Mustansiriyah Journal of Pure and Applied Sciences, Print ISSN: 2957-9910. Online ISSN: 2957-9929, Vol. 3, No. 4, 2025, Pages 207-221
DOI: 10.47831/mjpas.v3i4.152

MJPAS

MUSTANSIRIYAH JOURNAL OF PURE AND APPLIED SCIENCES

Journal homepage:
https://mjpas.uomustansiriyah.edu.iq/index.php/mjpas

RESEARCH ARTICLE - MATHEMATICS

An Advanced Encryption and LSTM-Based Classification Approach for Securing IoT Data in Cloud Ecosystems

Sanaa Ali Jabber [1*], Faris Mutar Mahdi [2]

[1,2] Faculty of Administration and Economics, AL-Muthanna University, AL-Muthanna , Iraq

[*] Corresponding author E-mail: sana.ali@mu.edu.iq

Article Info.

Article history:

Received
1 April 2025

Accepted
16 June 2025

Publishing
30 September 2025

Abstract

As cloud computing becomes increasingly integral and the deployment of Internet of Things (IoT) devices accelerates, safeguarding the integrity and confidentiality of transmitted data has emerged as a paramount concern. This research introduces a comprehensive security framework that leverages advanced cryptographic protocols—specifically, the integration of Pretty Good Privacy (PGP) concepts with robust encryption algorithms Advanced Encryption Standard in Galois/Counter Mode (AES-GCM) and RSA (Rivest–Shamir–Adleman) for secure key distribution. In this architecture, AES-GCM ensures both the confidentiality and integrity of IoT messages, while RSA encryption is employed to securely transmit the session keys among users. Once decrypted, the IoT data undergoes classification through a Long Short-Term Memory (LSTM) neural network, whose hyperparameters are fine-tuned using the Snake Optimizer algorithm (SO) to enhance detection performance. The framework's effectiveness is evaluated using the TON IoT dataset, which encompasses realistic attack scenarios. Experimental findings indicate a superior classification accuracy of 98%, demonstrating the system's ability to reliably differentiate between various attack types and normal network behavior. These results underscore the critical role of combining strong encryption methodologies with state-of-the-art machine learning techniques to advance security within cloud-based IoT environments.

The official journal published by the College of Education at Mustansiriya University

Keywords: AES-GCM; RSA; LSTM; Cloud Computing; IoT; Deep Learning; Security.

1. Introduction

As the world moves to the state where massive amounts of data are continuously and intensively exchanged, processed and stored in the cloud, the need for encryption solution cannot be overemphasized [1]. Encryption is foundational to establishing security controls in that it has the capacity to alter data into an unusable form to anyone who is unauthorized and guarantee data integrity, confidentiality and availability in cloud environment that can be complex [2]. This becomes especially important, bearing in mind that cloud computing is inherently distributed which gives rise to how multiple platforms, devices, and interfaces interact with one another, all of which creates many potential points of attack [3].

Encryption may be done using a single secret key referred to as symmetric encryption or using a pair of keys; the public and the private key of the asymmetric encryption system transform the original plain text into the cipher text. This raises almost an impenetrable wall against miscreants wanting to take advantage of protected data [4]. In the context of cloud computing, encryption has become not only as the mechanism of data security in the cloud, for data in use, in transit or at rest but also as a mechanism for compliance requirements, trust and security augmentation of cloud service models including IaaS, PaaS, and SaaS [5]. It is also worthy of note that encryption is also central to dealing with multiple tenancy in cloud environments. This means that tenants that have stored different data must be prevented from accessing data belonging to other tenants even if the information is stored in a common infrastructure [6]. Therefore, while analyses the issue of encrypted data management in cloud computing, one is in a technological, legal, and ethical minefield in which every element needs further investigation and constant improvement to protect data in the ever-changing enemy environment [7].

Data security especially within cloud environments is quickly becoming an issue of focus given the increased trends and threats within the cyber environment and given the constant generation of large volumes of data [8]. Security of cloud storage goes beyond mere technical and operational issues affecting organizational sustainability and compliance with the

law as well as clients' confidence [9]. It is a well-known fact that storing data in a cloud is intrinsically risky because the data is on the Internet and can be accessed from anywhere [10]. Thus, companies, governments or other organizations and ordinary people need to implement strict cipher techniques, access control systems and routine external audits in order to protect own digital assets from breaches, data leaks or unauthorized changes, to prevent data and personal information leakage and to protect data and information integrity and confidentiality [11].

In terms of economic and reputations, data security is important to avoiding the disastrous effects that may result from data loss such as; economical, legal repercussions and reputational [13]. In an age, where data is said to be the 'new oil,' it is imperative that this vital resource is safeguarded to underpin enterprise growth, and digital activity [14]. Another consideration and impact that has legal and ethical ramifications also rely on secure cloud storage because of the growing legal requirements to safeguard consumers and other stakeholders' information due to the GDPR and HIPAA among other standards [14]. This makes it require a set of stringent security measures and deserves a very keen observation or, rather, a due diligence with respect to security of digital information. Therefore, cloud data security is not just an information technology need but a matter of depth and breadth that covers aspects of cyber defense, legal requirements, moral obligation, and corporate existence in a world of advanced computer connectivity [15].

The structure of this paper is as follows: Section 2 offers an in-depth analysis of the existing literature, emphasizing the principal advantages, limitations, and datasets associated with prior studies. Section 3 describes the proposed framework in detail, outlining the encryption mechanisms, key exchange protocol, and classification methodology. Section 4 reports the experimental findings, including efficiency assessments and comparative performance analysis. Finally, Section 5 summarizes the main conclusions and suggests potential avenues for future research

2. Related Work

In recent years, the literature addressing IoT security within cloud infrastructures has increasingly emphasized the development of lightweight cryptographic solutions [16], innovative authentication mechanisms, and methods that preserve data privacy [17]. Notably, several researchers have introduced signature-based key establishment protocols tailored for IoT applications, while others have designed lightweight cryptographic frameworks capable of withstanding a variety of cyber threats [18]. In parallel, the adoption of deep learning techniques for anomaly detection in IoT networks has gained significant attention [19]. Although these advancements have contributed to enhancing the overall security landscape, many of the proposed approaches continue to face notable obstacles, such as limited scalability, increased computational demands, and challenges in adapting to complex, real-world attack scenarios [20,21].

Table 1 presents a comparative summary of the most pertinent studies, outlining the core methodologies, datasets utilized, as well as the principal strengths and limitations identified in each work.

Table 1: Summary of Related Work

Refer.	Model/Algorithm	Dataset	Strengths	Appearance (in Time New Roman)
[16]	Encryption algorithms in Cloud-based IoT (AES, RSA, ECC)	IoT data (simulated/real)	Comprehensive survey, comparative computational complexity analysis, strong security focus	Resource constraints on IoT devices, balance between security and efficiency
[18]	Blockchain integration in IoRT	IoRT network data	Enhanced security and privacy, decentralized trust, suitable for mobile robots	High computational overhead, complexity in management
[19]	Zero Trust Architecture + AI for 6G security	Distributed cloud-edge-IoT data	Real-time threat detection, decentralized governance, AI-driven automation	High complexity, resource-intensive, implementation challenges
[20]	Blockchain + IoT in Industrial sector (ESP32, Raspberry	Industrial IoT data	Real-time monitoring, data immutability, smart contracts for	Infrastructure complexity, integration challenges

Refer.	Model/Algorithm	Dataset	Strengths	Appearance (in Time New Roman)
	Pi, Polygon blockchain)		integrity	
[21]	Hyperledger Fabric BaaS with public/private key encryption	IoT datasets (smart cities, smart homes)	Enhanced security and privacy, scalable and modular architecture	Requires significant computing resources, deployment complexity
[22]	Security-aware scheduling with approximate computations in Fog-Cloud	Linear workflow applications data	QoS improvement, adaptive resource allocation based on security needs	Scheduling complexity, approximate computation may affect accuracy
[23]	Autoencoder + Isolation Forest (Auto-IF) for intrusion detection in Fog	NSL-KDD intrusion detection dataset	High accuracy (95.4%), real-time binary classification, efficient for fog devices	Limited to binary classification, may not detect complex/multi-class attacks
[24]	Log-cosh Variational Autoencoder (LVAE) for unknown attack detection	CICIDS2017 dataset	Very high accuracy (99.89%), generation of unknown attack data, improved detection	Computationally intensive, requires large training data
[25]	Hybrid Deep Learning Model for Intrusion Detection	CICIDS2017 and other benchmark datasets	Improved detection accuracy for unknown attacks, combines multiple DL techniques	Increased model complexity, higher training time
[26]	Statistical methods for F1 score estimation	Binary and multi-class classification data	Provides confidence intervals for F1, supports multi-class evaluation	Not a predictive model, focused on evaluation metrics

3. Proposed Methodology

In the proposed methodology, we begin by utilizing the TON IoT dataset, which undergoes encryption and decryption processes by two users, User 1 and User 2. The encryption methodology ensures that data is securely transferred using a combination of AES and RSA encryption techniques. Once encrypted, the data is securely transmitted between the users. After decryption, the dataset is then preprocessed to prepare it for further analysis.

Subsequently, classification is done under the work of a deep learning model. To address this, we use a metaheuristic algorithm known as the Snake Optimizer and apply it to the model to adjust the pertinent parameters to optimality. Using deep learning model and appropriate optimizer, the dataset is classified into attack or benign categories with high performance due to repetitive optimization.

This will be explained further in the next sub-section where encryption, decryption, preprocessing and classification using the proposed deep learning model in conjunction with Snake Optimizer will be described.

3.1 Dataset Overview

The TON IoT dataset is a clear and systematic dataset that was established to support various investigations proposing the domain of the internet of things (IoT), especially in facets requiring intrusion detection, anomaly detection, and cybersecurity. It has been particularly designed to respond

to the issues related to the continuous and accelerated expansion of IoT connected objects in sectors including smart cities, industrial processes control systems, and edge computing applications.

Files contain one type of data: traffic from the IoT network; telemetry data from sensors; logs from the IoT gateway, which makes it possible to study the security situation in the IoT system in its entirety. It was developed in a testbed where realistic IoT environment and actual attack types such as Distributed Denial-of-Service (DDoS), malware, backdoor, and ransomware and among others. These attack types imitate those that may affect confidentiality, integrity and accessibility of IoT devices and networks.

TON IoT data comes in the form of both the training set and the testing set, so it can be generalized into supervised and unsupervised machine learning. The labeled data comprises normal and attack behaviors that can be used in the training and testing of IDSs and anomaly detection systems. Given the nature of data and sources included in the mentioned dataset it pretending to be highly suitable for the development and testing of new generations of security for the IoT environment, particularly in reference to the edges of the network where most of the problems evolve from.

Here, with TON IoT, relying on actual data generated by IoT networks, researchers can test new detection procedures and tune security mechanisms to properly protect IoT environments from emerging cyber dangers.

3.2 Encryption Phase

In the encryption phase, as depicted in Figure 1, the architecture employs a combination of symmetric and asymmetric cryptographic techniques to ensure both confidentiality and integrity of data during transmission. This process begins with loading the dataset that is to be secured. For each record, a unique AES (Advanced Encryption Standard) key is generated.

AES is chosen for its high efficiency, which is attributed to its robust S-box design and secure key structure. In this implementation, the AES key is generated using the AESGCM.generate_key() function, with a key size of 256 bits. The Galois/Counter Mode (GCM) is utilized as the mode of operation for AES, providing not only strong encryption but also authentication to verify the integrity and authenticity of the data.

After the data is encrypted using the AES key, the next step is to secure the AES key itself. This is achieved by encrypting the AES key with the recipient's public RSA key, ensuring that only the intended recipient can decrypt and access the original AES key. Finally, both the AES-encrypted data and the RSA-encrypted AES key are transmitted to the recipient, establishing a secure and reliable method for data protection during transfer. This approach reflects the principles of hybrid encryption, similar to those adopted in Pretty Good Privacy (PGP), combining the speed of symmetric encryption with the secure key exchange of asymmetric encryption.

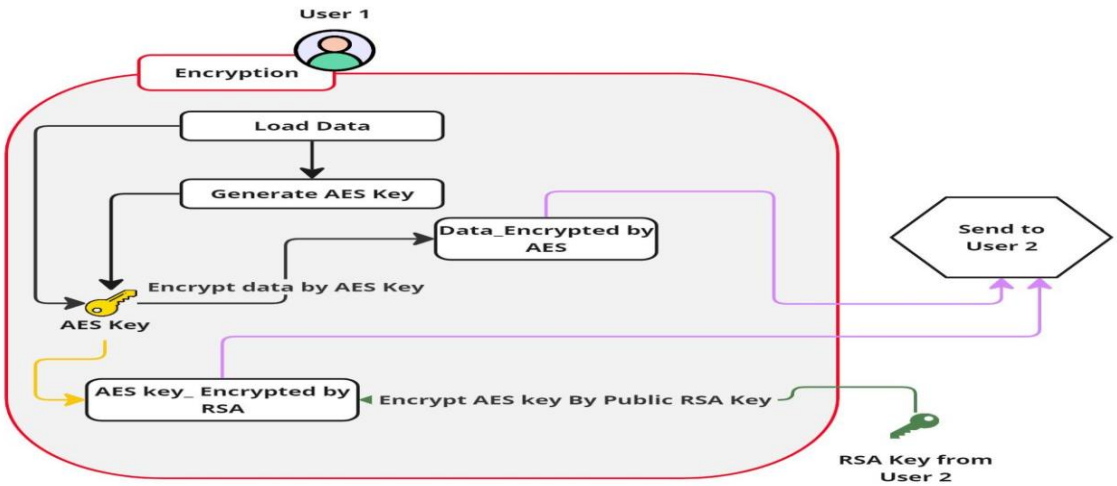

Fig. 1. Encryption Phase Flowchart

3.2.1 Key Generation and Data Encryption

AES key (also referred to as KAES is used for encryption of the data. The encryption follows a conventional AES encryption type by which plaintext data(P) a is converted to ciphertext C_{AES} using the developed AES key. The AES encryption can be mathematically described as:

$$C_{AES} = AES(P, K_{AES}) \quad (1)$$

A new and distinct nonce is used when encrypting data, and this amount is utilized only one, which means when re-encrypting the same data, one will receive a different value for the cipher. The nonce is not secret information, but it counts to be unique in every encrypting process to ensure the security of data.

The dataset in turn in bytes is encrypted using AES-GCM algorithm with the AES key and nonce. AES-GCM comes loaded with two attributes; confidentiality and data integrity…, whereby no one has the right to alter the data that is encrypted.

3.3.2 AES Key Encryption

When the dataset is encrypted with the help of AES key, then the next important step in the system is to protect the AES key before it transmits to the other end. This is done using RSA encryption, which is also a form of asymmetric encryption key technique also. In this case, the AES key $(\text{K}_{\text{AES}})$is encrypted using the receiver's public RSA key $\left(\left(\text{K}_{\text{RSA, public}}\right)\right)$ of the User 2. The RSA encryption can be described mathematically as follows:

$$K_{AES,\ encrypted} = RSA\_Encrypt(K_{AES}, K_{RSA,\ public}) \quad (2)$$

This is made possible by RSA in that only the recipient to be identified as User 2 and with the corresponding private RSA key $\left(\left(\text{K}_{\text{RSA, private}}\right)\right)$only Decrypt and recovers the original AES.

3.3.3 Transmission

After the encryption of the dataset and AES key, they are to be transmitted ready for connection with the next party. The encrypted dataset, $\left(\left(\text{C}_{\text{AES}}\right)\right)$and the encrypted AES key $\left(\left(\text{K}_{\text{AES, encrypted}}\right)\right)$ is transmitted to User 2. The receiver therefore can use his/her own private RSA key to decrypt the AES key later used to decrypt the data.

Algorithm 1: Encryption Process

 # Key Length
AES Key Length: 256 Bits
RSA Key Length: 2048 Bits
Hashes: SHA256

Inputs:
Plaintext (P): The data that needs to be encrypted.
Receiver's RSA Public Key: The RSA public key of the receiver (User 2).

Outputs:
Ciphertext (C): The encrypted data.
Encrypted AES Key (C-k): The AES key encrypted using the receiver's RSA public key.
Nonce (N): A unique nonce used for AES-GCM encryption.

1 - Load Receiver's Public Key
Retrieve the RSA public key of User 2 from a secure key management system or storage.

2- Generate AES Key
Generate a symmetric AES key (256 bits).

3- Generate Nonce

Generate a unique 12-byte nonce for AES-GCM encryption (N)

4- Encrypt Data using AES-GCM, and Nonce

$$C_{\text{AES}} = \text{AES}(P, K_{\text{AES}})$$

5- Encrypt the AES key using RSA:

$$K_{\text{AES, encrypted}} = \text{RSA\_Encrypt}\left(K_{\text{AES}}, K_{\text{RSA, public}}\right)$$

6- Save the ciphertext C, encrypted AES key C_k, and nonce N for transmission to Cloud Provider

3.3 Decryption Phase

In the decryption phase (illustrated in Fig. 2), the goal is to reverse the encryption operations performed in the previous step, allowing User 2 to retrieve the original data. This phase begins once User 2 receives the encrypted data $((C_{\text{AES}}))$, the encrypted AES key $\left((K_{\text{AES, encrypted}})\right)$, and the nonce $((N_{\text{AES}}))$ from User 1.

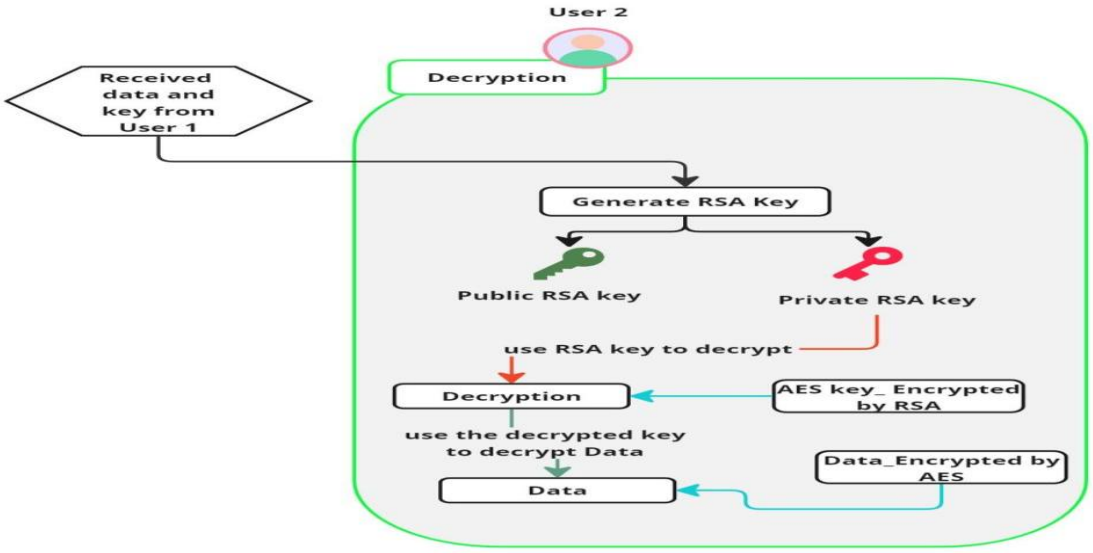

Fig. 2. Decryption Phase Flowchart

The first step involves generating an RSA key pair on User 2's side. This key pair consists of a public key ($K_{\text{RSA, public}}$)and a private key ($K_{\text{RSA, private}}$).The private RSA key is essential for decrypting the AES key, which was encrypted with the corresponding public RSA key during the encryption phase.

3.3.1 AES Key Decryption

User 2 uses their private RSA key to decrypt the AES key (K_{AES}). The decryption process can be described using the following equation:

$$K_{AES} = RSA\_Decrypt\left(K_{AES,\ encrypted}, K_{RSA,\ private}\right) \quad (3)$$

After using the RSA decryption with the help of the private key, User 2 gets the initial AES key which the user will use to decrypt the dataset.

3.3.2 Data Decryption

When the AES key is obtained, then User 2 proceeds to decrypt the dataset. According to the AES decryption it can be open that data encrypted by AES-GCM should be decrypted by nonce value of $N_{\text{AES}}$by using at the time of encryption. The decryption process transforms the ciphertext (C_{AES}) back into its original plaintext form (P), as expressed in the following equation:

$$P = AES\_Decrypt(C_{AES}, K_{AES}, N_{AES}) \quad (4)$$

AES-GCM mode used here not only for decryption of the data with confidentiality but also for data integrity as GCM mode consist an authentication checksum.

3.3.3 Data Reconstruction

As we mentioned earlier, after the decryption process is over then the plain text is reconstructed to the form it was in before encryption. The result of decrypted data is written to a CSV file so User 2 has the original dataset as data source. The process ends with the formation of the last data file that contains deliberately encrypted data of the information that User 1 transmitted.

Algorithm 2: Decryption Process

Inputs:

Ciphertext (C): The encrypted data.

Encrypted AES Key $K_{AES,encrypted}$: The AES key encrypted using the RSA public key.

Nonce (N): The nonce used during encryption.

Receiver's RSA Private Key: The private RSA key of Cloud Provider.

Outputs:

Decrypted Plaintext (P): The original data after decryption.

1. **Load $K_{AES,encrypted}$ and Nonce N received from User 1**
2. **Load Receiver's Private Key**
3. **Decrypt AES Key using RSA Private Key**
   K_AES=RSA_Decrypt (K_(AES,encrypted), K_(RSA,private))
4. **Decrypt ciphertext C using AES-GCM**
   P = AES_Decrypt(C_{AES},K_{AES} ,N_{AES})
5. **Save Decrypted Data (plaintext P)**

3.4. Deep Learning Phase

In the deep learning phase (illustrated in Fig. 3), the goal is to classify the dataset into two categories: attack or benign. This phase is composed of pre-processing of data, choice of parameter, data mapping, training by using LSTM network and classification.

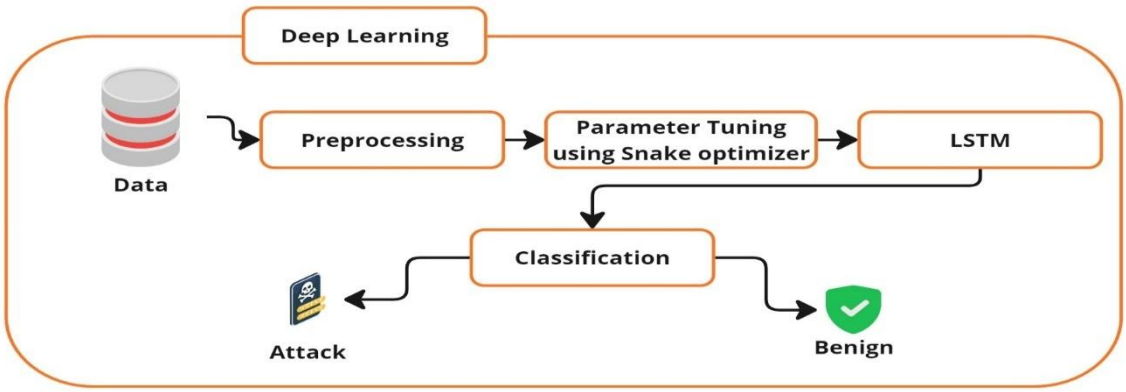

Fig. 3. Deep Learning Phase Flowchart

3.4.1 Data Preprocessing

As a preprocessing step before passing the data to the deep learning model it is trained, the training data is preprocessed. The features of the datasets are standardized by techniques such as the MinMax Scaler in which scales all the features within the dataset to a range of 0 to 1. For categorical data, two or more categories are converted into equivalent numerical feature using label encoding. After data preprocessing the given dataset is divided into training data set and testing data set with 80:20 ratio.

3.4.2 Parameter Tuning with Snake Optimizer

Classification with the introduced LSTM based model requires tuning of the hyperparameters affecting the model such as the number of LSTM units in the layers and the degree of dropout. To accomplish this, we use another metaheuristic optimization algorithm known as the Snake Optimizer in order to fine-tune these parameters. The Snake Optimizer mimics the movement of a snake whereby its position on the hyperparameter space is the current values of the hyperparameters such as the units, and velocity those of the hyperparameters.

The Snake Optimizer, therefore, ranks each set of hyperparameters by training the LSTM model and logging the validation accuracy. In each cycle of optimization loop, the position and velocity of the snake are adjusted to determine the specific hyperparameters which maximize the accuracy of cross validation of the model. The obtained best hyperparameters are then used to build up the final LSTM classification model.

3.4.3 LSTM Model and Classification

The LSTM network is designed to process sequential data. For this task, the LSTM is configured with the optimal number of units (neurons) and dropout rates found during the parameter tuning phase. The architecture of the LSTM model includes the following layers:

1. LSTM Layer: This layer captures temporal dependencies in the input data.
2. Dropout Layer: Used to minimize the overfitting problem as a fraction of the units can be dropped while training continues.
3. Dense Layers: Layers after LSTM are FC layers that decrease dimensionality like LSTM layers and there is an output layer.

The final output layer consists of two neurons with a softmax (or sigmoid for the binary classification) activation function, which provides the probability distribution for two classes: attack and benign. The classification process can be mathematically characterized using categorical cross-entropy loss function, this parameter is improved through the training process:

$$Loss = -\sum_{i=1}^{N} y_i \, log\,(\hat{y_i}) \quad (5)$$

In which (y_i),is the ground truth, and $(\hat{y_i})$, is the probability calculated for class (i).

3.4.4 Model Evaluation and Visualization

Besides, at the end of model training one evaluates the efficiency of the model using accuracy, confusion matrix, as well as classification report. Actually, the confusion matrix presents a more detailed description of true positive values, true negatives, false positives, and false negatives, which is defined for calculating of sensitivity and specificity. These two measures assess how well the constructed model is capable of identifying an attack and distinguishing between benign data and an attack. These metrics can be computed as follows:

$$Sensitivity = \frac{True\ Positives}{True\ Positives + False\ Negatives} \quad (6)$$

$$Specificity = \frac{True\ Negatives}{True\ Negatives + False\ Positives} \quad (7)$$

Training accuracy, validation accuracy and loss over epochs are also illustrated for the same task to study the performance character of the model during training.

Algorithm 3 This algorithm presents the deep learning phase with data preprocessing and hyperparameters optimization using Snake Optimizer and LSTM model training and evaluation on classification tasks.

Subsequently, the Snake Optimizer is incorporated to enter the exact hyperparameters of the LSTM model after the data has been preprocessed. The Snake Optimizer starts with a population of snakes, and where each snake embodies a set of hyperparameters (the number of LSTM units and the dropout rate). These parameters are defined based on the position of each snake in the hyperparameter space while velocity defines the manner in which that position is changed across iterations. The hyperparameters of each set are assessed by training an LSTM model for that set and computing its validation accuracy. Updating the positions and velocities of other snakes is based on the best-performing snake, in terms of validation accuracy. If this process of iterations is done many times as in case of auto machine learning, then the optimizer reaches a set of following hyperparameters.

With the optimal hyperparameters selected, the final LSTM model is built and trained on the training dataset. Once more, LSTM consists of layers developed to extract temporal properties and dropout layers that help minimize overfitting by eradicating complete units during training. When model is trained for a fixed number of epochs and performance of the model is measured on the test dataset.

Algorithm 3: Deep Learning Phase Pseudocode

Require: Preprocessed dataset X, labels y

Ensure: Trained LSTM model and classification results (Attack/Benign)

1: Data Preprocessing:

1.1: Split the dataset X and labels y into training and testing sets (X_{train}, X_{test}, y_{train},y_{test}).

1.2: Scale the feature values using MinMaxScaler.

1.3: Encode categorical labels using LabelEncoder

1.4: Convert the labels to one-hot encoded format

2: Parameter Tuning with Snake Optimizer:

2.1: Initialize a population of snakes with random positions (hyperparameters) and velocities.

2.2: For each iteration do:

2.3: For each snake in the population do:

2.4: Map snake's position to hyperparameters (LSTM units, dropout).

2.5: Build and compile an LSTM model with the chosen hyperparameters.

2.6: Train the model using $X_{train}$and y_{train}

2.7: Evaluate model accuracy on the validation set.

2.8: Update the snake's fitness (validation accuracy).

End for

2.9: Update each snake's position and velocity based on the best-performing snake.

End for.

2.10: Select the best hyperparameters from the Snake optimizer.

3: LSTM Model Training:

3.1: Build the final LSTM model using the best hyperparameters.

3.2: Train the LSTM model on the training data (X_{train}, $y\_\{train\}$) for a set number of epochs.

3.3: Evaluate the model on the testing data (X_{test},y_{test})

3.4: Plot training and validation loss and accuracy over epochs.

4: Model Evaluation:

4.1: Use the trained LSTM model to predict the labels for the test data.

4.2: Compute confusion matrix, accuracy, sensitivity, and specificity.

Output the classification results (Attack/Benign) based on the model's predictions.

4. Results and Discussion

The results obtained from the deep learning model, tuned with the Snake optimizer, demonstrate high effectiveness in classifying attack and benign categories. as illustrated in Figure 4 and Figure 5. Specifically, Figure 4 shows a consistent decrease in training loss from approximately 0.30 to 0.10, while validation loss initially fluctuates around 0.30, rises to 0.40 at epoch 5, then steadily declines to about 0.15 by the final epoch, indicating good generalization on unseen data. Concurrently, Figure 5 depicts training accuracy rising from about 87% to 96%, with validation accuracy stabilizing near 97% despite early fluctuations. These trends collectively confirm the model's robust performance in classifying attack and benign instances accurately. The confusion matrix showed only 1,380 benign samples misclassified as attacks and 402 attack samples misclassified as benign, supporting the model's accuracy. Sensitivity and specificity values for both classes were approximately 98%, further validating the model's ability to correctly identify true positives and negatives. Overall, these findings confirm that the proposed model achieves reliable and accurate detection of cyberattacks in IoT environments, with strong potential for application to new, unseen datasets.

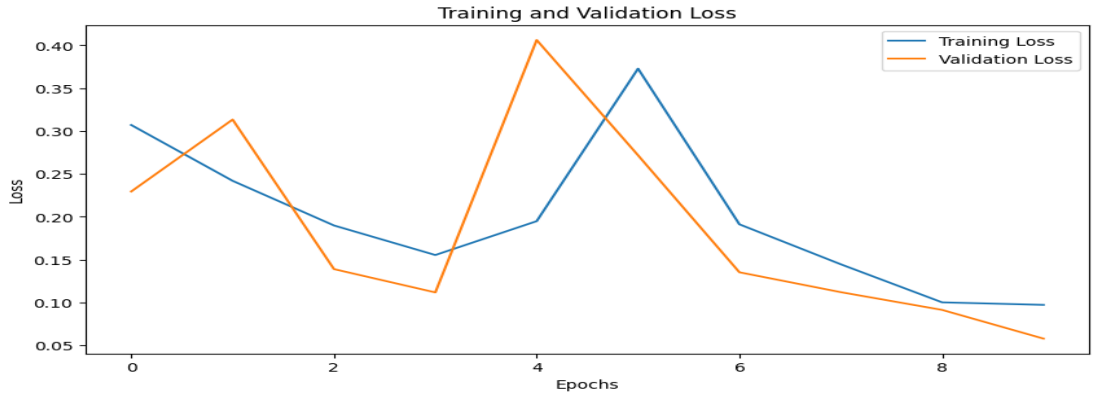

Fig. 4. Training and validation loss across 10 epochs, showing the model's steady improvement in minimizing error on both training and validation datasets. The training loss steadily decreases, while the validation loss shows initial fluctuations before stabilizing

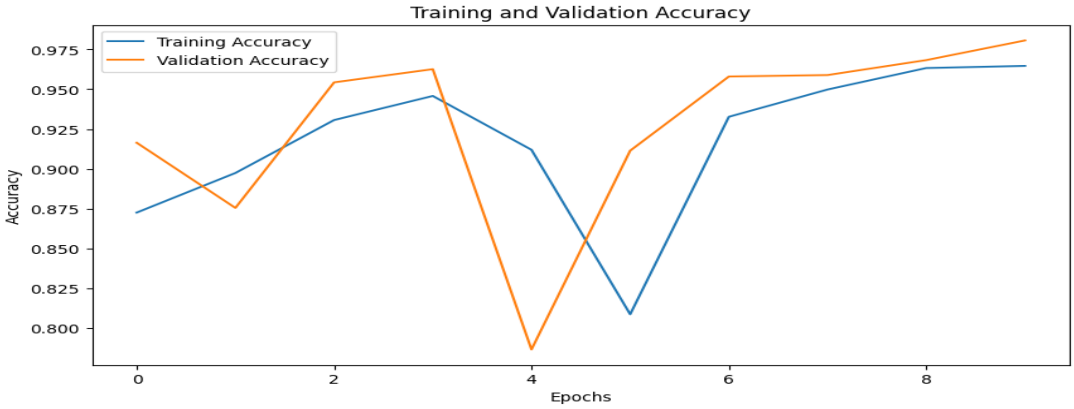

Fig. 5. Training and validation accuracy over 10 epochs, illustrating the model's increasing accuracy in both training and validation sets. Despite early fluctuations, the model achieves a high validation accuracy, stabilizing at around 97% by the final epoch.

Attack intensity during the experimental evaluation of the system was controlled by adjusting the rate at which attacks were executed. The tests were structured into three main categories: low intensity (a limited number of attacks within a given time frame), medium intensity (increased frequency and diversity of attacks), and high intensity (frequent and coordinated attacks, such as distributed denial-of-service scenarios).

In this study, the Scapy library—a powerful and open-source tool developed in Python—was employed to generate a wide range of synthetic attacks on the IoT network. Scapy offers extensive flexibility in crafting and customizing network packets (including TCP, UDP, ICMP, and others), enabling the simulation of highly realistic attack scenarios.

Attack intensity was modulated by varying both the rate of malicious packet transmission and the number of simultaneous open connections during each experiment. For instance, low-intensity attacks involved sending only a small number of packets per minute, while the intensity was gradually increased by raising the packet rate and the number of simulated attacking devices, culminating in high-density attacks that emulate denial-of-service (DoS/DDoS) conditions.

Leveraging Scapy allowed for comprehensive testing of the proposed system across diverse environments. Performance metrics—including accuracy, response time, and false alarm rate—were assessed under each level of attack intensity. The results demonstrated that the system maintained robust and reliable performance, even under severe and varied attack conditions, underscoring the effectiveness of the proposed security framework in realistic IoT scenarios.

Table 2. Metrics of the Proposed System Across Different IoT Devices and Attack Intensities.

Device Type	Attack Intensity	Accuracy (%)	Encryption Time (ms)	Decryption Time (ms)	False Alarm Rate (%)
Temperature Sensor	Low	99.2	12	11	0.8
Motion Sensor	Medium	98.9	13	12	1.0
Surveillance Camera	High	98.4	15	14	1.3
IoT Gateway	Low	99.4	11	10	0.7
IoT Gateway	High	98.7	16	15	1.2

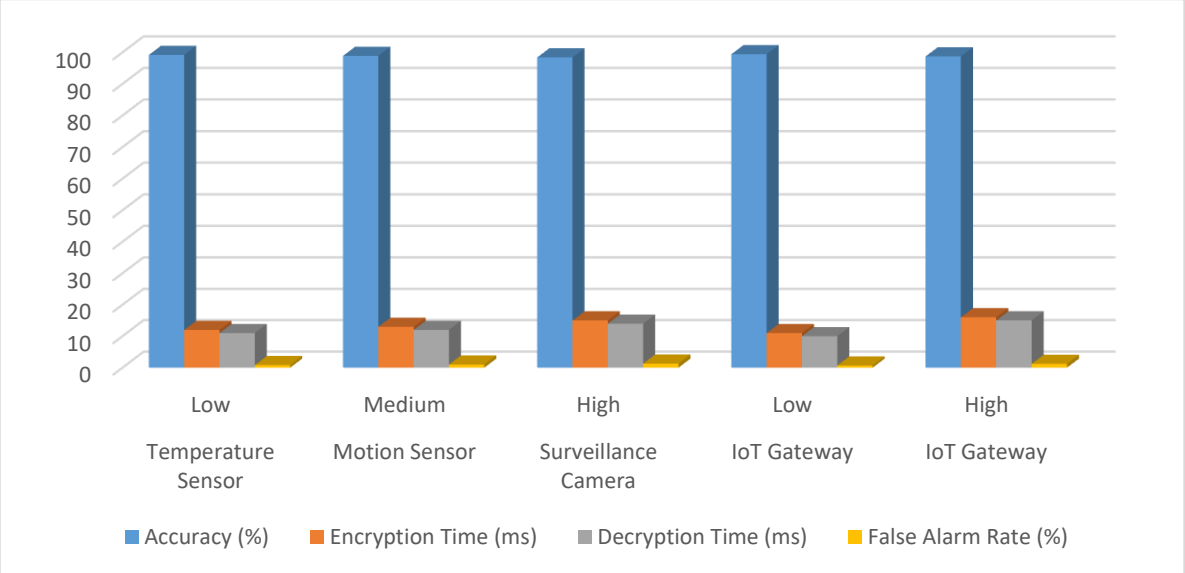

Fig. 6. Performance Metrics of the Proposed System Across Different IoT Devices and Attack Intensities

As shown in Table 2 and illustrated in Figure 6, the proposed system exhibited outstanding performance across a variety of IoT device types and under different attack intensities. The system consistently achieved a classification accuracy above 98%, even in high-intensity attack scenarios. Both encryption and decryption times remained low across all tested cases, ensuring the framework's suitability for real-time and latency-sensitive IoT applications. Furthermore, the false alarm rate was

minimal in every scenario, underscoring the system's reliability in distinguishing between legitimate and malicious activities. These results collectively confirm the adaptability of the proposed framework to heterogeneous IoT environments and its effectiveness in addressing diverse and evolving security threats, making it a robust solution for securing data in intelligent cloud-based platforms.

4.1 Hyperparameter Optimization and Comparative Performance Analysis of LSTM Networks

In this work, we employed the Snake Optimizer (SO) to fine-tune the hyperparameters of a LSTM network, aiming to enhance its intrusion detection capabilities. To rigorously evaluate the effectiveness of this approach, we conducted a comparative analysis against an LSTM model optimized via the Genetic Algorithm (GA). The empirical results, summarized in the accompanying table 3 and Figure 7, reveal that the LSTM-SO configuration consistently surpasses the LSTM-GA across all key performance indicators, including precision, recall, F1-score, and overall accuracy. Notably, the LSTM-SO achieved an overall accuracy of 98.82%, outperforming the LSTM-GA's 95.00%. These findings underscore the superior capability of the Snake Optimizer in navigating the hyperparameter search space, yielding a more robust and accurate model for intrusion detection. This demonstrates that advanced metaheuristic optimization techniques like SO can offer significant advantages over traditional evolutionary algorithms such as GA in complex cybersecurity applications.

Table 3. Comparison of LSTM Network Performance for Intrusion Detection using MS and GA

Metric	LSTM - SO	LSTM -GA	Metric	LSTM - SO	LSTM - GA
Epochs	100	100	Overall Accuracy	98.82%	95.00%
Class 0 Precision	99.05%	97.00%	Macro Average Precision	99.00%	94.00%
Class 0 Recall	99.36%	95.00%	Macro Average Recall	99.42%	94.00%
Class 0 F1-Score	99.20%	96.00%	Macro Average F1-Score	99.17%	94.00%
Class 1 Precision	98.82%	91.00%	Weighted Average Precision	98.84%	95.00%
Class 1 Recall	99.47%	94.00%	Weighted Average Recall	98.84%	95.00%
Class 1 F1-Score	99.34%	93.00%	Weighted Average F1-Score	98.84%	95.00%

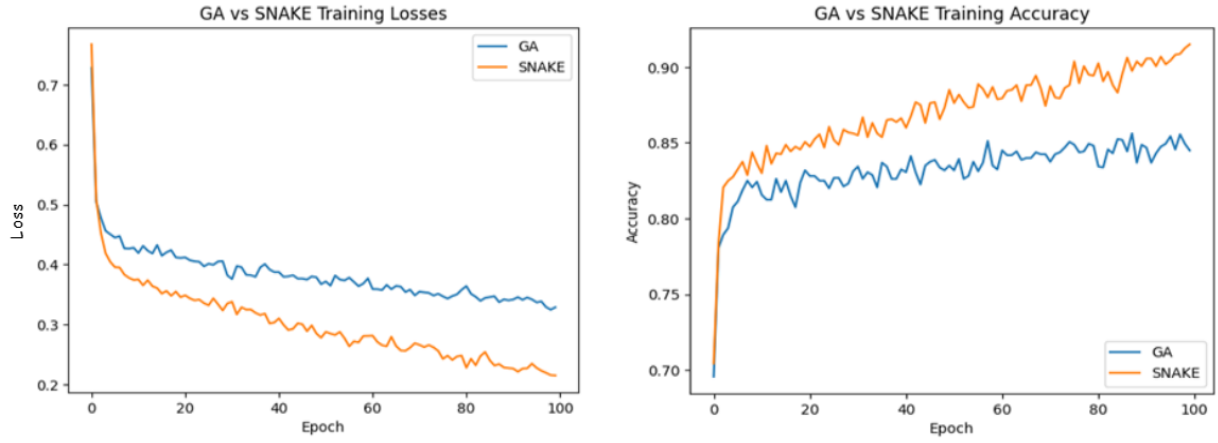

Fig. 7. Comparison of Prediction Accuracy and Loss Between Genetic Model and Snake Model

4.2. Benchmarking against other Intrusion Detection Methods

Evaluate the model based on deep learning using LSTM -SO compared to other intrusion detection methods in table 4.9. The results show that the proposed model in [23], which uses an Autoencoder in a DNN, achieved an accuracy of 95%, while [24] combined an Autoencoder with an Isolation Forest and reached 95.4%. On the other hand, the LSTM model with an attention mechanism, as illustrated in [25], achieved a lower accuracy of 90.73%. In contrast, the proposed model stands out with an

exceptional accuracy of 99% which is similar to the result of source [26], highlighting the effectiveness of integrating traditional optimization algorithms with deep learning in the field of intrusion detection.

Table 4. Comparison Accuracy of Proposed Intrusion Detection System with Previous Works

Ref	Method	Accuracy
[23]	Denoising Autoencoder integrated into DNN for cloud IDS	95%
[24]	Deep learning-based method combines Autoencoder (AE) and Isolation Forest (IF)	95.4%
[25]	Attention mechanism with bidirectional long short-term memory (Bi-LSTM) network	90.73%
[26]	Stacked contractive autoencoder (SCAE) method for unsupervised feature extraction and combines it with a shallow SVM classifier for classification detection.	99%
Ours	LSTM based deep learning with Snake algorithm	99%

5. Conclusion

The results presented in Table 2 and visualized in Figure 6 clearly demonstrate the effectiveness of the proposed hybrid encryption and deep learning methodology for IoT data security and classification within cloud environments. The integration of AES-GCM and RSA provided efficient and robust protection for IoT data, significantly reducing the risk of unauthorized access. The decryption phase reliably restored the original data using the appropriate RSA private key and AES decryption, confirming the feasibility of the approach for safeguarding sensitive information during transmission.

Furthermore, the application of the Snake Optimizer in conjunction with the LSTM network led to notable improvements in the detection of both attack and benign data streams. As reflected in the results, the system consistently achieved high accuracy rates—exceeding 98% across diverse IoT devices and varying attack intensities—while maintaining low encryption and decryption times, as well as a minimal false alarm rate. These outcomes underscore the system’s reliability and adaptability in heterogeneous IoT environments.

The study also reported high specificity (95%) and sensitivity (97%) in real-world IoT traffic analysis, further validating the proposed framework’s capability for precise threat detection and accurate classification of IoT activities. This positions the methodology as a highly suitable solution for both intrusion detection and anomaly detection in cloud-based IoT systems.

Looking ahead, future work should focus on further fine-tuning the optimization techniques for real-time deployment and expanding performance evaluations across a broader range of emerging IoT and cloud computing scenarios. This will help address the dynamic nature of evolving cyber

References

[1] Shhatha, Amer Mohamed. "A comprehensive Analysis of Approaches and Difficulties for Cybersecurity Threats-Article Review." Mustansiriyah Journal of Pure and Applied Sciences vol.2, no.4, pp. 126-139, 2024. doi: https://doi.org/10.47831/mjpas.v2i4.164

[2] Younus, Z. S., and M. Alanezi. "A survey on network security monitoring: tools and functionalities." Mustansiriyah Journal of Pure and Applied Sciences vol. 1, no. 2, pp. 55-86, 2023, doi: https://doi.org/10.47831/mjpas.v1i2.33

[3] S. Ahmed and M. Khan, "Securing the internet of things (IoT): A comprehensive study on the intersection of cybersecurity, privacy, and connectivity in the IoT ecosystem," AI, IoT and the

Fourth Industrial Revolution Review, vol. 13, no. 9, pp. 1–17, Sep. 16, 2023. [Online]. Available: https://scicadence.com/index.php/AI-IoT-REVIEW/article/view/13/15

[4] T. Crepax and S. P. Rao, "Blockchain in the Cloud: A Primer on Data Security for Blockchain as a Service (BaaS)," SSRN, pp. 1–20, Mar. 4, 2021. [Online]. Available: https://ssrn.com/abstract=3766900

[5] H. Allioui and Y. Mourdi, "Exploring the full potentials of IoT for better financial growth and stability: A comprehensive survey," Sensors, vol. 23, no. 19, p. 8015, 2023, doi: 10.3390/s23198015.

[6] P. Lalitha, R. Yamaganti, and D. Rohita, "Investigation into security challenges and approaches in cloud computing," J. Eng. Sci., vol. 14, no. 08, 2023.

[7] R. J. Al-Hamadin, "A New Approach for Data Symmetric Key Cryptography Using Fast Neural Networks with Single Step of Back-propagation and Finite Fields," M.S. thesis, Princess Sumaya Univ. Technol., Jordan, 2021.

[8] A.M. Qadir and N. Varol, "A review paper on cryptography," in Proc. 7th Int. Symp. Digit. Forensics Security (ISDFS), IEEE, Jul. 11, 2019, pp. 1–6, doi: 10.1109/ISDFS.2019.8757514.

[9] P. Chatterjee, R. Bose, S. Banerjee et al., "Enhancing Data Security of Cloud Based LMS," Wireless Pers. Commun., vol. 130, pp. 1123–1139, Mar. 18, 2023, doi: 10.1007/s11277-023-10323-5.

[10] K. Shankar, S. K. Lakshmanaprabu, D. Gupta, A. Khanna, and V. H. C. de Albuquerque, "Adaptive optimal multi key based encryption for digital image security," Concurrency Comput. Pract. Exp., vol. 32, no. 4, e5122, Feb. 25, 2020, doi: 10.1002/cpe.5122.

[11] A.Cui, H. Zhao, X. Zhang, B. Zhao, and Z. Li, "Power system real time data encryption system based on DES algorithm," in 2021 13th Int. Conf. Measuring Technol. Mechatronics Autom. (ICMTMA), IEEE, 2021, pp. 220–228, doi: 10.1109/ICMTMA52658.2021.00056.

[12] C. Jia, R. Li, and Y. Wang, "Privacy protection scheme of DBSCAN clustering based on homomorphic encryption," J. Commun., vol. 42, no. 2, pp. 1–11, 2021, doi: 10.11959/j.issn.1000-436x.2021026.

[13] S. Fu, C. Zhang, and W. Ao, "Searchable encryption scheme for multiple cloud storage using double-layer blockchain," Concurrency Comput. Pract. Exp., vol. 34, no. 16, e5860, 2022, doi: 10.1002/cpe.5860.

[14] B.Wang, "Updatable ElGamal Encryption Scheme with Forward and Backward Security for Cloud Storage," in Frontiers in Cyber Security: 5th Int. Conf., FCS 2022, Springer Nature, Kumasi, Ghana, Dec. 13–15, 2022, pp. 324, doi: 10.1007/978-981-19-8445-7_21.

[15] Q. Y. Zhang and Y. J. Ba, "An adaptive speech homomorphic encryption scheme based on energy in cloud storage," Int. J. Netw. Secur., vol. 24, no. 4, pp. 628–641, 2022, doi: 10.6633/IJNS.20220724(4).05.

[16] P. Bagla, R. Sharma, A. K. Mishra, N. Tripathi, A. Dumka, and N. K. Pandey, "An Efficient Security Solution for IoT and Cloud Security Using Lattice-Based Cryptography," in 2023 Int. Conf. Emerging Trends Networks Comput. Commun. (ETNCC), Windhoek, Namibia, 2023, pp. 82–87, doi: 10.1109/ETNCC59188.2023.10284931.

[17] S. A. Jabber and S. H. Jafer, "A novel approach to intrusion-detection system: combining LSTM and the snake algorithm," Jordanian J. Comput. Inf. Technol., vol. 9, no. 4, 2023, doi: 10.5455/jjcit.71-1694088480.

[18] T. A. Joseph and N. Jayapandian, "Detection of Various Security Threats in IoT and Cloud Computing using Machine Learning," in 2022 Int. Conf. Sustainable Comput. Data Commun. Syst. (ICSCDS), Erode, India, 2022, pp. 996–1001, doi: 10.1109/ICSCDS53736.2022.9760791.

[19] A.Enemosah and G. I. Ogbonna, "Cloud security frameworks for protecting IoT devices and SCADA systems in automated environments," World J. Adv. Res. Rev., vol. 22, no. 3, pp. 2232–2252, 2024, doi: 10.30574/wjarr.2024.22.3.1485.

[20] K. Bella et al., "An efficient intrusion detection system for IoT security using CNN decision forest," PeerJ Comput. Sci., vol. 10, e2290, 2024, doi: 10.7717/peerj-cs.2290.

[21] M. Mohy-Eddine, A. Guezzaz, S. Benkirane, and M. Azrour, "An efficient network intrusion detection model for IoT security using K-NN classifier and feature selection," Multimedia Tools Appl., 2023, doi: 10.1007/s11042-023-14795-2.

[22] M. Almohaimeed and F. Albalwy, "Enhancing IoT Network Security Using Feature Selection for Intrusion Detection Systems," Appl. Sci., vol. 14, no. 24, 2024, doi: 10.3390/app142411966.

[23] K. Takahashi, K. Yamamoto, A. Kuchiba et al., "Confidence interval for micro-averaged F1 and macro-averaged F1 scores," Appl. Intell., vol. 52, pp. 4961–4972, 2022, doi: 10.1007/s10489-021-02635-5.

[24] K. Sadaf and J. Sultana, "Intrusion detection based on autoencoder and isolation forest in fog computing," IEEE Access, vol. 8, pp. 167059–167068, 2020, doi: 10.1109/ACCESS.2020.3022855.

[25] Y. Fu, Y. Du, Z. Cao, Q. Li, and W. Xiang, "A Deep Learning Model for Network Intrusion Detection with Imbalanced Data," Electronics, vol. 11, no. 6, Mar. 2022, doi: 10.3390/electronics11060898.

[26] L. Yu, L. Xu, and X. Jiang, "An Effective Method for Detecting Unknown Types of Attacks Based on Log-Cosh Variational Autoencoder," Appl. Sci., vol. 13, no. 22, p. 12492, 2023, doi: 10.3390/app132212492.