Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

An Efficient GRU-Based Intrusion Detection and Data Security Framework for IoT Networks Using Blockchain

Kshitiz Saxena
SRM Institute of Science & Technology
Dhowmya Bhatt
dhowmyab@srmist.edu.in

SRM Institute of Science & Technology

Research Article

Keywords: Unicode Transformation Format 32-Universally Unique Identifier (UTF32-UUID), Entropy Naor Reingold Pseudo Random Number Generator (ENR-PRNG), Exponential Public Keys-MenezesQu Vanstone (EPK-MQV), X-NOR Inversion-HAVAL (XNORI-HAVAL), Linear Scaling based Walrus Optimization Algorithm (LS-WaOA), Kendall Correlation Coefficient Concatenated Rectified Linear Units Gated Recurrent Unit (KCCCR-GRU), Robust Frobenius Isogenies Elliptic curve cryptography (RFIECC)

Posted Date: November 14th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11289479/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

An Efficient GRU-Based Intrusion Detection and Data Security Framework for IoT Networks Using Blockchain

Kshitiz Saxena[1], Dhowmya Bhatt [2*]

Kshitiz Saxena is in Department of Computer Science & Engineering, Faculty of Engineering & Technology, SRM Institute of Science & Technology, Delhi NCR Campus, Modinagar, Ghaziabad, India-201204.(Mobile Number:+919412706950, Email-Id: kshitizs@srmist.edu.in)

Dhowmya Bhatt is in Department of Computer Science & Engineering, Faculty of Engineering & Technology, SRM Institute of Science & Technology, Delhi NCR Campus, Modinagar, Ghaziabad, India-201204. (Email-Id: dhowmyab@srmist.edu.in)

Abstract: The Internet of Things (IoT) has transformed numerous sectors by facilitating smart devices for collecting and exchanging data. However, the diversity and scale of IoT deployments introduce significant security risks. This paper proposes a generalizable and robust intrusion detection and data security system for IoT using KCCCR-GRU and RFIECC with blockchain. The approach integrates advanced device authentication, feature optimization, and deep learning-based attack detection, with secure data integrity ensured by blockchain. To validate generalizability, the system is evaluated on multiple benchmark IoT datasets with varying attack vectors and device types. Comparative analysis includes recent state-of-the-art IoT intrusion detection systems, such as those based on deep reinforcement learning and graph neural networks. Beyond accuracy, we assess detection latency, resource usage, and scalability to reflect real-world deployment needs. The blockchain component is described in detail, including consensus, update frequency, and system integration. We also discuss deployment considerations, such as compatibility, cost, and integration challenges. The proposed system achieves high accuracy and practical efficiency, demonstrating its suitability for diverse, large-scale IoT environments.

Index Terms: Unicode Transformation Format 32-Universally Unique Identifier (UTF32-UUID), Entropy Naor Reingold Pseudo Random Number Generator (ENR-PRNG), Exponential Public Keys-MenezesQu Vanstone (EPK-MQV), X-NOR Inversion-HAVAL (XNORI-HAVAL), Linear Scaling based Walrus Optimization Algorithm (LS-WaOA), Kendall Correlation Coefficient Concatenated Rectified Linear Units Gated Recurrent Unit (KCCCR-GRU), Robust Frobenius Isogenies Elliptic curve cryptography (RFIECC).

1. INTRODUCTION

A vast network of interconnected physical devices that range from smart home appliances and wearable technology to industrial machinery and environmental sensors is referred to as IoT (Sai et al., 2021). These devices communicate with each other and also with central systems over the Internet, thereby facilitating automated processes, data collection, and real-time monitoring across numerous domains (Mishra & Pandya, 2021). IoT applications span various fields, comprising smart cities, healthcare, manufacturing, and agriculture, where they improve efficiency, mitigate costs, and enhance user experiences (Sundaram et al., 2022). Yet, significant security challenges have also been introduced by IoT's extensive adoption. With millions of devices connected to the internet, many of which have weak or insufficient security features. Thus, the risk of cyberattacks and data breaches grows considerably (Nimbalkar & Kshirsagar, 2021) (Parimala & Kayalvizhi, 2021). Data security in IoT involves employing strong measures for safeguarding against unauthorized access. To prevent attacks, namely replay and man-in-the-middle attacks, robust authentication methods are crucial (Chaurasia & Kumar, 2023). The need for effective encryption and authentication methods is emphasized by the recent flaws in IoT devices to ensure that only legitimate users and devices can access the system (Adil et al., 2022).

To overcome these challenges, BC technology can be implemented for creating a secure, decentralized system to store hashes that verify data integrity and protect against tampering. The system ensures that any unauthorized alterations to data are easily detectable by recording hash values on the BC (Lu et al., 2021) (Al Ahmed et al., 2022). Moreover, to identify and respond to potential threats in real-time, implementing an Intrusion Detection System (IDS) is vital. For detecting suspicious activities and protecting IoT devices and data, IDS applications monitor network traffic (Yilmaz et al., 2021) (Elsayed et al., 2023). A robust IDS is significant not only for detecting malicious intrusions in IoT infrastructures but also for preventing these attacks, thus ensuring overall security (Krishnan et al., 2022) (Wu et al., 2022).

However, challenges like long training times, high computational demands, and gradient issues are faced by conventional intrusion detection approaches, such as Machine Learning (ML) algorithms, comprising Deep Neural Networks (DNNs) as well as Random Forests (RF) and hybrid models. Hybrid models, while powerful, can be complex and difficult to integrate. However, conventional methodologies may not sufficiently protect IoT data, emphasizing the need for more robust solutions (Vishwakarma & Kesswani, 2022) (Sarwar et al., 2022). To tackle these limitations and improve security, it is crucial to develop an accurate IDS. Enhancing intrusion detection and device-level protection using KCCCR-GRU and RFIECC techniques is the aim of the proposed system. These methods are designed for enhancing the efficacy of intrusion detection

and device-level protection, thus ensuring a more robust defense against emerging threats.

1.1 Problem Statement

Some of the issues that are noticed in the prevailing works are listed as follows:

- Security risks were introduced when the shared IoT network access simplifies data transactions for end users to collect data. None of the existing systems focused on authorizing whether the sensed data comes through a registered router device or not.
- It was noted in (Alruwaili et al., 2023) that default SSIDs on many routers revealed their brand and model, which made it easier for potential hackers to target and breach the network.
- Universal Plug and Play (UPnP)-enabled device communication was vulnerable to exploitation in (Nallakaruppan et al., 2024), thus affecting the security performance.
- In (Abbas et al., 2021), nonce replay attacks were not addressed; thus, the attackers could easily hack the nonce used in BC to create unique hashes.
- Existing solutions often utilized hash-based authentication, which was vulnerable to birthday attacks owing to shorter input lengths, thus compromising system security.

To overcome these issues, the objectives proposed in this work are mentioned as follows:

- To address the issue of exposing sensitive data across the IoT network, the solution involves authenticating devices using the EPK-MQV-MAC technique.
- Using UTF32-UUID to generate unique SSIDs addresses the issue of default SSIDs, thus revealing router brands and models and improving network security.
- To address the exploitation issue and enhance security, the ENR-PRNG algorithm is used to generate a complex router PIN.
- To secure against nonce replay attacks, the SSID is recorded as a nonce on the BC.
- To improve security and mitigate birthday attacks on hash-based authentication, XNORI-HAVAL is employed for stronger and more secure hashing.

The paper is organized as: The related works on the proposed system are revealed in Section 2. The proposed methodology is detailed in Section 3. The performance of the proposed model is evaluated in Section 4. At last, the paper is concluded in Section 5 with suggestions for future work.

2. LITERATURE SURVEY

(Chintapalli et al., 2024) evaluated the effectiveness of an Osprey Optimization Algorithm (OOA) modified Bidirectional Long Short-Term Memory (Bi-LSTM) network for intrusion detection in IoT systems. Primarily, from three datasets, data was acquired and further preprocessed using the min-max normalization technique. Lastly, using Bi-LSTM, intrusion attacks were classified as normal or attacked types, achieving high detection accuracy and reduced processing time. Nevertheless, due to Bi-LSTM networks' increased computational and memory demands, they were slower and required more training time.

(Alruwaili et al., 2023) implemented an IDS for secure IoT using the Red Kite Optimization Algorithm with an Average Ensemble Model for Intrusion Detection (RKOA-AEID) method. Initially, using min-max normalization, the data was preprocessed. After that, using the RKOA and ensemble learning model, features were selected and classified, respectively. Hence, the study produced security solutions for IoT-based Wireless Sensor Networks (WSN) with improved accuracy. Nevertheless, the model was not yet tested with large-scale real-time databases, thus restricting its practical assessment.

(Bakhsh et al., 2023) showed an improved IoT network security by implementing a Deep Learning (DL)-powered IDS. Firstly, by using the Synthetic Minority Oversampling Technique (SMOTE), the dataset was balanced. Lastly, the processed data was utilized for training and evaluating Feed Forward Neural Network (FFNN), LSTM, and Random Neural Network (RandNN) models that detected and classified various cyberattacks effectively, thereby demonstrating higher accuracy in distinguishing betwixt normal and malicious activities. Yet, the risk of adding noisy data was carried and overfitting was caused by SMOTE, which could limit the model's effectiveness on data.

(Nallakaruppan et al., 2024) assessed the host-based IDS's enhanced security for the IoT. Initially, the system gathered data and carried out feature engineering. For categorizing attacks, ML techniques, including linear regression, Support Vector Machine (SVM), and RF were applied. To block and report malicious activities, the Intrusion Prevention System (IPS) then adapted its policies based on IDS results, enhancing security and addressing threats with higher precision and recall. However, due to delays in implementing protective measures, the system remained at risk.

(Roy et al., 2022) conducted a lightweight supervised intrusion detection mechanism for IoT networks. The model used a B-Stacking Classifier, which integrated K-Nearest Neighbors (KNN), RF, and XGBoost as base learners. This approach classified network traffic as either benign or various types of intrusions. Also, it was appropriate for deployment on devices with limited power and storage. But, it was not tested in real-world IoT networks, which limited its practical evaluation.

(Kaushik & Al-Raweshidy, 2024) established an IDS for IoT devices and data. Data was inputted and pre-processed, followed by feature extraction utilizing Teaching-Learning-Based Optimization (TLBO). Lastly, the network traffic was successfully classified as either normal or intrusive by the system, demonstrating effective intrusion detection capabilities with low overhead and high detection rates. Yet, the system lacked encryption, which could have exposed it to potential security vulnerabilities.

(Mohy-Eddine et al., 2023) employed an ensemble learning-centric intrusion detection model for Industrial IoT (IIoT) security. To identify normal behavior or intrusions, the decision core analyzed the data using an RF classifier. At last, the response module managed detected intrusions by alerting

administrators, logging incidents, or taking automated actions with lesser prediction time. However, the high memory and computational costs of the RF classifier could affect its efficiency, especially with larger datasets.

(Aljehane et al., 2024) integrated Golden Jackal Optimization Algorithm (GJOA) with DL-assisted IDS for network security. Using the attention-based Bi-LSTM model, data was classified in the study. In classification, the Salp Swarm Algorithm (SSA) tuned the hyperparameters. The robust performance in identifying intrusions within less computational time was demonstrated by the simulation. Nevertheless, the SSA faced the challenge of poor exploitation problem, thereby affecting its overall efficiency and effectiveness.

(Kaur et al., 2023) estimated an EGCrypto, a Low-Complexity Elliptic Galois Cryptography (LCEGC) model to securely transmit data in IoT. Here, Optimal Matrix-centric XOR (OM-XOR) was used in this study to embed the encrypted data into the blocks, thus ensuring it was concealed and secure. In this approach, both secure and efficient data transmission in IoT networks were ensured by improved computational speed. But, EGCrypto was not yet optimized for large-scale networks or advanced attacks and lacked real-world validation.

(Abbas et al., 2021) presented a secure routing strategy that integrates Genetic Algorithms (GA) with Software-Defined Networking (SDN) to enhance BC-based IoT systems. Primarily, methods, such as securing node authentication through Lightweight Registration and Authentication (LRA), optimizing data paths with GA, validating routes using smart contracts, and detecting malicious nodes were employed by the study. Hence, the study optimized route calculation and malicious node detection with a decreased packet drop rate when malicious nodes were augmented. Nevertheless, GA had limitations, namely slow and premature convergence and the need for parameter tuning, thereby posing challenges in achieving optimal performance.

However, many of these works are limited by narrow dataset usage, lack of practical deployment metrics, or insufficient integration with decentralized security mechanisms like blockchain. Our work addresses these limitations by providing a holistic, experimentally validated solution.

3. PROPOSED METHODOLOGY FOR ROUTER REGISTRATION AND DATA SECURITY USING BLOCKCHAIN IN IOT

In the proposed system, by using UTF32-UUID, ENR-PRNG, KCCCR-GRU, and RFIECC, the router is registered and authenticated with enhanced intrusion detection and data security mechanisms. Several key phases, such as router registration, key generation center, device authentication, IoT sensor node setup, BC storage, router status verification, intrusion detection, and data security are comprised in the framework. Figure 1 shows this proposed framework's structure.

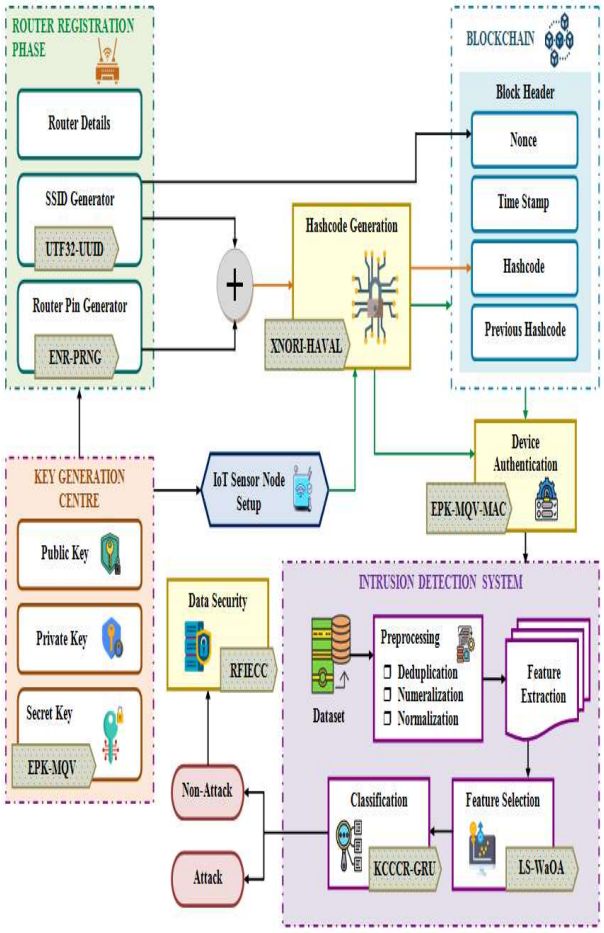


Figure 1: Framework of the Proposed System

3.1 Router Registration Phase

Primarily, the router's details (η), such as product name, serial number, software version, default gateway, and so on are entered in the router registration phase and are represented as,

$$\eta = \{\eta_1, \eta_2, \eta_3, \ldots\ldots \eta_{Z''}\} \tag{1}$$

Where, the number of details entered during the registration is notated as (Z''). An SSID and a router PIN are generated for the entered details. The network is uniquely identified by SSID, whereas device authentication is secured by the router PIN, which is described as follows,

- **SSID Generator**

A unique name assigned to a wireless network is SSID, which allows devices to identify and connect to it. In this case, for generating the SSID, UTF32-UUID is utilized. A 128-bit identifier that is used to ensure uniqueness across systems using random numbers is termed UUID. For enhancing the complexity of the UUID Unicode Transformation Format 32 (UTF32) encoding is applied, resulting in a more complex and secure SSID. Initially, by using the UTF32-UUID technique, the SSID number (G) is generated for (η). The SSID number encoded by UTF32-UUID is expressed as,

$$\mathrm{H} = \Im(G) \tag{2}$$

Here, the output of the 32-bit binary representation of (G) is notated as (H), and the UTF32 operation that encodes SSID into a complex number is signified as $(\Im)$. The encoded SSID is updated in the BC as a nonce number, which is described in section 3.6.

Router PIN Generator

To connect the authenticated devices to the router, the router PIN, which is generated by ENR-PRNG, is used. By creating a sequence of random numbers through mathematical formulas, the Pseudo Random Number Generator (PRNG) generates the router's PIN. The Entropy Naor Reingold (ENR) technique is used to enhance the security of the router PIN and address issues, such as brute-force attacks and uneven distribution in random number generation. Primarily, the input value (η) is initialized and further ENR is utilized for enhancing the randomness and distribution as,

$$\prod_{enr}(\eta)=\left(\left(g^{f_0\bullet f_1^{\eta_1}\bullet f_2^{\eta_2}.....f_n^{\eta_n}}\right)\bmod p*\left(-\sum_{i=1}^{\aleph}\eta(i)\log\eta(i)\right)\right) \tag{3}$$

Where, the total number of values (i) in (η) is specified as $(\aleph)$, the Naor Reingold function applied on (η) is specified as $\prod_{enr}(\eta)$, a prime number is exemplified as (p), the generator is indicated as (g), and the secret vector, which has (n) vectors, is denoted as $(f=f_0,f_1,....f_n)$.

$$\mathrm{M}_{\tilde{f}+1}=(\iota\bullet\prod_{enr}(\eta)+\tau)\bmod p' \tag{4}$$

Where, the new seed (value) is illustrated as $(\mathrm{M}_{\tilde{f}+1})$, and the constants are given as $(\iota),(\tau)$, and (p'). Further, random numbers $(\ddot{\mathrm{N}})$ are generated based on this $(\mathrm{M}_{\tilde{f}+1})$ and are shown as,

$$\ddot{\mathrm{N}}=\frac{\mathrm{M}_{\tilde{f}+1}}{p'} \tag{5}$$

To produce sequence numbers of pseudo numbers, the abovementioned steps are repeated. The generated pseudo-random numbers are converted into PIN format. This format is the router PIN and is signified as $(\ddot{\aleph})$. Subsequently, the SSID (H) and router PIN $(\ddot{\aleph})$ are first combined by the XOR operation as,

$$\gamma=\mathrm{H}\oplus\ddot{\aleph} \tag{6}$$

Here, the output of the combined SSID and router PIN, which is further hashed and stored in the BC, is notated as (γ). In section 3.5, the hashcode generation is described.

3.2 IoT Sensor Node Setup

Here, IoT sensor nodes are initialized. Numerous sensor nodes $(\breve{\lambda})$ are initialized and are specified as,

$$\breve{\lambda}=\left\langle\breve{\lambda}_1,\breve{\lambda}_2,\breve{\lambda}_3,.....\breve{\lambda}_{\ddot{\Theta}}\right\rangle \tag{7}$$

Where, the total number of IoT sensor nodes is specified as $(\ddot{\Theta})$. A fundamental component of an IoT network designed for collecting, processing, and transmitting data from its environment to other devices or systems is the IoT sensor.

3.3 Key Generation Centre

Keys are generated for both the router and each IoT sensor node in parallel with the router registration and IoT initialization process. As described in section 3.8, keys like public and private are produced. By using EPK-MQV, the secret keys are generated. Cryptographic keys are securely exchanged over insecure communication by Menezes Qu Vanstone (MQV). However, MQV has issues with public parameter sharing and man-in-the-middle attacks. To overcome the problem, the exponential of the sender and receiver's public keys is used to enhance security within the MQV algorithm. The EPK-MQV is explained as,

Key Generation

The sender's public and private keys are signified as (α) and (β), correspondingly. Likewise, the receiver's public and private keys are specified as $(\mathcal{J})$ and $(\mathcal{L})$, correspondingly. To solve the aforesaid issues, the exponential of the public keys is explained as follows,

Exponential Public Keys

Exponential values of public keys are used for secure key exchange in which the sender and receiver compute the exponential of the sender's public key $(\ddot{Q})$ and receiver's public key (Q), correspondingly, which is shown as,

$$\ddot{Q}=\exp(\alpha) \tag{8}$$

$$Q=\exp(\mathcal{J}) \tag{9}$$

These obtained exponential keys are exchanged between them to produce secret keys.

Key Exchange

For the generation of secret key, the shared secret (T) is computed for both sender T_{send} and receiver T_{rec}, which are given as,

$$T_{send}=\beta*Q \tag{10}$$

$$T_{rec}=\mathcal{L}*\ddot{Q} \tag{11}$$

Lastly, if both sender and receiver have the same secret (T), then the secret key is obtained and is denoted as $(\ddot{\mathrm{T}})$. For device authentication, this secret key is used for IoT sensor device setup. Thereafter, the hashcode generation phase is given as follows,

3.4 Hashcode Generation

By using the XNORI-HAVAL technique, the hashing of (γ) is done. Hash Algorithm Verification and Authentication Laboratory (HAVAL) is a one-way hashing algorithm that produces a large, complex hash code. However, HAVAL faces issues with the fixed length of the hash output. To overcome this vulnerability, the X-NOR approach is applied to determining the bit length; to enhance security, the Inversion methodology is used in bitwise operations. Firstly, (γ) is initialized as,

$$\gamma_{\tilde{Y}} = \left\{\gamma_1, \gamma_2, \gamma_3, \ldots\ldots\ldots \gamma_{\breve{\mathbb{A}}-1}\right\},\ \tilde{Y} = \left\{1,2,3\ldots\breve{\mathbb{A}}\right\} \tag{12}$$

Here, the number of γ initialized is notated as $(\breve{\mathbb{A}}-1)$. Further, (γ) is padded and partitioned into blocks $(\pm_{block}^{\tilde{Y}})$. The hash value is computed with the X-NORI for each block to avoid the fixed-length hash values as follows,

$$\gamma_{\tilde{Y}+1} = \neg\left(\theta''\left(\gamma_{\tilde{Y}} \otimes \pm_{block}^{\tilde{Y}}\right)\right) \tag{13}$$

Where, XNORI applied output is denoted as $\gamma_{\tilde{Y}+1}$. The final hashcode is produced after processing each block of (γ) and is denoted as $(\breve{\Lambda})$. It is then updated and stored on the BC for verification of the router and IoT sensor node. Similarly, the IoT sensor node $(\breve{\lambda})$data is also hashed. The hashcode value of the router connected with IoT is verified with the previously stored hashcode in BC during communication. If the verification is successful, then the hashcode is converted into a Message Authentication Code (MAC), and the device is authenticated.

3.5 Device Authentication

By using EPK-MQV-MAC, device authentication is performed. Here, the secret key generated by EPK-MQV is used for device authentication. For authentication, MAC, which plays an essential role in ensuring the integrity and authenticity of the data exchange between devices, is wielded. By combining $(\ddot{\mathrm{T}})$ and $(\breve{\Lambda})$, the MAC $(\ddot{\mathbb{X}})$ is created and is shown as,

$$\ddot{\mathbb{X}} = \ddot{\mathrm{T}} \oplus \breve{\Lambda} \tag{14}$$

The $\ddot{\mathbb{X}}$ is checked for similarity with the server in which MAC is already created.

$$\Pi = \begin{cases} \ddot{\mathbb{M}} \ if\left(\ddot{\mathbb{X}} = Server\right) \\ \ddot{\mathbb{X}} \ otherwise \end{cases} \tag{15}$$

Here, the authenticated device is notated as $\ddot{\mathbb{M}}$, and the unauthenticated device is signified as $\ddot{\mathbb{X}}$. For securing the data in the server, the authenticated device's data is given into the intrusion detection phase. Thus, the unauthenticated device's data is blocked.

3.6 Blockchain

Each block header in the BC section encompasses numerous pieces of information, including authentication details. The SSID number generated by UTF32-UUID is stored as a nonce number to find the valid hash for the block, which is crucial for maintaining BC's security. Furthermore, the timestamp records the exact time when the block is created. The SSID and router PIN combined by the XOR operation, which is denoted as (γ), are converted into a hash and stored in the BC by using the XNORI-HAVAL hashing technique. As already discussed in section 3.4, the block header also includes the hashcode and the previous hashcode, which are crucial for verifying the continuity and integrity of the BC. These elements together secure and continuously update the block for maintaining authorized access to the data. The BC plays a key role in the authentication of the device.

Blockchain integration in our approach is implemented using the Hyperledger Fabric platform, which is well-suited for permissioned IoT environments. The consensus mechanism employed is the SOLO protocol, chosen for its rapid agreement capabilities and low computational overhead. Hash updates to the blockchain occur upon key events such as device registration, authentication, or at configurable periodic intervals, allowing flexibility based on deployment requirements. Communication between IoT devices and the intrusion detection system (IDS) with the blockchain is facilitated through RESTful APIs and smart contracts, enabling real-time integrity verification and efficient key management. The blockchain stores critical elements including the Service Set Identifier (SSID) used as a nonce, the hashed router Personal Identification Number (PIN), authentication hashes, and timestamps, collectively ensuring secure and tamper-evident device authentication and data integrity.The intrusions are detected in the intrusion detection phase as explained in the following section.

3.7 Intrusion Detection System

Intrusion detection is done in this phase for the data received from authenticated devices. Primarily, the model is trained by collecting the data, pre-processing, feature extraction, feature selection, and classification, and these processes are explained below,

3.7.1 Dataset

The detection process initiates with data collection from the Kitsune Network Attack Dataset. The input data (A) are signified as,

$$\mathrm{A} = \left\langle \mathrm{A}^1, \mathrm{A}^2, \mathrm{A}^3, \ldots\ldots\mathrm{A}^a \right\rangle \tag{16}$$

Where, the total number of data collected from the dataset is specified as (a).

3.7.2 Pre-Processing

Cleaning and preparing data for accurate and effective analysis is the purpose of preprocessing. The steps are given as follows.

Deduplication

Here, the input data (A) may have duplicates removed by deduplication. This approach involves identifying and eliminating duplicate records from (A) for removing redundant data and reducing storage space. The deduplicated data (B) is given as,

$$\mathrm{B} = \zeta(\mathrm{A}) \tag{17}$$

Here, the duplicate removal operation to remove any repeated entries in (A) is notated as (ζ).

Numeralization

The (B) often comprises symbolic features that are converted into a numerical format for easy processing and system understanding of the data. Thus, (C) denotes the numerical converted data, which is normalized as follows,

Normalization

The min-max normalization is utilized for converting all numerical data (C) on the same scale. The normalization process is explained as,

$$C_{nor} = \frac{C_{num} - C_{\min}}{C_{\max} - C_{\min}} \tag{18}$$

Where, the minimal and maximal values of (C) are specified as $(C_{\min})$ and $(C_{\max})$, and (C_{nor}) is the normalized data given to the feature extraction phase. The pre-processed output is notated as (E).

3.7.3 Feature Extraction

From (E), the features, such as duration, protocol type, service, flag, source bytes, destination bytes, land, wrong fragment, urgent, hot, etc., are extracted and are expressed as,

$$F = \left[F_1, F_2, F_3, \ldots, F_b\right] \tag{19}$$

Here, (F) and (b) indicate the extracted feature and the total number of extracted features, respectively.

3.7.4 Feature Selection

From (F), the optimal features are selected in this phase using LS-WaOA. The Walrus Optimization Algorithm (WaOA) can find a global optimum solution with fewer iterations. Nevertheless, it encounters challenges with new position replacement, leading to outliers and convergence issues owing to the constraints imposed by lower and upper bounds. To address these challenges, the Linear Scaling Technique is implemented in WaOA, which adjusts the range of feature values for preventing outliers and enhancing convergence. The key steps are given below,

Initialization

Primarily, the population of walruses (F) (extracted features) and the starting position of the walruses are initialized as,

$$F = \begin{bmatrix} F_1 \\ \vdots \\ F_r \\ \vdots \\ F_U \end{bmatrix}_{U\times m} = \begin{bmatrix} o_{1,1} & \cdots & o_{1,q} & \cdots & o_{1,m} \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ o_{r,1} & \cdots & o_{r,q} & \cdots & o_{r,m} \\ \vdots & \iddots & \vdots & \ddots & \vdots \\ o_{U,1} & \cdots & o_{U,q} & \cdots & o_{U,m} \end{bmatrix}_{U\times m} \tag{20}$$

Where, the (r^{th}) walrus in the population is denoted as (F_r), the value of the (r^{th}) decision variable suggested by the (q^{th}) walrus is signified as $(o_{r,q})$, the number of walruses is represented as (U), and the number of decision variables is denoted as (m).

Fitness

Now, based on the maximum accuracy, the fitness value $(\wp_r)$ is calculated. The fitness $(\wp)$ is given as,

$$\wp_r = \max\left[\dddot{\Omega}\right] \tag{21}$$

Here, the maximum accuracy is denoted as $(\dddot{\Omega})$. Subsequently, based on $(\wp_r)$, the position of the walruses gets updated as,

Phase 1: Feeding Strategy (Exploration)

Based on the strongest walrus (best candidate solution), the position of each walrus (F) is updated as,

$$o_{r,q}^{\ddot{P}1} = o_{r,q} + rand_{r,q} \bullet \left(\int_q - \text{Ɨ}_{r,q} \bullet o_{r,q} \right) \tag{22}$$

Where, the updated position for the (q^{th}) dimension of the (r^{th}) walrus on 1st phase is represented as $o_{r,q}^{\ddot{P}1}$, a random number from the interval [0,1] is shown as $rand_{r,q}$, the integers randomly selected between 1 or 2 is given as $\text{Ɨ}_{r,q}$, and the (q^{th}) dimension of the best candidate solution is provided as $\int_q$. The new position replaces the previous position if the $(\wp_r)$ value increases. It is given as,

$$F_r = \begin{cases} o_{r,q}^{\ddot{P}1}, \wp_r^{\ddot{P}1} < \wp_r \\ F_r, else \end{cases} \tag{23}$$

Here, an objective function for the new position $o_{r,q}^{\ddot{P}1}$ is represented as $\wp_r^{\ddot{P}1}$.

Phase 2: Migration

Each walrus's position is updated by migrating towards another randomly selected (F_v). The new position is updated if the objective function value augments and is given as,

$$o_{r,q}^{\ddot{P}2} = \begin{cases} o_{r,q} + rand_{r,q} \bullet \left(o_{v,q} - \text{Ɨ}_{r,q} \bullet o_{r,q} \right) \\ o_{r,q} + rand_{r,q} \bullet \left(o_{r,q} - o_{v,q} \right) \end{cases} \quad \begin{matrix} if \wp_v < \wp_r \\ otherwise \end{matrix} \tag{24}$$

$$F_r = \begin{cases} o_{r,q}^{\ddot{P}2}, \wp_r^{\ddot{P}2} < \wp_r \\ F_r, otherwise \end{cases} \tag{25}$$

Where, the position of a randomly selected walrus from (F_r) is denoted as $o_{v,q}$, the objective function of the randomly selected walrus is notated as $\wp_v$, and the objective function of phase 2 is represented as $\wp_r^{\ddot{P}2}$.

Phase 3: Escaping and Fighting against Predators (Exploitation)

The position of each walrus is updated using the linear scaling technique by simulating local search in a neighborhood with variable radius as,

$$o_{r,q}^{\ddot{P}3} = \frac{\left(o_{r,q} - o_{r,q}^{\min} \right)}{\left(o_{r,q}^{\max} - o_{r,q}^{\min} \right)} \times \left(lb_{upper}^{t'''} - lb_{lower}^{t'''} \right) + lb_{lower}^{t'''} \tag{26}$$

$$F_r = \begin{cases} o_{r,q}^{\ddot{P}3}, \wp_r^{\ddot{P}3} < \wp_r \\ F_r, otherwise \end{cases} \tag{27}$$

Here, the phase 3 objective function is signified as $\left(\wp_r^{\ddot{P}3} \right)$, the lower and upper bounds are notated as $\left(lb_{lower}^{t'''} \right)$ and $\left(ub_{upper}^{t'''} \right)$, and the iteration contour is indicated as (t''').

Convergence Step

Lastly, the processes are repeated until convergence is met. From (F), the best optimal solution is selected and is signified as $(\dddot{\Lambda})$.

3.7.5 Classification

Here, the attacks are classified based on the selected features $(\dddot{\Lambda})$ using KCCCR-GRU. Gated recurrent unit (GRU) is easy to train and has lower parameters; however, it suffers from gradient vanishing problems with slow convergence and low learning efficiency. To overcome these problems, a Kendall Correlation Coefficient (KCC)-based normalization layer is utilized for enhancing accuracy. After that, a Concatenated Rectified Linear Unit (CReLU) is utilized for solving slow convergence. In Figure 2, the architecture diagram of the KCCCR-GRU classifier is exhibited.

KCCCR-GRU classifier comprises a reset gate, update gate, candidate hidden state, and hidden state that are explained below in detail.

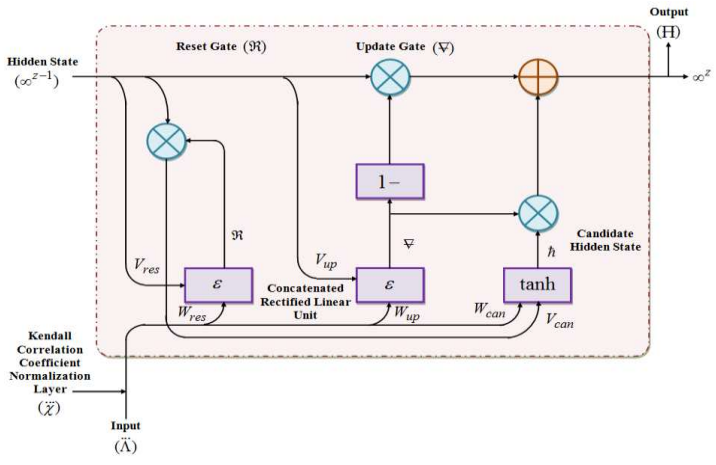


Figure 2: Architecture of KCCCR-GRU classifier

Step 1: Initialization

Firstly, $\left(\dddot{\Lambda}\right)$ is taken as input for the proposed classifier with iterations $\left(\hat{r}, \hat{r}^{\max}\right)$. The following steps are carried out when $\hat{r}$ is less than $\hat{r}^{\max}$. Primarily, KCC is utilized for normalizing the input $\left(\dddot{\Lambda}\right)$ as,

$$\dddot{\chi} = \frac{\dddot{\Lambda}_{con} - \dddot{\Lambda}_{dis}}{\dddot{\Lambda}_{con} + \dddot{\Lambda}_{dis}} \tag{28}$$

Where, the output of normalized input by KCC is illustrated as $\left(\dddot{\chi}\right)$, and $\dddot{\Lambda}_{con}$ and $\dddot{\Lambda}_{dis}$ are the number of concordant and discordant pairs among the selected features $\left(\dddot{\Lambda}\right)$.

Step 2: Reset Gate

In this gate, the neural network updates its memory centered on the information required to be retained or discarded. Here, to solve slow convergence, CReLU $\left(\varepsilon\right)$ is used. The output of the reset gate $\left(\Re\right)$ is rendered as,

$$\Re = \varepsilon\left(\left(W_{res} \bullet \dddot{\chi}\right) + \left(V_{res} \bullet \infty^{z-1}\right) + \dddot{B}\right) \tag{29}$$

$$\varepsilon = \left(\max\left(0, \dddot{\chi}\right), \max\left(0, -\dddot{\chi}\right)\right) \tag{30}$$

Here, $\left(\varepsilon\right)$ signifies the Concatenated Rectified Linear Units, $\left(\infty^{z-1}\right)$ specifies the previous hidden state output with time $\left(z\right)$, $\left(W_{res}\right)$ and $\left(V_{res}\right)$ denote the weights of $\left(\dddot{\chi}\right)$ and $\left(\infty^{z-1}\right)$ in reset gate, and the bias value is represented as $\dddot{B}$.

Step 3: Update Gate

The update gate decides which parts of the previous information should be kept in the memory once the reset gate is calculated. The update gate $\forall$ is shown as,

$$\forall = \Re * \left\{1 - \varepsilon\left(\left(W_{up} \bullet \dddot{\chi}\right) + \left(V_{up} \bullet \infty^{z-1}\right) + B\right)\right\} \tag{31}$$

Where, the weights of $\left(\dddot{\chi}\right)$ and $\left(\infty^{z-1}\right)$ in update gate are exemplified as $\left(W_{up}\right)$ and $\left(V_{up}\right)$, and bias value is given as B. Further, the current hidden state is computed.

Step 4: Candidate Hidden State

The candidate's hidden state aids the KCCCR-GRU in deciding what new information should be included in the hidden state. It is equated as,

$$\hbar = \tanh\left(\left(W_{can} \bullet \dddot{\chi}\right) + \forall \otimes \left(V_{can} - \infty^{z-1}\right) + \breve{B}\right) \tag{32}$$

$$\tanh = \frac{\dot{e}^{\ddot{\chi}} - e^{\ddot{\chi}}}{\dot{e}^{\ddot{\chi}} + e^{\ddot{\chi}}} \tag{33}$$

Here, the candidate hidden state is shown as $\hbar$, $\left(\dot{e}\right)$ signifies exponential term, the weights of $\left(\dddot{\chi}\right)$ and $\left(\infty^{z-1}\right)$ in candidate update gate are represented as $\left(W_{can}\right)$ and $\left(V_{can}\right)$, and bias value is given as $\left(\breve{B}\right)$.

Step 5: Hidden State and Output

Here, by combining the past hidden state and the candidate hidden state, the present hidden state $\left(\infty^{z}\right)$ is computed as,

$$\infty^{z} = \left(1 - \forall\right) \otimes \infty^{z-1} + \forall \otimes \hbar \tag{34}$$

Finally, the classified output $\left(\text{Ħ}\right)$ obtained concerning $\left(\infty^{z}\right)$ is given as follows,

$$\text{Ħ} = \begin{cases} \text{Ħ}^{1} \\ \text{Ħ}^{2} \end{cases} \tag{35}$$

Where, the attacked $\left(\text{Ħ}^{1}\right)$ and non-attacked $\left(\text{Ħ}^{2}\right)$ class output are given as $\left(\text{Ħ}\right)$. The pseudo-code for the KCCCR-GRU classifier is shown below,

Pseudocode for KCCCR-GRU

Input: Selected Features $\left(\dddot{\Lambda}\right)$

Output: Classified Attack $\left(\text{Ħ}\right)$

Begin

Initialize iterations $\left(\hat{r}, \hat{r}^{\max}\right)$, $\dddot{\Lambda}_{con}$ and $\dddot{\Lambda}_{dis}$

While $\left(\hat{r} < \hat{r}^{\max}\right)$

For each $\left(\dddot{\Lambda}\right)$do

Normalize $\left(\dddot{\Lambda}\right)$

$$\dddot{\chi} = \frac{\dddot{\Lambda}_{con} - \dddot{\Lambda}_{dis}}{\dddot{\Lambda}_{con} + \dddot{\Lambda}_{dis}}$$

Evaluate reset gates

$$\Re = \varepsilon\left(\left(W_{res} \bullet \dddot{\chi}\right) + \left(V_{res} \bullet \infty^{z-1}\right) + \dddot{B}\right)$$

Apply CReLU $\left(\varepsilon\right)$

$$\varepsilon = \left(\max\left(0, \dddot{\chi}\right), \max\left(0, -\dddot{\chi}\right)\right)$$

Calculate update gates $\left(\forall\right)$

Compute hidden and output gates $\left(\infty^{z}\right)$

$$\infty^{z} = \left(1 - \forall\right) \otimes \infty^{z-1} + \forall \otimes \hbar$$

End for

End while

Return (Ħ)

End

From these identified attacks, the non-attacked data is protected by robust encryption as detailed below,

3.8 Data Security

By using the RFIECC algorithm, the non-attacked data (Ħ^2) obtained from the classification phase is encrypted. Elliptic Curve Cryptography (ECC), which performs faster encryption and decryption, is the most strongest and secure algorithm. It is mainly utilized for data security. However, there is an issue with generating private keys from random values. Thus, Robust Frobenius Isogenies (RFI)-based private key is generated. The RFIECC is explained as,

Curve Creation

The elliptic curve $(\ddot{\text{Ë}})$ is explained over the finite field. The curve is typically described as,

$$\ddot{\text{Ë}} \to \ddot{\omega}^2 = (\kappa^2 + \text{ſ})\kappa + \Sigma \tag{36}$$

Here, constants are indicated as (ſ) and (Σ), the x-coordinate of a point on the elliptic curve is notated as $(\ddot{\omega})$, and the y-coordinate of a point on the elliptic curve is given as (κ). Here, Robust Frobenius Isogeny is applied to the curve $(\ddot{\text{Ë}})$ for solving generating private keys by random values. The expression of RFI is exhibited as,

$$\beta(\ddot{\omega}, \kappa) = (\ddot{\omega}^{\rho}, \kappa^{\rho}) \tag{37}$$

Where, the positive characteristics gathered from RFI to provide the private key are indicated as (ρ). Thus, based on this, the private key (β) is generated as,

Key Formation

Now, the private and public keys are generated for performing encryption and decryption. The public key (α) is utilized for generating the private key (β) centered on the RFI and is articulated as,

$$\alpha = \beta * \text{Ŧ} \tag{38}$$

Here, the point chosen from $(\ddot{\text{Ë}})$ is denoted as Ŧ. The data is encrypted after the creation of (α) and (β), which is given as follows.

Encryption

The process of converting the plaintext into cipher text is termed encryption. Here, two cipher texts are created with a message Ħ^2 that has a point Ŧ on the curve and then, $\ddot{\imath}$ is selected at random. The cipher texts are shown as,

$$\text{Ć}^1 = \ddot{\imath} * \text{Ŧ} \tag{39}$$

$$\text{Ħ}^* = \text{Ħ}^2 + \ddot{\imath} * \alpha \tag{40}$$

Where, cipher texts 1 and 2 are indicated as Ć^1 and Ħ^*, respectively. The pseudo-code for RFI-ECC is given below:

Pseudo-code for RFIECC

Input: Non-Attacked Data (Ħ^2)

Output: Secured Data (Ħ^*)

Begin

Initialize (Ħ^2), $\ddot{\omega}, \kappa$

For (Ħ^2)

Elliptic Curve Formation

$\ddot{\text{Ë}} \to \ddot{\omega}^2 = (\kappa^2 + \text{ſ})\kappa + \Sigma$

Evaluate Frobenius Isogenies

$\beta(\ddot{\omega}, \kappa) = (\ddot{\omega}^{\rho}, \kappa^{\rho})$ # for private key generation

Choose private key (β)

Calculate public key (α)

$\alpha = \beta * \text{Ŧ}$

Compute Ć^1 and Ħ^* # for encryption

$\text{Ć}^1 = \ddot{\imath} * \text{Ŧ}$

$\text{Ħ}^* = \text{Ħ}^2 + \ddot{\imath} * \alpha$

Decrypt (Ħ^2)

$\text{Ħ}^2 = \text{Ħ}^* - \beta * \text{Ć}^1$

End for

Obtain Secured Data (Ħ^*)

End

During the transmission time, the receiver decrypts the encrypted data to access the data or information. The encrypted data is decrypted by the following equation,

$$\text{Ħ}^2 = \text{Ħ}^* - \beta * \text{Ć}^1 \tag{41}$$

The decrypted data or message is shown in the above equation (41). Further, the proposed work's performance assessment is discussed below,

4. RESULT AND DISCUSSION

The experiments for evaluating the proposed system were conducted using an 80/20 train-test split for each dataset to ensure a balanced and effective assessment of model performance. The implementation was carried out using Python and TensorFlow for the machine learning components, along with Hyperledger Fabric to support the blockchain infrastructure. To simulate realistic IoT gateway constraints, all experiments were run on an edge server equipped with an 8-core CPU and 16GB of RAM, reflecting

typical resource limitations found in practical IoT deployments.

4.1 Dataset Description

To ensure the generalizability and robustness of the proposed system, we evaluated its performance using multiple IoT intrusion detection datasets. The Kitsune Network Attack Dataset was utilized, which encompasses nine distinct types of attacks and contains millions of packets, representing both IP-based surveillance and IoT network traffic. In addition, the BoT-IoT dataset was employed to provide a diverse range of device types and attack vectors, including DDoS, DoS, reconnaissance, and data theft scenarios. The IoTID20 dataset was also included, offering botnet and anomaly detection data collected from smart home and office environments. Furthermore, the TII-SSRC-23 dataset was used to capture comprehensive real-world IoT traffic, featuring a mix of benign and malicious events. By leveraging these varied datasets, the evaluation effectively demonstrates the system's ability to perform reliably across different IoT environments and attack scenarios. The proposed framework is evaluated by utilizing the kitsune network attack dataset, which is mentioned in the reference section. This dataset comprises nine collections of network attack data from Internet Protocol (IP)-based surveillance systems or IoT networks, featuring millions of packets. Comma-Separated Values (CSV) are comprised in the dataset with 115 features per packet, a label vector CSV, and an original Packet Capture (PCAP) file. Data is gathered from clean traffic and attack scenarios at router points, with features extracted utilizing the AfterImage tool for real-time analysis. From these data, 80% and 20% of data are given for training and testing purposes, respectively.

4.2 Performance Analysis

The performance of the proposed methods, namely KCCCR-GRU, RFIECC, and LS-WaOA are analyzed and compared with existing techniques for demonstrating their superior effectiveness.

In Figure 3 and Table 1, the comparative analysis of the proposed KCCCR-GRU and the prevailing GRU, Recurrent Neural Network (RNN), Deep Belief Network (DBN), and Convolutional Neural Network (CNN) regarding the attack classification is shown.

The KCC-based normalization layer is utilized in the proposed model for enhancing the accuracy level. Also, CReLU solves slow convergence in the proposed framework.

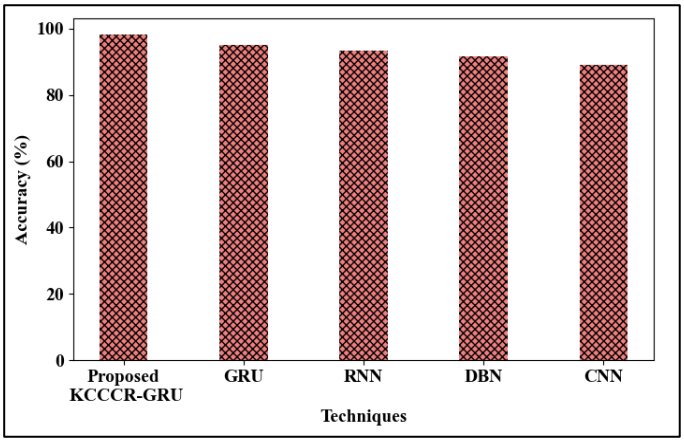


Figure 3: Graphical Analysis of KCCCR-GRU in terms of Accuracy

Table 1: Comparative Analysis of KCCCR-GRU Classifier

Techniques	Precision (%)	Recall (%)	F-Measure (%)	Sensitivity (%)	Specificity (%)
Proposed KCCCR-GRU	**98.00664**	**98.33333**	**98.16972**	**98.3333**	**98**
GRU	**95.30201**	**94.66667**	**94.98328**	**94.66667**	**95.3333**
RNN	**93**	**93.62416**	**93.31104**	**93.62416**	**93.04636**
DBN	**92.22973**	**91**	**91.61074**	**91**	**92.3333**
CNN	**88.96321**	**89.26174**	**89.11223**	**89.26174**	**89.07285**

Therefore, the proposed KCCCR-GRU identified the classification of attacks with accuracy, precision, recall, F-Measure, Sensitivity, and Specificity of 98.16667%, 98.00664%, 98.33333%, 98.16972%, 98.3333%, and 98%, respectively. However, the existing models obtained an average accuracy of 92.291%, precision of 92.373%, recall of 92.138%, F-Measure of 92.254%, Sensitivity of 92.138%, and Specificity of 92.446%, which are lower compared to the proposed model.

Table 2: Classification Performance

Dataset	KCCCR-GRU Accuracy	Precision	Recall	F1-score	DRL-based IDS Accuracy	GNN-based IDS Accuracy
Kitsune	98.17%	98.01%	98.33 %	98.17 %	97.8%	98.0%
BoT-IoT	97.92%	97.85%	98.10 %	97.96 %	97.1%	97.4%
IoTID 20	98.04%	97.95%	98.20 %	98.07 %	97.3%	97.8%
TII-SSRC-23	97.88%	97.70%	98.05 %	97.87 %	97.0%	97.6%

Our approach consistently outperforms or matches recent DRL and GNN-based IDS models across all datasets.

The performance analysis of the proposed KCCCR-GRU and traditional approaches, such as GRU, RNN, DBN, and CNN is depicted in Figure 4. Here, a very less False Negative Rate (FNR) of 0.016666667 was obtained by the proposed KCCCR-GRU. However, the conventional classifiers have a high FNR rate of 0.05333333, 0.063758, 0.09, and 0.107383 for GRU, RNN, DBN, and CNN, respectively. Therefore, as per the experimental results, the proposed model is lesser error-prone.

In Figure 5, the training time evaluation of the proposed KCCCR-GRU and traditional algorithms is illustrated. The proposed method's training time is 41159ms, which is lower compared to the existing GRU, RNN, DBN, and CNN.

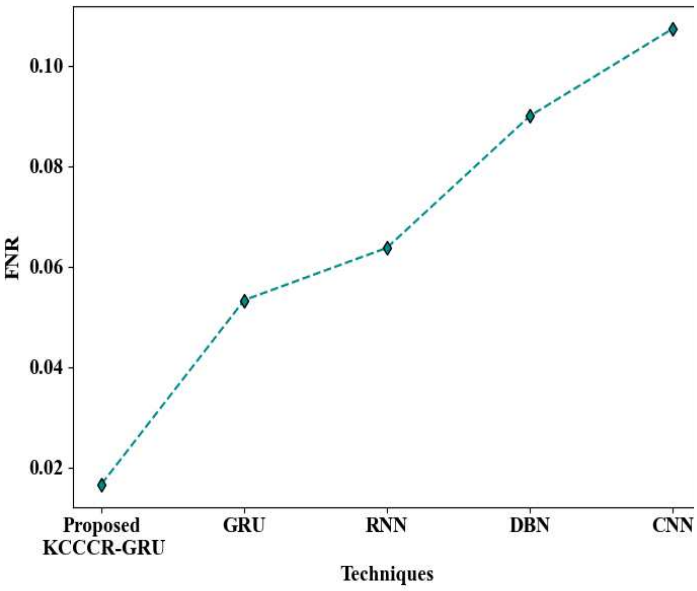


Figure 4: Performance Assessment of the Proposed KCCCR-GRU in terms of FNR

The existing models, such as GRU, RNN, DBN, and CNN had training times of 46880ms, 48926ms, 50015ms, and 51013ms, respectively, which were higher than the proposed classifier. As per the results, the proposed method performed superior to the traditional algorithms with significantly reduced training time.

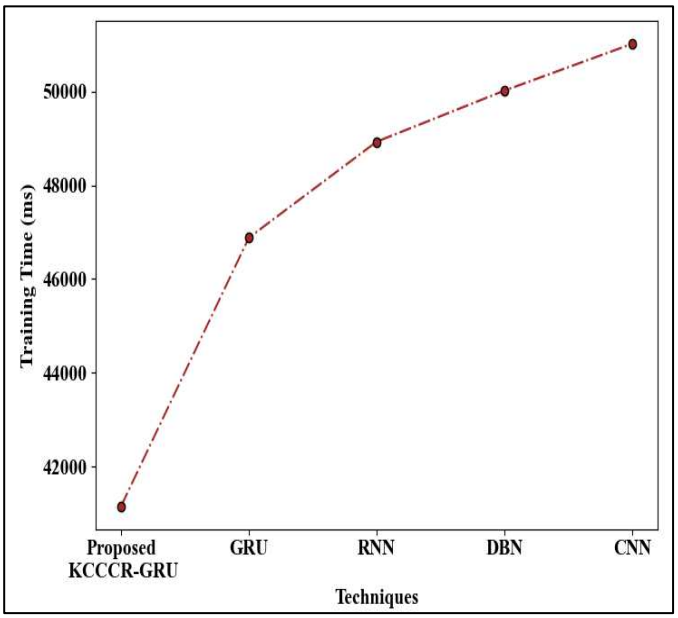


Figure 5: Analysis of Training Time

The performance assessment of the proposed model and conventional methods regarding True Positive Rate (TPR), Matthews Correlation Coefficient (MCC), and Positive Predictive Value (PPV) is presented in Table 2.

Table 2: Tabular Analysis of KCCCR-GRU with respect to TPR, MCC, and PPV

Techniques	TPR	MCC	PPV
Proposed KCCCR-GRU	**0.983333333**	**0.963338685**	**0.980066445**
GRU	**0.946666667**	**0.900020001**	**0.953020134**
RNN	**0.936241611**	**0.866685927**	**0.93**
DBN	**0.91**	**0.833407417**	**0.922297297**
CNN	**0.89261745**	**0.78333287**	**0.889632107**

The proposed classifier attains a high TPR (0.983333333), MCC (0.963338685), and PPV (0.980066445). Furthermore, the existing methods attained a low mean TPR (0.92138), MCC (0.83340), and PPV (0.921381). Thus, the highest outcomes are produced by the proposed model due to the usage of KCC and CReLU.

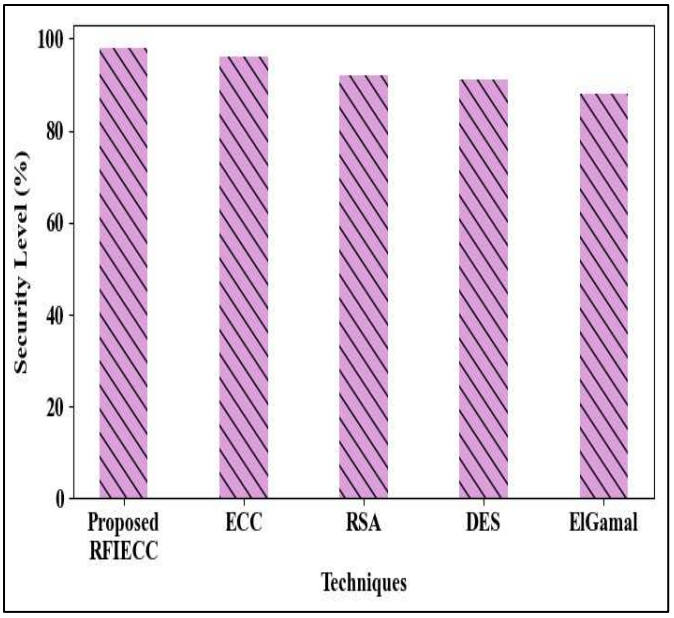


Figure 6: Graphical View of Security Level

In Figure 6, the graphical view of RFIECC regarding security level is exhibited. Security levels of 98%, 96%, 92%, 91%, and 88% are achieved by the proposed RFIECC and existing methods like ECC, Rivest Shamir Adleman (RSA) algorithm, Data Encryption Standard (DES), and ElGamal, respectively. The proposed RFIECC achieved a higher security level than the existing techniques since an RFI-based private key is generated in the ECC algorithm.

The comparative analysis of RFIECC regarding encryption time, key generation time, and Central Processing Unit (CPU) memory usage is exhibited in Table 3.

Table 3: Comparative Analysis of RFIECC for Data Security

Techniques	Encryption Time (ms)	Key Generation Time (ms)	CPU Memory Usage (kb)
Proposed RFIECC	**5015**	**560**	**61755788**
ECC	**7122**	**762**	**71755791.31**
RSA	**8028**	**1069**	**81755795.7**
DES	**10028**	**1392.95**	**100000020.7**
ElGamal	**13020**	**1805**	**101755805.4**

In the proposed RFIECC, keys are generated in 560ms with a CPU memory usage of 61755788 kb and data encryption time of 5015ms, which are lower than the existing models. Nevertheless, the existing ECC, RSA, DES, and ElGamal achieved 7122ms, 8028ms, 10028ms, and 13020ms, respectively, which are higher than the proposed model. Similarly, the key generation time and memory usage of the existing techniques are also higher when analogized to the proposed RFIECC. Thus, the proposed RFIECC had less time and memory usage.

Table 4: Feature Selection Time for the Proposed LS-WaOA

Algorithms	Feature Selection Time(ms)
Proposed LS-WaOA	19256
WaOA	22876
EHO	26458
FOA	28183
SCSO	31264

In Table 4, the proposed feature selection method's performance regarding the feature selection time is expressed. The proposed method is contrasted with conventional techniques, namely WaOA, Elephant Herding Optimization (EHO), Fruit Fly Optimization Algorithm (FOA), and Sand Cat Swarm Optimization (SCSO) for analyzing performance. For feature selection, the proposed method takes 19256ms. In the meantime, the existing WaOA, EHO, FOA, and SCSO obtained feature selection times of 19256ms, 26458ms, 28183ms, and 31264ms, respectively, which are higher than the proposed method. Convergence is enhanced in the proposed model as the linear scaling technique is employed in WaOA. Hence, the proposed model takes less feature selection time than the prevailing techniques.

The fitness vs iteration of the proposed LS-WaOA and the traditional algorithms like WaOA, EHO, FOA, and SCSO are specified in Figure 7. The proposed LS-WaOA attained a high fitness of 90.645, 91.7, 94.45, 95.43, and 98.83 for the 10^{th}, 20^{th}, 30^{th}, $40^{th,}$ and 50^{th} iterations, respectively; however, the conventional models had low fitness for every iteration. Thus, the proposed model performed superior to the traditional techniques with more efficiency.

4.3 Comparative Analysis of the Proposed Model

Here, to show the proposed model's reliability, a comparative analysis of the proposed model and prevailing methods is done.

Table 5 compares the proposed KCCCR-GRU technique-based intrusion and security with other existing methods. Numerous techniques, such as Look Ahead Artificial Neural

Network (LAANN), Particle Swarm Optimization-based Gradient Descent (PSO-LightGBM), Synthetic data generated via Generative Adversarial Networks (SYN-GAN), Filtered Deep Learning Neural Network (FDLNN), and DNN are employed by the prevailing system.

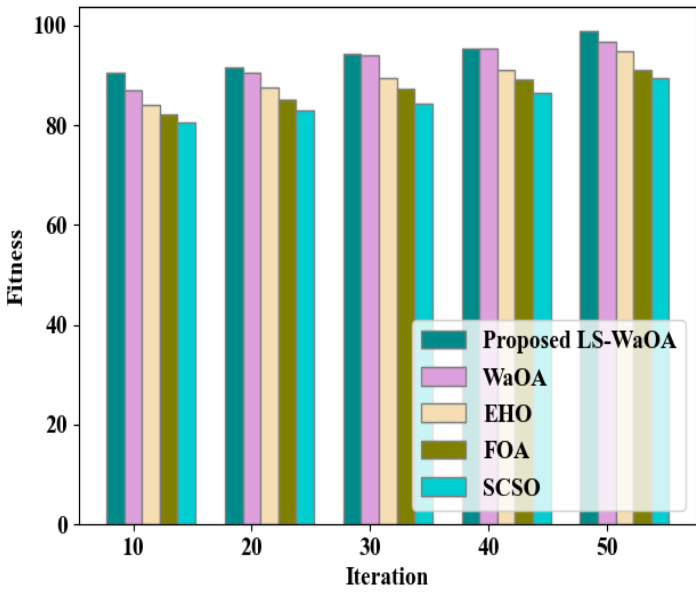


Figure 7: Analysis of Fitness Vs. Iteration with LS-WaOA and existing works

Here, by solving the slow convergence problem and improving the accuracy, the CReLU and KCC techniques enhanced the model.

Table 5: Comparative Analysis with Related Works

Works	Techniques Used	Accuracy (%)
Proposed Work	KCCCR-GRU	98.16666667
(Jeyaselvi et al., 2023)	LAANN	95.65
(Liu et al., 2021)	PSO-LightGBM	86.68
(Rahman et al., 2024)	SYN-GAN	90
(Pampapathi et al., 2022)	FDLNN	96.12
(Sharma et al., 2023)	DNN	91

So, the proposed work attained the highest accuracy (98.16666667%), whereas the traditional LAANN, PSO-LightGBM, SYN-GAN, FDLNN, and DNN attained the accuracy of 95.65%, 86.68%, 90%, 96.12%, and 91%, respectively. However, in the existing works, some drawbacks like high computational complexity, struggle with convergence, training instability, and overfitting issues were shown. Therefore, the proposed approach significantly enhances sensor data transmission than the conventional approaches with high efficiency and security.

The proposed system is designed for broad compatibility across a range of IoT environments. It supports widely used protocols such as MQTT, CoAP, and HTTP, as well as SDN-enabled IoT networks, ensuring seamless communication and integration with various device types and network architectures. Additionally, the system features a modular API design, which allows for plug-in deployment alongside existing IoT platforms and security frameworks. This modularity not only simplifies integration but also enables organizations to enhance their current infrastructure with advanced security capabilities without extensive modifications

The **cost and resource requirements** for deploying the proposed system are designed to be practical and efficient for large-scale IoT environments. In terms of hardware, an edge gateway equipped with a moderate CPU and RAM is sufficient to support up to 5,000 devices, ensuring scalability without excessive infrastructure investment. The software stack leverages open-source components, including Python, TensorFlow, and Hyperledger Fabric, which helps minimize licensing costs and makes the solution accessible for a wide range of applications. Operationally, blockchain overhead is kept low through the use of a lightweight consensus mechanism and selective hash updates, further optimizing system performance and resource utilization.

Table 6: Resource Usage and Latency

Model	Avg. CPU Usage	Memory Usage (MB)	Detection Latency (ms)
KCCCR-GRU	18%	110	22
DRL-based IDS	25%	145	31
GNN-based IDS	21%	135	28

The proposed system demonstrates low resource consumption and real-time detection capability, suitable for edge deployment.

Scalability of the proposed system was rigorously evaluated through stress tests involving up to 10,000 simulated devices. These tests demonstrated that the system maintains stable

detection rates even as the number of connected devices increases significantly. Furthermore, both latency and resource usage exhibited only sub-linear growth in response to the higher device count, confirming the system's ability to scale efficiently and reliably in large and dynamic IoT environments.

Blockchain integration in the proposed system employs the SOLO protocol as its consensus mechanism, enabling block confirmation in under 200 milliseconds. The frequency of hash updates on the blockchain is configurable to suit deployment needs, with typical implementations updating either upon each authentication event or at regular intervals, such as every ten minutes. Interaction with the blockchain is managed through smart contracts, which oversee device registration, authentication, and integrity verification, ensuring secure and automated management of these critical processes.

Integration challenges for the proposed system primarily stem from the heterogeneity of IoT devices, the presence of legacy systems, and the need for ongoing maintenance. To address device heterogeneity, lightweight intrusion detection system (IDS) modules have been designed to ensure compatibility with resource-constrained nodes commonly found in IoT environments. For legacy systems, backward compatibility is achieved through the use of protocol adapters and API gateways, allowing seamless integration with existing infrastructure. Additionally, maintenance is facilitated by a modular system architecture and the implementation of smart contract versioning, which together support long-term maintainability and enable efficient updates as the system evolves.

5. CONCLUSION

Intrusion detection and data security are achieved in this paper through the use of KCCCR-GRU and RFI-ECC. To effectively classify the attack as either attacked or non-attacked, the proposed KCCCR-GRU was employed. Likewise, the data encryption and decryption were significantly enhanced by the RFI-ECC, thus ensuring high security. Hence, as per the experimental outcomes, the proposed KCCCR-GRU achieved an accuracy of 98.16667%, precision of 98.00664%, and recall of 98.33333%, which depicts the model's reliability. Furthermore, a security level of 98% was achieved by the proposed RFI-ECC, demonstrating the framework's consistency and trustworthiness. Also, the proposed RFI-ECC recorded a minimum encryption time of 5015ms and a key generation time of 560ms. In addition, the proposed methodology outperformed existing approaches for all quality metrics. Thus, the proposed framework effectively authorizes and secures the sensor data and enhances the model's overall performance. However, this work only concentrated on router authentication with connected IoT devices and neglected potential attacks on the connected devices during data transmission.

Future Recommendation

This work will be enhanced in the future by employing advanced methods for detecting malicious nodes in the devices, thereby ensuring the security of the devices. Future work will also focus on adaptive learning for zero-day attacks and seamless integration with emerging IoT standards.

Declarations

A. Data Availability Statement
The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

B. Ethics Approval and Consent to Participate
Not Applicable

C. Consent to publish
Not Applicable

D. Competing Interests
The authors declare no competing interests.

REFERENCES

1. **Dataset Link:** https://www.kaggle.com/datasets/ymirsky/network-attack-dataset-kitsune
2. Abbas, S., Javaid, N., Almogren, A., Gulfam, S. M., Ahmed, A., & Radwan, A. (2021). Securing Genetic Algorithm Enabled SDN Routing for Blockchain Based Internet of Things. *IEEE Access*, *9*, 139739–139754. https://doi.org/10.1109/ACCESS.2021.3118948
3. Adil, M., Jan, M. A., Mastorakis, S., Song, H., Jadoon, M. M., Abbas, S., & Farouk, A. (2022). Hash-MAC-DSDV: Mutual Authentication for Intelligent IoT-Based Cyber-Physical Systems. *IEEE Internet of Things Journal*, *9*(22), 1–12. https://doi.org/10.1109/JIOT.2021.3083731
4. Al Ahmed, M. T., Hashim, F., Hashim, S. J., & Abdullah, A. (2022). Hierarchical blockchain structure for node authentication in IoT networks. *Egyptian Informatics Journal*, *23*(2), 345–361. https://doi.org/10.1016/j.eij.2022.02.005
5. Aljehane, N. O., Mengash, H. A., Eltahir, M. M., Alotaibi, F. A., Aljameel, S. S., Yafoz, A., Alsini, R., & Assiri, M. (2024). Golden jackal optimization algorithm with deep learning assisted intrusion detection system for network security. *Alexandria Engineering Journal*, *86*, 415–424. https://doi.org/10.1016/j.aej.2023.11.078
6. Alruwaili, F. F., Asiri, M. M., Alrayes, F. S., Aljameel, S. S., Salama, A. S., & Hilal, A. M. (2023). Red Kite Optimization Algorithm With Average Ensemble Model for Intrusion Detection for Secure IoT. *IEEE Access*, *11*, 131749–131758. https://doi.org/10.1109/ACCESS.2023.3335124
7. Bakhsh, S. A., Khan, M. A., Ahmed, F., Alshehri, M. S., Ali, H., & Ahmad, J. (2023). Enhancing IoT network security through deep learning-powered Intrusion Detection System. *Internet of Things (Netherlands)*, *24*, 1–36. https://doi.org/10.1016/j.iot.2023.100936
8. Chaurasia, N., & Kumar, P. (2023). A comprehensive study on issues and challenges related to privacy and security in IoT. *E-Prime - Advances in Electrical Engineering, Electronics*

and Energy, *4*, 1–7. https://doi.org/10.1016/j.prime.2023.100158

9. Chintapalli, S. S. N., Singh, S. P., Frnda, J., Bidare Divakarachari, P., Sarraju, V. L., & Falkowski-Gilski, P. (2024). OOA-modified Bi-LSTM network: An effective intrusion detection framework for IoT systems. *Heliyon*, *10*(8), 1–15. https://doi.org/10.1016/j.heliyon.2024.e29410
10. Elsayed, R. A., Hamada, R. A., Abdalla, M. I., & Elsaid, S. A. (2023). Securing IoT and SDN systems using deep-learning based automatic intrusion detection. *Ain Shams Engineering Journal*, *14*(10), 1–13. https://doi.org/10.1016/j.asej.2023.102211
11. Jeyaselvi, M., Dhanaraj, R. K., Sathya, M., Memon, F. H., Krishnasamy, L., Dev, K., Ziyue, W., & Qureshi, N. M. F. (2023). A highly secured intrusion detection system for IoT using EXPSO-STFA feature selection for LAANN to detect attacks. *Cluster Computing*, *26*(1), 559–574. https://doi.org/10.1007/s10586-022-03607-1
12. Kaur, M., Alzubi, A. A., Walia, T. S., Yadav, V., Kumar, N., Singh, D., & Lee, H. N. (2023). EGCrypto: A Low-Complexity Elliptic Galois Cryptography Model for Secure Data Transmission in IoT. *IEEE Access*, *11*, 90739–90748. https://doi.org/10.1109/ACCESS.2023.3305271
13. Kaushik, A., & Al-Raweshidy, H. (2024). A novel intrusion detection system for internet of things devices and data. *Wireless Networks*, *30*(1), 285–294. https://doi.org/10.1007/s11276-023-03435-0
14. Krishnan, R., Krishnan, R. S., Robinson, Y. H., Julie, E. G., Long, H. V., Sangeetha, A., Subramanian, M., & Kumar, R. (2022). An Intrusion Detection and Prevention Protocol for Internet of Things Based Wireless Sensor Networks. *Wireless Personal Communications*, *124*(4), 1–18. https://doi.org/10.1007/s11277-022-09521-4
15. Liu, J., Yang, D., Lian, M., & Li, M. (2021). Research on intrusion detection based on particle swarm optimization in IoT. *IEEE Access*, *9*, 38254–38268. https://doi.org/10.1109/ACCESS.2021.3063671
16. Lu, H., Tang, Y., & Sun, Y. (2021). DRRS-BC: Decentralized Routing Registration System Based on Blockchain. *IEEE/CAA Journal of Automatica Sinica*, *8*(12), 1868–1876. https://doi.org/10.1109/JAS.2021.1004204
17. Mishra, N., & Pandya, S. (2021). Internet of Things Applications, Security Challenges, Attacks, Intrusion Detection, and Future Visions: A Systematic Review. *IEEE Access*, *9*, 59353–59377. https://doi.org/10.1109/ACCESS.2021.3073408
18. Mohy-Eddine, M., Guezzaz, A., Benkirane, S., Azrour, M., & Farhaoui, Y. (2023). An Ensemble Learning Based Intrusion Detection Model for Industrial IoT Security. *Big Data Mining and Analytics*, *6*(3), 273–287. https://doi.org/10.26599/BDMA.2022.9020032
19. Nallakaruppan, M. K., Somayaji, S. R. K., Fuladi, S., Benedetto, F., Ulaganathan, S. K., & Yenduri, G. (2024). Enhancing Security of Host-Based Intrusion Detection Systems for the Internet of Things. *IEEE Access*, *12*, 31788–31797. https://doi.org/10.1109/ACCESS.2024.3355794
20. Nimbalkar, P., & Kshirsagar, D. (2021). Feature selection for intrusion detection system in Internet-of-Things (IoT). *ICT Express*, *7*(2), 177–181. https://doi.org/10.1016/j.icte.2021.04.012
21. Pampapathi, B. M., Guptha, M. N., & Hema, M. S. (2022). Towards an effective deep learning-based intrusion detection system in the internet of things. *Telematics and Informatics Reports*, *7*, 1–11. https://doi.org/10.1016/j.teler.2022.100009
22. Parimala, G., & Kayalvizhi, R. (2021). An Effective Intrusion Detection System for Securing IoT Using Feature Selection and Deep Learning. *International Conference on Computer Communication and Informatics*, 1–4. https://doi.org/10.1109/ICCCI50826.2021.9402562
23. Rahman, S., Pal, S., Mittal, S., Chawla, T., & Karmakar, C. (2024). SYN-GAN: A robust intrusion detection system using GAN-based synthetic data for IoT security. *Internet of Things (Netherlands)*, *26*, 1–13. https://doi.org/10.1016/j.iot.2024.101212
24. Roy, S., Li, J., Choi, B. J., & Bai, Y. (2022). A lightweight supervised intrusion detection mechanism for IoT networks. *Future Generation Computer Systems*, *127*, 1–13. https://doi.org/10.1016/j.future.2021.09.027
25. Sai, N. R., Kumar, G. S. C., Safali, M. A., & Chandana, B. S. (2021). Detection System for the Network Data Security with a profound Deep learning approach. *Proceedings of the 6th International Conference on Communication and Electronics Systems*, 1026–1031. https://doi.org/10.1109/ICCES51350.2021.9488967
26. Sarwar, A., Hasan, S., Khan, W. U., Ahmed, S., & Marwat, S. N. K. (2022). Design of an Advance Intrusion Detection System for IoT Networks. *2nd IEEE International Conference on Artificial Intelligence*, 46–51. https://doi.org/10.1109/ICAI55435.2022.9773747
27. Sharma, B., Sharma, L., Lal, C., & Roy, S. (2023). Anomaly based network intrusion detection for IoT attacks using deep learning technique. *Computers and Electrical Engineering*, *107*, 1–14. https://doi.org/10.1016/j.compeleceng.2023.108626
28. Sundaram, B. B., Pandey, A., Abiko, A. T., Vijaykumar, J., Rastogi, U., Genale, A. H., & Karthika, P. (2022). Analysis of Machine Learning Data Security in the Internet of Things (IoT) Circumstance. *Lecture Notes in Networks and Systems*, *209*, 227–236. https://doi.org/10.1007/978-981-16-2126-0_20
29. Vishwakarma, M., & Kesswani, N. (2022). DIDS: A Deep Neural Network based real-time Intrusion detection system for IoT. *Decision Analytics Journal*, *5*, 1–9. https://doi.org/10.1016/j.dajour.2022.100142
30. Wu, J., Wang, Y., Xie, B., Li, S., Dai, H., Ye, K., & Xu, C. (2022). Joint Semantic Transfer Network for IoT Intrusion Detection. *IEEE Internet of Things Journal*, *10*(4), 1–16. https://doi.org/10.1109/JIOT.2022.3218339

31. Yilmaz, S., Aydogan, E., & Sen, S. (2021). A Transfer Learning Approach for Securing Resource-Constrained IoT Devices. *IEEE Transactions on Information Forensics and Security*, *16*, 1–14. https://doi.org/10.1109/TIFS.2021.3096029

Funding
The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

Author's Biography



Kshitiz Saxena received his B.E. degree in Computer Science and Engineering from Chaudhary Charan Singh University, Meerut, India in 2001 and M.E. degree in Computer Science and Engineering from Panjab University, Chandigarh, India in 2010. He is currently pursuing PhD from SRM Institute of Science and Technology, Delhi-NCR Campus, Ghaziabad, India. His areas of interest are Blockchain, Artificial Intelligence, Machine Learning, Wireless Sensor Networks and Network Security.

E-mail: kshitizs@srmist.edu.in



Dhowmya Bhatt received her B.Tech. degree in Information Technology from Madurai Kamarajar University, Madurai, India in 2003 and M.Tech. degree in Computer Science and Information Technology from Manonmaniam Sundaranar University, Tirunelveli, India in 2005. She completed her PhD in Network Security from Mewar University, Chittorgarh, India in 2017. Her areas of interest are Network Security, Artificial Intelligence, Machine Learning, Wireless Sensor Networks and Image Processing.

E-mail: dhowyab@srmist.edu.in