: Initialise the key with 256 bits
3: Initialise $AppKey^1$ key
4: Convert $AppKey^1$ into a string
5: Compute the string length of $AppKey^1$
6: For 1 to 3 do

$RanSet^1$ RAND($AppKey^1$) $RanSet^2$ RAND($AppKey^1$) $RanSet^3$ RAND($AppKey^1$) end for

3.2. Procedure 2: Network Session Key ($NwkSKey^1$) Generation

Generating network session keys for an IoT (Internet of Things) communication protocol is a critical step in securing data transmission between IoT devices and the network infrastructure. Network session keys are typically used to provide encryption, integrity, and authenticity for the data exchanged during a specific communication session. Ensure that the session keys are generated with sufficient entropy to resist cryptographic attacks. In the proposed work, mathematical steps for computing network session keys are described below:

- Generate a random prime number ($RandPrm^1$) with a specified size.
- Concatenating $RanSet^1$ with a random prime number ($RandPrm^1$).
- Apply the SHA_1 hash function on the concatenated result that produces a 160-bit hash code.
- Apply a random function to extract 128 bits from 160 bits, and the result is considered an $NwkSKey^1$.
- $NwkSKey^1$ is shared between the IoT device and the network server. The mathematical steps for deriving $NwkSKey^1$ are illustrated in Algorithm 2.

Algorithm 2: Network Session Key ($NwkSKey^1$) Generation at Edge/Fog Node

Input: $RandPrm^1$, $RanSet^1$

Output: $NwkSKey^1$

1: Compute $RandPrm^1$ with a specified size
2: Compute $result^1$ ($RanSet^1$||$RandPrm^1$)
3: Determine $result^2$ $SHA_1(result^1)$
4: Initialise $result^2$ with 160 bits
5: Retrieve 128-bit characters randomly from $result^2$ such that $result^3$ $random^{16}$ ($result^2$) initialize $NwkSKey^1$ $result^3$
6: Record $NwkSKey^1$ for further processing

3.3. Procedure 3: Application Session Key ($AppSKey^1$) Generation

An Application Session Key in an IoT (Internet of Things) communication protocol is a temporary cryptographic key used to protect the confidentiality, integrity, and authenticity of data exchanged between an IoT device and an Application Server

during a specific communication session. This key differs from long-term keys and is often used for a limited duration or for a specific purpose. In the proposed work, mathematical steps for computing the application session key are described below:

- Generate a random prime number ($RandPrm^2$) with a specified size.
- Concatenating $RanSet^2$ with a random prime number ($RandPrm^2$).
- Apply the SHA_1 hash function on the concatenated result that produces a 160-bit hash code.
- Apply a random function to extract 128 bits from 160 bits, and the result is considered an $AppSKey^1$.
- $AppSKey^1$ is shared between the IoT device and the application server. The mathematical steps for deriving $AppSKey^1$ are illustrated in Algorithm 3.

Algorithm 3: Application Session Key ($AppSKey^1$) Generation at Edge Node

Input: $RandPrm^2$, $RanSet^2$

Output: $AppSKey^1$

1: Compute $RandPrm^2$ with a specified size
2: Compute $result^4$ ($RanSet^2$——$RandPrm^2$)
3: Determine $result^5$ $SHA_1(result^4)$
4: Initialise $result^5$ with 160 bits
5: Retrieve 128-bit characters randomly from $result^5$ such that $result^6$ $random^{16}$ ($result^5$) initialize $AppSKey^1$ $result^6$
6: Record $AppSKey^1$ for further processing

3.4. Procedure 4: Authentication Session Key ($AuthSKey^1$) Generation

An IoT communication protocol uses authentication session keys (also known as Auth session keys) to secure the authentication process between an IoT device and a server, like an application server. These keys guarantee that the authentication exchange is genuine. When an IoT device requests authentication, such as during device onboarding or when establishing a secure connection with a server, the Auth Session Key generation process begins. Below is an illustration of the step-by-step process for calculating the Auth Session Key:

- Generate a random prime number ($RandPrm^3$) with a specified size.
- Concatenating $RanSet^3$ with random prime number.
- Apply SHA_1 hash function on concatenated result that produces 160-bits hash code.
- Apply a random function to extract 64 bits from 160 bits, and the result is considered an $AuthSKey^1$.
- $AuthSKey^1$ is shared between the IoT device and the application server, or between two IoT devices. Detailed steps for deriving $AuthSKey^1$ are described in Algorithm 4.

Algorithm 4: Application Session Key ($AuthSKey^1$) Generation at Edge Node

Input: $RandPrm^3$, $RanSet^3$

Output: $AuthSKey^1$

1: Compute $RandPrm^3$ with a specified size
2: Compute $result^7$ ($RanSet^3$——$RandPrm^3$)
3: Determine $result^8$ $SHA_1(result^7)$
4: Initialize $result^8$ with 160-bits
5: Retrieve 128-bit characters randomly from $result^8$ such that $result^9$ $random^{16}$ ($result^8$) initialize $AuthSKey^1$ $result^9$
6: Record $AuthSKey^1$ for further processing

Data origin authentication between an IoT node and a network server is a crucial security measure in IoT (Internet of Things) communication. It ensures that data transmitted from an IoT node is genuinely originating from the claimed source and has not been tampered with during transmission. Data origin authentication helps ensure the integrity and trustworthiness of data received by the network server from IoT nodes. Algorithm 5 is designed so that $NwkSKey^1$ is shared between the IoT device and the network server. The network server can verify the MAC using its own copy of the $NwkSKey^1$.

Algorithm 5: Data origin authentication betweenNode and Network Server

Input: $NwkSKey^1$, M

Output: True or False

1: Establish $NwkSKey^1$ between node and network server
2: Compute signature at node such that $node_{sig}$ HMAC($NwkSKey^1$(M))
3: Compute signature at network server such that $NwS_{sig} \leftarrow$ HMAC($NwkSKey^1$(M))
4: Network server receives $node_{sig}$ from node if $node_{sig} == NwS_{sig}$ then signature verified else something went wrong

Ensuring confidentiality between IoT devices and an application server is crucial to protect sensitive data transmitted over IoT networks from unauthorised access or eavesdropping. Confidentiality in this context refers to the ability to keep data secret and inaccessible to anyone except the intended recipients. Algorithm 6 is designed so that $AppSKey^1$ is securely distributed between the IoT node and the application server to perform encryption and decryption of shared data. The shared $AppSKey^1$ helps safeguard sensitive information from unauthorised access and maintain the privacy of users and organisations involved in IoT communication. Unauthorised devices should be denied access to sensitive resources. Minimise the amount of sensitive data transmitted between IoT devices and the application server. Only send data that is necessary for the intended purpose of reducing the risk of exposure. The suggested method is implemented using a Java-based JSP web application. The approach is being explored in a physical cloud environment, such as Amazon Web Services. The device configuration uses 64-bit Amazon Linux 2/4.3.9 and Tomcat 8.5 with Corretto 11.

Algorithm 6: Confidentiality betweenNode and Application Server

Input: $AppSKey^1$, M, C

Output: E, D

1: Establish $AppSKey^1$ between the node and application server
2: Compute ciphertext at node such that C E($AppSKey^1$(M))
3: Compute original message at application server M ← D($AppSKey^1$(C))

4. Result and Discussion

To present experimental results for the proposed key derivation scheme, researchers typically conduct empirical testing or simulations to assess its performance and security properties. Reported the performance metrics that researchers measured, such as key derivation speed (e.g., key derivations per millisecond) and memory consumption in terms of bytes. Table 2 shows the time taken (ms) for key derivation (Blowfish vs Proposed methodology). Initially, $AppKey^1$ was generated using the Blowfish algorithm, and cryptographic keys for the proposed method, such as NwkSKey1, AppSKey1, and AuthSKey1, are computed using the secure key generation scheme discussed in earlier sections of this research paper. Based on the experimental findings, the researchers conclude that the proposed scheme for secure key generation consumes less time than AppKey1 of the Blowfish algorithm. The recorded values of Table 2 are also represented graphically in Figure 3.

Table 2: Time taken (ms) for key derivation (Blowfish vs Proposed Methodology)

Sample	$AppKey^1$(Blowfish)	Proposed Method		
		$NwkSKey^1$	$AppSKey^1$	$AuthSKey^1$
1	0.328	0.124	0.125	0.131
2	0.311	0.248	0.190	0.169
3	0.243	0.207	0.212	0.136
4	0.206	0.130	0.154	0.115
5	0.324	0.178	0.107	0.115
6	0.243	0.216	0.239	0.116
7	0.224	0.124	0.128	0.114
8	0.336	0.122	0.115	0.120
9	0.236	0.121	0.146	0.111
10	0.214	0.139	0.139	0.167
11	0.226	0.120	0.109	0.117

12	0.220	0.114	0.155	0.111
13	0.220	0.117	0.127	0.115
14	0.260	0.147	0.116	0.184
15	0.227	0.106	0.108	0.121
16	0.208	0.121	0.114	0.111
17	0.285	0.104	0.108	0.151
18	0.228	0.132	0.105	0.127
19	0.210	0.101	0.124	0.128
20	0.222	0.106	0.129	0.124
21	0.210	0.158	0.113	0.110
22	0.210	0.143	0.101	0.109
Avg.	0.245	0.139	0.134	0.127

Memory consumption for cryptographic keys depends on several factors, including the cryptographic algorithm, key length, and the specific implementation of the cryptographic system. When deriving keys from other keys, memory consumption may increase as intermediate keys and related data structures are generated during the derivation. Table 3 shows memory utilization in KB (Blowfish vs proposed methodology). From the recorded values in Table 3, researchers can say that. The existing and proposed schemes for key generation may consume almost the same amount of memory by averaging the recorded values. The recorded values in Table 3 are also shown graphically in Figure 4. Several variables, such as the encryption method used, the hardware or platform on which encryption is performed, the amount of data being encrypted, and the implementation's effectiveness, can affect the time required to encrypt and decrypt data in an IoT communication protocol. In this work, approximately 5 transaction data sets were analysed, and the time (ms) required for encryption and decryption is presented in Tables 4 and 5. From the recorded values, researchers observed that computation time increases with data size.

5. Security Analysis

Security analysis of a cryptographic key's randomness is a crucial aspect of designing and evaluating cryptographic systems. The randomness of a key directly impacts the strength of the encryption or authentication scheme and its resistance to various attacks. Here are some key considerations for security analysis with respect to the randomness of the key:

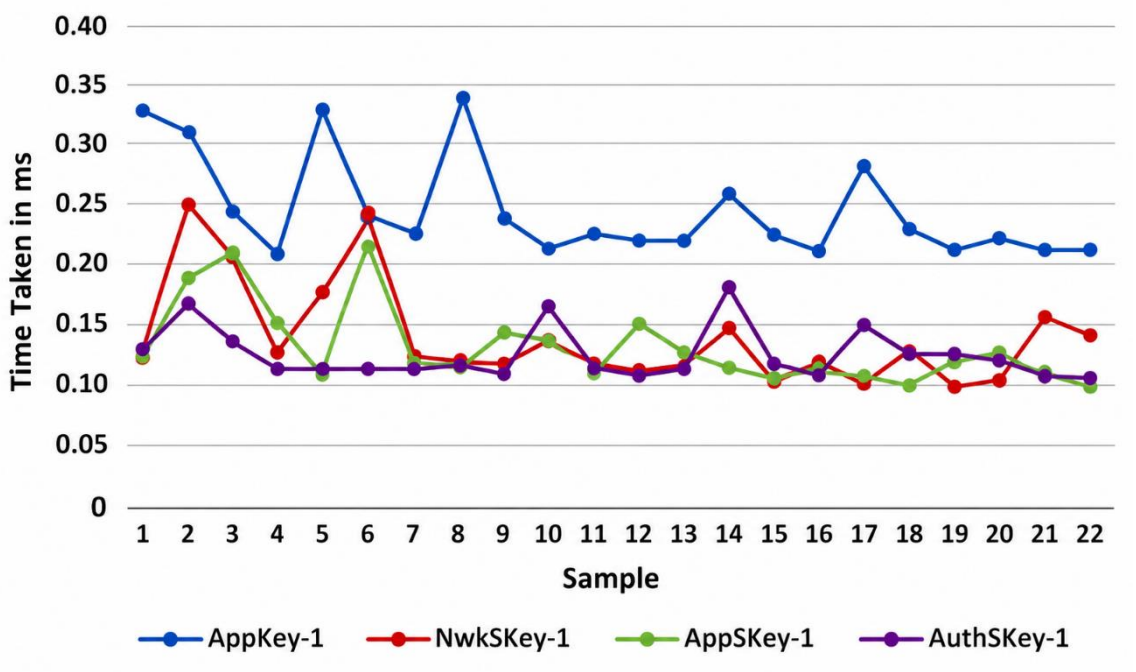

Figure 3: Time taken (ms) AppKey[1] vs proposed method

Figure 4 shows the memory usage (KB) of the four main components, namely AppKey-1, NwkSKey-1, AppSKey-1, and AuthSKey-1, for 22 different samples. The findings demonstrate that memory utilisation is rather steady, i.e., in the range of 8.1 KB to 8.6 KB, except for a few samples that show small changes. Slight peaks are observed in the authentication and

network session keys at some samples, indicating occasional increases in memory usage. In general, the graph shows that the suggested security framework maintains stable, efficient memory usage throughout the evaluation.

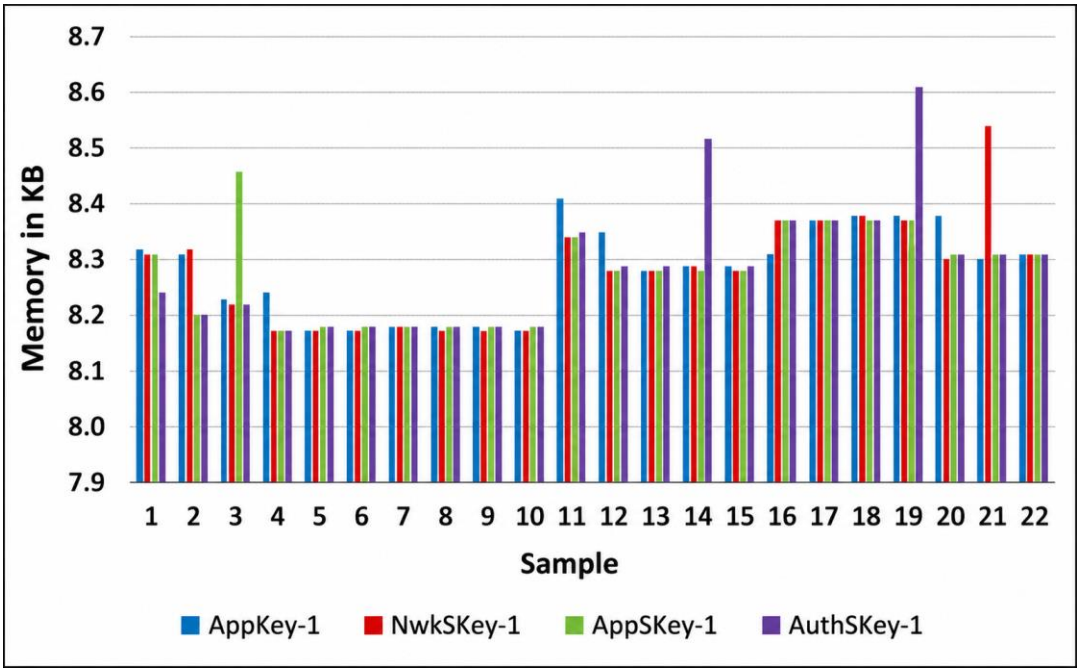

Figure 4: Memory utilisation (KB) AppKey[1] vs proposed method

5.1. Randomness of the Key in Shannon Entropy

Shannon entropy is a measure of the uncertainty or unpredictability connected to a random variable in information theory. The security of cryptographic encryption techniques frequently depends on the randomness and unpredictability of cryptographic keys. You may measure a key's randomness or uncertainty using the Shannon entropy. Entropy, or how random and unpredictable something is, should be as high as feasible for a cryptographic key.

Table 3: Memory utilisation in KB (Blowfish vs Proposed Methodology)

Sample	AppKey[1] (Blowfish)	Proposed Method		
		NwkSKey[1]	AppSKey[1]	AuthSKey[1]
1	8.32	8.31	8.31	8.24
2	8.31	8.32	8.20	8.20
3	8.23	8.22	8.46	8.22
4	8.24	8.17	8.17	8.17
5	8.17	8.17	8.18	8.18
6	8.17	8.17	8.18	8.18
7	8.18	8.18	8.18	8.18
8	8.18	8.17	8.18	8.18
9	8.18	8.17	8.18	8.18
10	8.17	8.17	8.18	8.18
11	8.41	8.34	8.34	8.35
12	8.35	8.28	8.28	8.29
13	8.28	8.28	8.28	8.29
14	8.29	8.29	8.28	8.52
15	8.29	8.28	8.28	8.29
16	8.31	8.37	8.37	8.37
17	8.37	8.37	8.37	8.37
18	8.38	8.38	8.37	8.37
19	8.38	8.37	8.37	8.61

20	8.38	8.30	8.31	8.31
21	8.30	8.54	8.31	8.31
22	8.31	8.31	8.31	8.31
Avg.	8.28	8.28	8.27	8.28

This is because a key with high entropy is safer for cryptographic applications since it is more difficult to guess or anticipate. The key's Shannon entropy measures how secure it is for encryption and other cryptographic applications. The higher the entropy of the source used to generate the key, the higher the Shannon entropy of the key itself. Table 8 compares Shannon entropy for $AppKey^1$ obtained using the Blowfish algorithm and the proposed method (Table 4).

Table 4: Time taken (ms) for encryption with $AppSKey^1$

Sample	T^1 (844 Bytes)	T^2 (1260 Bytes)	T^3 (1484 Bytes)	T^4 (1807 Bytes)	T^5 (2548 Bytes)
1	0.238	0.253	0.261	0.263	0.277
2	0.218	0.223	0.228	0.291	0.295
3	0.228	0.306	0.320	0.342	0.364
4	0.221	0.239	0.249	0.342	0.345
5	0.198	0.236	0.243	0.254	0.293

5.2. Randomness of the Key in the Chi-square Test

The Chi-Square test is an effective method for analysing frequency data and determining whether observed patterns differ significantly from expected ones (Table 5).

Table 5: Time taken (ms) for decryption with $AppSKey^1$

Sample	T^1 (844 Bytes)	T^2 (1260 Bytes)	T^3 (1484 Bytes)	T^4 (1807 Bytes)	T^5 (2548 Bytes)
1	0.191	0.202	0.296	0.331	0.593
2	0.183	0.201	0.276	0.293	0.748
3	0.137	0.197	0.202	0.294	0.671
4	0.181	0.204	0.289	0.387	0.667
5	0.186	0.268	0.273	0.285	0.782

It is commonly used for categorical variables to evaluate associations and test hypotheses about distributional differences (Table 6).

Table 6: Time taken (ms) for signature generation with $NwkSKey^1$

Sample	T^1 (844 Bytes)	T^2 (1260 Bytes)	T^3 (1484 Bytes)	T^4 (1807 Bytes)	T^5 (2548 Bytes)
1	0.118	0.126	0.154	0.166	0.179
2	0.142	0.159	0.166	0.179	0.186
3	0.165	0.172	0.184	0.189	0.195
4	0.116	0.123	0.144	0.154	0.174
5	0.125	0.134	0.201	0.234	0.253

The analysis reported in Table 9 demonstrates the usefulness of this method in identifying significant relationships between the variables (Table 7).

Table 7: Time taken (ms) for signature generation with $AuthSKey^1$

Sample	T^1 (844 Bytes)	T^2 (1260 Bytes)	T^3 (1484 Bytes)	T^4 (1807 Bytes)	T^5 (2548 Bytes)
1	0.118	0.126	0.154	0.166	0.179
2	0.142	0.159	0.166	0.179	0.186
3	0.165	0.172	0.184	0.189	0.195
4	0.116	0.123	0.144	0.154	0.174
5	0.125	0.134	0.201	0.234	0.253

The resulting p-values meet the established significance criteria at the 1%, 5%, and 10% levels, indicating strong support for the observed results.

5.3. Kolmogorov-Smirnov Test of Normality

A Kolmogorov–Smirnov analysis was conducted to investigate the distributional characteristics of the generated dataset. This method is commonly utilised to determine whether sample observations conform to a specified theoretical distribution (Table 8).

Table 8: Randomness of the key in Shannon entropy

Sample	AppKey[1] (Blowfish)	Proposed Method		
		NwkSKey[1]	AppSKey[1]	AuthSKey[1]
1	3.080	3.00	3.203	3.203
2	2.851	3.078	3.031	2.781
3	3.314	3.375	3.024	3.078
4	3.306	3.078	3.156	3.156
5	3.103	3.625	2.875	2.524
6	2.217	3.156	3.453	3.156
7	3.736	2.852	2.781	3.031
8	3.106	3.078	3.203	3.328
9	3.169	3.281	3.25	2.790
10	3.727	3.203	3.031	2.774
Avg.	3.160	3.172	3.100	2.982

In the context of random number generation, the test helps verify the uniformity of generated values, a fundamental indicator of randomness (Table 9). The results of the analysis are summarised in Table 10, and the distribution details are discussed below.

Table 9: Randomness of the key in the chi-square test

Sample	NwkSKey[1]	AppSKey[1]	AuthSKey[1]
1	1647391796	828454242	962671415
2	1701197106	909194290	926377267
3	946222129	1634022963	1647338086
4	1684419123	825569637	942946100
5	1701078370	1681143347	1684170081
6	1650603313	1684431462	875836723
7	879060326	862348390	946221624
8	859005491	942945333	895692849
9	845374564	929129574	808793957
10	945971556	808608097	862217014

5.4. Mathematical Analysis

The consumptions of memory for the proposed models are illustrated using equation (1):

$$U_m = (((T_m - F_m)/T_m) * 100) \quad (1)$$

In equation (1), the utilised memory U_m is characterised by the total memory Tm and the free or available memory F_m.

Table 10: Randomness of the key in the Kolmogorov-Smirnov test

Sample	NwkSKey[1]
1	1647391796
2	1701197106
3	946222129
4	1684419123

5	1701078370
6	1650603313
7	879060326
8	859005491
9	845374564
10	945971556

Further, as researchers have discussed from Algorithm 2 to Algorithm 4, extracted keys such as NwkSKey[1], AppSKey[1] and AuthSKey[1] from the outputted hash code are illustrated in equations (2) and (3). After obtaining the hash code HL from the standard SHA_1 algorithm, researchers must determine how many characters are extracted from HL and store the result in a variable called S. From equation (2), researchers see that S=16, where H_L=40:

$$S = H_L/2.5 \quad (2)$$

After calculating S, the next step is to use equation (3) to generate the actual secret keys. This step takes hash code H_L as input and retrieves a S character from H_L by looping from n=1 to S using a random function. A functional comparison of the proposed technique with pertinent studies is shown in Table 11:

$$S = n^S H_L * RAND\ () \quad (3)$$

Table 11: Functional comparison of the proposed method with state-of-the-art

Parameters	Diro et al. [9]	Jammula et al. [10]	Khalifa et al. [11]	Khashan et al. [12]	Proposed Method
Confidentiality	Yes	Yes	Yes	Yes	Yes
Key Derivation Cost	High	High	High	High	Low
Encryption and Decryption Time	High	High	High	High	Low
Key Strength	Medium	Medium	Medium	Medium	High
Data Integrity	Medium	Medium	Medium	Medium	High
Resilience to Attacks	Medium	Medium	Medium	Medium	High

6. Conclusion

The development of Internet of Things (IoT) devices has raised new security and privacy concerns for data exchanged in IoT networks. In this paper, the authors studied the application of simple cryptography techniques to improve the security of the IoT communication protocols. The results of our investigation are highly useful for understanding whether, and how well, lightweight cryptography works on Internet of Things devices with minimal resources. The experimental results and performance evaluations showed that lightweight cryptography can be implemented with minimal impact on latency and is acceptable for real-time IoT applications. Lightweight encryption techniques have been shown to provide strong security with the resources available. This is important for Internet of Things devices with limited memory and computing power. Furthermore, the experimental evaluation demonstrates that the proposed lightweight cryptographic method achieves a trade-off between computational efficiency and strong data safety. The implementation has been evaluated using key performance indicators, including execution time, memory utilisation, and communication overhead, demonstrating that cryptographic operations can be performed efficiently without negatively impacting overall system performance. The results show that lightweight cryptographic algorithms may provide confidentiality, integrity, and authentication while saving limited energy and processing resources in IoT contexts. Furthermore, the proposed approach improves secure communication between linked devices by reducing the risk of unauthorised access, data tampering, and cyberattacks. The observed performance characteristics indicate that the solution is a good fit for resource-constrained platforms commonly used in smart homes, healthcare, industrial automation, agriculture, and environmental monitoring applications. In general, the study emphasises that lightweight cryptography is an efficient and viable security solution for IoT systems today, enabling safe data exchange while keeping computational complexity low, memory usage minimal, and communication delays low. These features make lightweight cryptography an attractive candidate for future IoT deployments that require dependable, scalable, and energy-efficient security.

Acknowledgement: The authors sincerely acknowledge the support, resources, and academic environment provided by Dayananda Sagar Academy of Technology and Management, Nitte Meenakshi Institute of Technology, PES Institute of

Technology and Management, M. S. Ramaiah Institute of Technology, and Corvinus University of Budapest, which contributed to the successful completion of this research.

Data Availability Statement: The data supporting the findings of this study are available from the corresponding author upon reasonable request. Data containing confidential or proprietary information is not publicly available to protect participant privacy and institutional confidentiality.

Funding Statement: No funding has been obtained to help prepare this manuscript and research work.

Conflicts of Interest Statement: The authors declare that they have no competing interests.

Ethics and Consent Statement: Ethical approval was obtained from the relevant organisation before conducting this study. Informed consent was obtained from the participating organisation and all individual participants during data collection, and the study was conducted in accordance with applicable ethical guidelines and institutional requirements.

References

1. N. Alassaf, A. Gutub, S. A. Parah, and M. Al Ghamdi, "Enhancing speed of SIMON: A lightweight cryptographic algorithm for IoT applications," *Multimedia Tools and Applications*, vol. 78, no. 11, pp. 32633–32657, 2019.
2. S. S. Dhanda, B. Singh, and P. Jindal, "Lightweight cryptography: A solution to secure IoT," *Wireless Personal Communications*, vol. 112, no. 1, pp. 1947–1980, 2020.
3. N. A. Gunathilake, A. Al-Dubai, and W. J. Buchana, "Recent advances and trends in lightweight cryptography for IoT security," in *Proc. 16th Int. Conf. Network and Service Management (CNSM)*, Izmir, Turkey, 2020.
4. M. K. Hasan, M. Shafiq, S. Islam, B. Pandey, Y. A. Baker El-Ebiary, N. S. Nafi, R. Ciro Rodriguez, and D. E. Vargas, "Lightweight cryptographic algorithms for guessing attack protection in complex Internet of Things applications," *Complexity*, vol. 2021, no. 1, pp. 1–13, 2021.
5. N. A. Gunathilake, W. J. Buchanan, and R. Asif, "Next generation lightweight cryptography for smart IoT devices: Implementation, challenges and applications," in *Proc. IEEE 5th World Forum on Internet of Things (WF-IoT)*, Limerick, Ireland, 2019.
6. S. Singh, P. K. Sharma, S. Y. Moon, and J. H. Park, "Advanced lightweight encryption algorithms for IoT devices: Survey, challenges and solutions," *Journal of Ambient Intelligence and Humanized Computing*, vol. 15, no. 5, pp. 1625–1642, 2017.
7. B. Seok, J. C. S. Sicato, T. Erzhena, C. Xuan, Y. Pan, and J. H. Park, "Secure D2D communication for 5G IoT network based on lightweight cryptography," *Applied Sciences*, vol. 10, no. 1, p. 217, 2020.
8. H. Mrabet, S. Belguith, A. Alhomoud, and A. Jemai, "A survey of IoT security based on a layered architecture of sensing and data analysis," *Sensors*, vol. 20, no. 13, p. 3625, 2020.
9. A. A. Diro, N. Chilamkurti, and Y. Nam, "Analysis of lightweight encryption scheme for fog-to-things communication," *IEEE Access*, vol. 6, no. 4, pp. 26820–26830, 2018.
10. M. Jammula, V. M. Vakamulla, and S. K. Kondoju, "Hybrid lightweight cryptography with attribute-based encryption standard for secure and scalable IoT system," *Connection Science*, vol. 34, no. 1, pp. 2431–2447, 2022.
11. M. Khalifa, F. Algarni, M. A. Khan, A. Ullah, and K. Aloufi, "A lightweight cryptography (LWC) framework to secure memory heap in Internet of Things," *Alexandria Engineering Journal*, vol. 60, no. 1, pp. 1489–1497, 2021.
12. O. A. Khashan, R. Ahmad, and N. M. Khafajah, "An automated lightweight encryption scheme for secure and energy-efficient communication in wireless sensor networks," *Ad Hoc Networks*, vol. 115, no. 4, p. 102448, 2021.
13. S. Cirani, G. Ferrari, and L. Veltri, "Enforcing security mechanisms in the IP-based Internet of Things: An algorithmic overview," *Algorithms*, vol. 6, no. 2, pp. 197–226, 2013.
14. L. M. Shamala, G. Zayaraz, K. Vivekanandan, and V. Vijayalakshmi, "Lightweight cryptography algorithms for Internet of Things enabled networks: An overview," in Journal of Physics: Conference Series, *IOP Publishing*, Bristol, United Kingdom, 2021.
15. R. S. Bali, F. Jaafar, and P. Zavarasky, "Lightweight authentication for MQTT to improve the security of IoT communication," in *Proc. 3rd Int. Conf. Cryptography, Security and Privacy*, Kuala Lumpur, Malaysia, 2019.
16. V. A. Thakor, M. A. Razzaque, and M. R. A. Khandaker, "Lightweight cryptography algorithms for resource-constrained IoT devices: A review, comparison and research opportunities," *IEEE Access*, vol. 9, no. 1, pp. 28177–28193, 2021.
17. V. Rao and K. V. Prema, "A review on lightweight cryptography for Internet-of-Things based applications," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 11, pp. 8835–8857, 2021.
18. I. Bhardwaj, A. Kumar, and M. Bansal, "A review on lightweight cryptography algorithms for data security and authentication in IoTs," in *Proc. 4th Int. Conf. Signal Processing, Computing and Control (ISPCC)*, Solan, India, 2017.

19. T. M. Fernández-Caramés and P. Fraga-Lamas, "A review on the use of blockchain for the Internet of Things," *IEEE Access*, vol. 6, no. 5, pp. 32979–33001, 2018.
20. A. Shah and M. Engineer, "A survey of lightweight cryptographic algorithms for IoT-based applications," in Smart Innovations in Communication and Computational Sciences, *Springer*, Singapore, 2019.
21. R. Sanchez-Iborra, J. Sánchez-Gómez, S. Pérez, P. J. Fernández, J. Santa, J. L. Hernández-Ramos, and A. F. Skarmeta, "Enhancing LoRaWAN security through a lightweight and authenticated key management approach," *Sensors*, vol. 18, no. 6, p. 1833, 2018.
22. P. K. Sharma, N. Kumar, and J. H. Park, "Blockchain technology toward green IoT: Opportunities and challenges," *IEEE Network*, vol. 34, no. 4, pp. 263–269, 2020.
23. S. P. Jadhav, "Towards light weight cryptography schemes for resource constraint devices in IoT," *Journal of Mobile Multimedia*, vol. 15, no. 1–2, pp. 91–110, 2019.
24. J. Han and J. Wang, "An enhanced key management scheme for LoRaWAN," *Cryptography*, vol. 2, no. 4, p. 34, 2018.