SN Computer Science (2024) 5:394
https://doi.org/10.1007/s42979-024-02725-4

ORIGINAL RESEARCH

Improved Adaptive Spiral Seagull Optimizer for Intrusion Detection and Mitigation in Wireless Sensor Network

Swathi Darla[1] · C. Naveena[2]

Received: 9 October 2023 / Accepted: 15 February 2024

Abstract
A system that leverages blockchain technology to protect network data and provide tamper-proof administration, privacy, and intrusion detection for sensor networks. This blockchain technology takes advantage of the decentralised and open nature of blockchain technology to address the issues of security risks and data privacy concerns in sensor networks. This research put out a cutting-edge method for sensor network intrusion detection and mitigation combining deep reinforcement learning (DRL) and blockchain technology. There are several stages to the suggested method for wireless sensor network intrusion detection. First, using data cleaning and transformation techniques, the datasets (NSL-KDD, CSE-CIC-IDS2018) are gathered and pre-processed. For the intrusion detection task, pertinent characteristics are chosen, including statistical features, protocol-based features, higher-order statistical features (HOS), and Pearson Correlation Based Principal Component Analysis (PC-PCA). To prevent unauthorised access, the best features are encrypted with the latest AES. Following storage in the blockchain network, the encrypted data is guaranteed for integrity, immutability, and transparency. The chosen ideal features are input to the multi-layer perceptron's (MLP) recurrent neural network (RNN) during the intrusion detection phase. To increase detection accuracy, the weight function of the RNN is adjusted using the Adaptive Spiral Seagull Optimisation (ASSO). The blockchain network takes the appropriate steps to mitigate the attack (BAIT) if an intrusion is discovered. The A* algorithm determines the shortest path for data transmission, and the gateway node uses that path to transfer the encrypted data to the destination node. The destination node receives the encrypted data, decrypts it using the proper decryption method, and then processes it for various applications. Python is used to implement the suggested model.

Keywords Deep reinforcement learning · Blockchain · I-RFE · MLP · RNN · A* algorithm · ASSO

Introduction

By providing decentralised and transparent verification of network activities, a blockchain-based solution can improve the security of sensor networks. This can offer a strong defence against cyberattacks by helping to detect and prevent intrusions. A trustworthy and impenetrable technique for monitoring and securing network data can be provided by combining blockchain technology with sensor networks, improving the performance and resilience of the entire system. By resolving concerns with privacy, security, and third-party dependencies, the development of more dependable and secure Internet of Things (IoT) applications has been made possible [5]. Automated threat vector detection and response creation can reduce the requirement for human interaction in the threat management process [6]. By limiting data manipulation and disclosure, the use of blockchain technology to record communication agent operations improves system security [7]. Blockchain technology is utilised in a novel method for secure authentication in wireless sensor networks to increase effectiveness and security [8]. Blockchain has become a favoured digital storage solution

This article is part of the topical collection "Advances in Computational Approaches for Artificial Intelligence, Image Processing, IoT and Cloud Applications" guest edited by Bhanu Prakash K N and M. Shivakumar.

✉ Swathi Darla
swathidarla.rvitm@rvei.edu.in

[1] Department of ISE, RV Institute of Technology and Management, Visvesvaraya Technological University, Belagavi 590018, India

[2] Department of CSE, SJB Institute of Technology, Visvesvaraya Technological University, Belagavi 590018, India

Published online: 04 April 2024

for IoT-based WSNs due to its safe, trustworthy, and transparent nature [9].

Collaborative intrusion detection networks (CIDNs) are widely used by organisations to enhance detection capabilities and safeguard their resources from online dangers [10]. For better routing security, a WSN routing method that makes use of blockchain and reinforcement learning techniques has been developed [11, 30, 31]. Decentralised and distributed analytics can be made possible by blockchain technology while maintaining data privacy and process reliability [12]. Fog and mobile edge computing, along with an SDN and blockchain-based decentralised security architecture, can improve IoT network security and make it possible for smart city applications to effectively detect attacks [13]. SDN can enable effective traffic anomaly detection techniques, but their design and implementation face additional difficulties [14, 32].

An optimized and secure IoT framework that leverages SDN and blockchain technologies can efficiently manage resource utilization and enable secure network communication in smart networks [15]. As security attacks on IoT systems become more complex and diverse, it is crucial to analyze techniques specific to the IoT context for identifying, preventing, and detecting novel attacks [16, 33]. A real-time smart contract intrusion detection method that successfully identifies attacks against smart contracts [17]. A new trust-based range-free method that uses blockchain technology for secure localization in adversarial WSNs is part of Deep Coin, a new energy framework that makes use of deep learning and blockchain technologies to improve smart grid security and guard against cyberattacks [18, 34]. Based on a number of variables, including reputation, mobility, residual energy, and neighbour node list, the algorithm assesses the trust levels of beacon nodes [19].

This study's major contribution is exemplified below:

- To extract the necessary features, the proposed approach utilizes the Improved Principal Component Analysis (I-PCA) method, which is a novel technique introduced in this study.
- To protect the extracted features from unauthorized access, a new Advanced Encryption Standard (AES) is used to encrypt and decrypt them.
- To develop a more accurate intrusion detection model, a Deep Reinforcement Learning (DRL) approach based on MLP and RNN is proposed. The weight function of the RNN model is fine-tuned using the novel ASSO to further enhance detection accuracy.
- It is crucial to act quickly in the event that an intrusion is discovered in order to maintain the security and integrity of the blockchain network. In these circumstances, the blockchain network launches the appropriate countermeasures, or BAIT, to neutralise the assault and lessen its effects.
- The gateway node uses the A* algorithm to find the shortest path for data transmission. This method determines the shortest path between the source and destination nodes, making data transfer effective and ideal.

The remaining sections of this essay are structured as follows: The literature overview on intrusion detection and mitigation in sensor network-based blockchain systems is covered in "Literature Review", and the proposed mechanism is presented in "Proposed Methodology". The experimental findings are described in "Result and Discussion". This research is concluded in "Conclusion".

Literature Review

In 2018, Sun et al. [20] have proposed an intrusion detection model, WSN-NSA, based on an improved V-detector algorithm for Wireless Sensor Networks (WSN). The WSN-NSA aimed to overcome the problem of resource constraints by establishing a three-level detection mechanism and modifying the V-detector algorithm. PCA was used was used to reduce detect features.

In 2018, Qu et al. [21] have proposed a knowledge-based intrusion detection strategy (KBIDS) to bridge the gap. First, an unsupervised learning method called the Mean Shift Clustering Algorithm (MSCA) was employed to separate the normal environment from ill-defined anomalous patterns that characterise the behaviour of a WSN under attack. Following that, a support vector machine was used to increase the space between abnormal and normal features in order to more accurately detect aberrant features. In order to allow the system to co-evolve with network changes, a technique for feature update was finally adopted to take into account network dynamics.

In 2021, Safaldin et al. [22] have suggested minimising false alarm rates and the number of characteristics produced by IDSs in order to improve the accuracy and detection rate of intrusion detection and shorten processing times in the WSN environment. Using actual data, the researchers assessed the performance of their suggested strategy and showed that it was effective at enhancing intrusion detection in WSNs.

In 2017, Jin et al. [23] a multi-agent model architecture for intrusion detection in both cluster heads and standard sensor nodes has been proposed. The Mahalanobis distance theory was employed by the system to determine normalcy and create typical node trust qualities. The Beta distribution and a tolerance factor were used to generate and update trust levels, which enabled successful node intrusion detection.

In 2020, Miranda et al. [24] have presented a software-defined security framework that incorporated cooperative anomaly detection and intrusion prevention. To offer a simple intrusion prevention system on the data plane, an IPS-based authentication mechanism was developed. A cost-effective intrusion detection solution close to the data plane was provided by utilising a cooperative anomaly detection system. The control plane used a Smart Monitoring System (SMS) to correlate the true positive alarms generated by the sensor nodes at the network edge.

In 2017, Wang et al. [25] have suggested a trust-based intrusion detection system for wireless sensor networks at the protocol layer. The trustworthiness of a sensor node was assessed in light of important parameter variations at each protocol layer. The parameters of the associated protocol layers were necessarily affected by attacks launched at different protocol layers.

In 2019, Han et al. [26] have suggested a game theory-based and autoregressive intrusion detection approach to boost wireless sensor network efficiency and control energy use. The goal was to fend off security risks and extend the network's life. When compared to conventional IDS, which uses more energy and shortens network lifetime, the suggested Intrusion Detection System (IDS) was able to identify a larger range of attacks.

In 2015, Butun et al. [27] have unveiled an IDS framework for hierarchical WSNs based on multi-level clustering. The framework used two different IDS approaches: "U-IDS" to identify abnormal behaviour in cluster leaders and "D-IDS" to identify abnormal behaviour in subordinate nodes. The usefulness of the suggested framework in identifying intrusions at various levels was determined by simulated evaluation, which produced the desired results.

In 2019, Alqahtani et al. [28] have suggested the use of a model termed GXGBoost to find intrusion attempts in wireless sensor networks. It combined a genetic algorithm and an extreme gradient boosting classifier to enhance the effectiveness of conventional models in spotting attacks from minority classes in highly unbalanced data traffic.

In 2020, Almomani and Alromi [29] have pioneered the use of Software Engineering (SE) methods in the development of complicated and important systems, such as security and networking systems like Wireless Sensor Networks (WSNs). As WSNs and their applications were present in several military and civilian systems, security threats were drawn to them. To address the rising hazards and system vulnerabilities of WSNs, researchers and developers proposed a variety of security solutions, including software-based Intrusion Detection Systems (IDSs).

Problem Statement

Author	Aim/objective	Drawback/Disadvantages
Sun et al. [20]	To propose an intrusion detection model for wireless sensor networks with an improved V-detector algorithm	No mention of performance evaluation results or comparison with existing intrusion detection models
Miranda et al. [24]	To propose a collaborative security framework combining intrusion prevention and anomaly detection in software-defined wireless sensor networks	The framework's effectiveness is dependent on the accuracy of anomaly detection, which may be impacted by the network's topology and data traffic
Wang et al. [25]	Evaluate the trust value of sensor nodes based on deviations in parameters at each protocol layer to detect intrusions	It may require significant computational resources to continuously monitor and evaluate the trust values of each sensor node
Han et al. [26]	To propose an energy-efficient intrusion detection model based on game theory and autoregressive model	limitations in detecting unknown or zero-day attacks
Alqahtani et al. [28]	A wireless sensor network's highly unbalanced data stream is being examined to find small attack classes	The effectiveness of the model depends on the quality and quantity of the training dataset
Almomani and Alromi [29]	To incorporate software engineering techniques in the creation of effective intrusion detection systems for wireless sensor networks	The article does not provide a specific drawback but rather emphasizes the importance of integrating software engineering processes in developing efficient intrusion detection systems

Proposed Methodology

A blockchain system can enhance the security of intrusion detection and mitigation in sensor networks by providing a tamper-resistant and decentralized ledger to store and manage network data. This adds an additional layer of security to the process of recognising and addressing security issues in the network by making it harder for attackers to influence or compromise the system. In this paper, we suggested a deep reinforcement learning-based and blockchain-based system for intrusion detection and mitigation in sensor networks. The system aims to enhance the security of sensor networks by detecting and preventing potential attacks. The proposed model includes ten major phases, and they are discussed below.

Step 1: **Data Collection:** Data cleaning and transformation, consistent dataset combining, and the selection of pertinent features for the intrusion detection task are all steps in the process of gathering and pre-processing datasets (NSL-KDD, CSE-CIC-IDS2018) for attack detection in WSN.

Step 2: **Pre-Processing:** Collected raw data i.e., related to intrusion detection in wireless sensor networks, denoted as R^{inp} pre-processed via data cleaning and min–max normalization approach. The data acquired after data cleaning and normalization is the pre-processed data, and it is pointed as R^{pre}.

Step 3: **Feature Extraction:** Then, from the pre-processed data R^{pre}, the features such as Improved Principal Component Analysis (I-PCA), statistical features, protocol-based features include the number of packets and bytes transmitted for each protocol, Higher-order statistical features (HOS) such as skewness, kurtosis and correlation are extracted. These extracted features are denoted as F.

Step 4: **Feature Encryption:** In order to prevent unauthorised access, the best features f from the extorted features F are encrypted with the new AES algorithm.

Step 5: **Blockchain-based Data Storage:** The gateway node stores the encrypted data in the blockchain network, which ensures the integrity, immutability, and transparency of the data.

Step 6: **DRL Based Intrusion Detection:** The multi-layer perceptron's (MLP), recurrent neural network (RNN) receives the chosen optimum features f as an input during the intrusion detection phase. The weight function of RNN is being fine-tuned with a novel algorithm called ASSO to improve the accuracy of detection.

Step 7: **Intrusion Mitigation:** If an intrusion is detected, the blockchain network takes necessary actions to mitigate the attack (BAIT).

Step 8: **Shortest Path Calculation:** To find the shortest path for data transmission, the gateway node uses the technique A* algorithm to calculate the shortest path between the source and destination nodes.

Step 9: **Data Transmission:** Once the shortest path is calculated, the gateway node sends the encrypted data to the destination node through the selected path.

Step 10: **Decryption and Processing:** The destination node receives the encrypted data and decrypts it using the appropriate decryption technique. The decrypted data is then processed and used for various applications. Figure 1 shows the overall proposed architecture.

Data Collection

The CSE-CIC-IDS2018 dataset1 [1] is sourced from logs of the University of New Brunswick's servers, containing

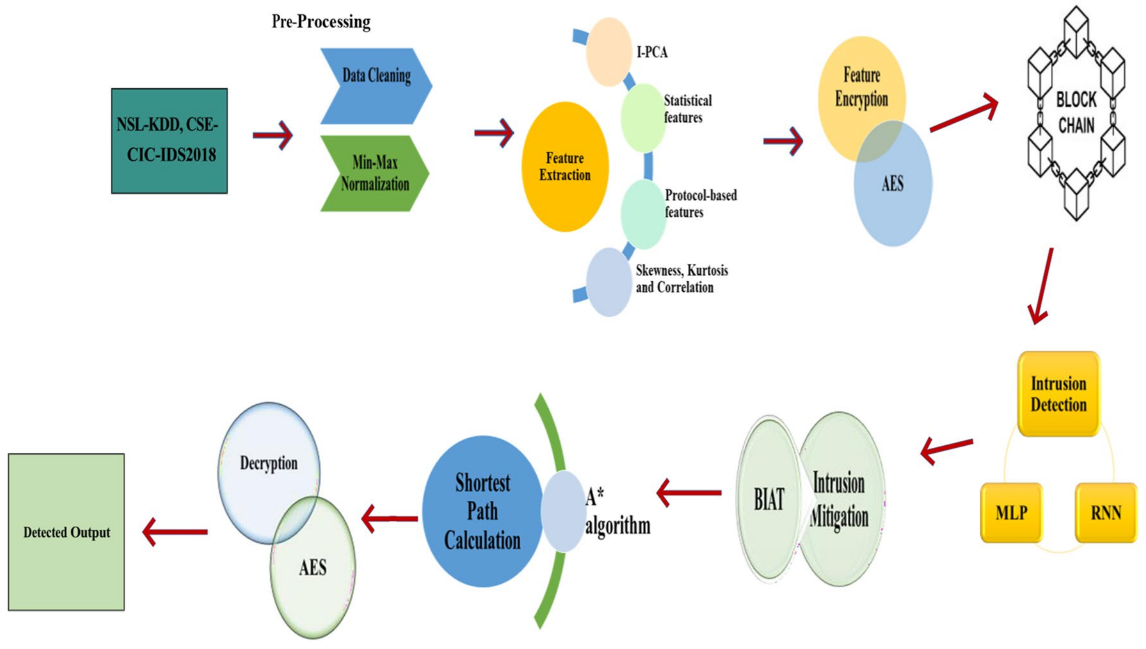


Fig. 1 Overall proposed architecture

various DoS attacks. The dataset is divided into multiple files based on date and it contains 80 columns namely flowID, sourceIP, destinationIP,source port, destination port, Statistical and other protocol based features [1, 3]. Among these columns the Label column being the most important for determining if the packets are malicious or not has been considered. CSE-CIC-IDS2018 dataset contains 16,000,000 instances collected over a period of ten days. It is the most updated recent dataset for intrusion detection, publicly available, and covers a wide range of attack types.

The NSL-KDD dataset2 [2] contains various files in both ARFF and CSV formats, including the full train set with binary and attack-type labels, a 20% subset of the train set, the full test set with binary and attack-type labels, and subsets of the test set that exclude difficult records. This dataset is an improvement over the original KDD dataset, as it includes no redundant or duplicate records, has a proportional selection of records from each difficulty level group, and has a reasonable number of records in the train and test sets. There are 41 features in dataset namely protocol type, service, flag, src-bytes, destination-bytes and so on [4]. These factors make it efficient to evaluate different machine-learning techniques consistently and comparably. The collected raw data is passed as an input to the pre-processing phase. Table 1 displays the datasets used in research work with its attack types and features.

Pre-Processing

Data cleaning and min–max normalization techniques were utilized in the pre-processing stage of this research work.

Data Cleaning

Data cleaning is an essential step in preparing data for intrusion detection and mitigation in sensor networks using a blockchain-based system. It involves identifying and correcting errors or inconsistencies in the data, such as missing or duplicate values, outliers, and irrelevant information. By performing data cleaning, we can ensure that the data is accurate, complete, and consistent, which can help to improve the performance of the intrusion detection system.

Min–Max Normalization

Min–max normalization is a prevalent pre-processing technique in intrusion detection for sensor networks based on blockchain systems. This method scales numerical features to a 0–1 range, aiding comparisons. Applied to sensor data, it ensures consistency for analysis, enhancing intrusion detection accuracy when integrated with blockchain, adding security and reliability.

Feature Extraction

In this research work, a new feature extraction method called Improved Principal Component Analysis (I-PCA) is proposed. The features extracted using I-PCA include Statistical features, Protocol-based features such as the number of packets and bytes transmitted for each protocol, and Higher-order statistical features (HOS) such as Skewness, Kurtosis, and Correlation.

Pearson Correlation Based Principal Component Analysis (PC-PCA)

Principal Component Analysis (PCA) is a statistical technique for feature extraction, which involves transforming a set of correlated variables into a new set of uncorrelated variables called principal components. PCA is used to identify the most important patterns and relationships among the original variables, and to represent them in a simplified manner. The structure of PC-PCA pseudocode is shown in Algorithm 1.

Algorithm 1 PC-PCA

```
1. Function PCA(X)
2. Compute PC based PCA matrix of pre-processed data X
   $C = \frac{1}{n-1}(X - \overline{X})^T (X - \overline{X})$ ; $X$ is the pre-processed data, and $\overline{X}$ defines the mean of $X$
3. Compute Eigen values $\lambda_i$ and Eigen vectors $V_i$ of $C$
   $CV_i = \lambda_i V_i$
4. Estimating high valued Eigen vectors
      i.  Sort Eigen vectors in descending order
      ii. Choose top k Eigen vectors based on pre-determined threshold
5. Form projection matrix $P$ using selection of top $k$ Eigen vectors, $P = eigvecs[0{:}k]$
6. End function
```

Table 1 Description of Datasets used in experiment

Dataset name	No. of features	Attack types	Instances
CSE-CIC-IDS2018 [1]	80	Distributed Denial of Service(DDoS), Denial of service(DoS), Bruteforce, Botnet, Infiltration, web attack	16,000,000
NSL-KDD [2]	41	Denial of Service (DoS), User to Root (U2R), Root to Local(R2L), Probe	1,25,973

Statistical Features

Mean The mean is the average of the numbers provided, and it is computed by dividing the total number of values by the sum of the numbers provided**.** This is mathematically shown in Eq. (1).

$$Mean = \frac{Sum\ of\ all\ observations\ of\ R^{pre}}{Total\ number\ of\ observations\ in\ R^{pre}} \tag{1}$$

Standard Deviation A measurement of how much the data deviates from the mean is referred to as the standard deviation (SD). Whenever the standard deviation is low, all of the values are close to the mean; when it is high, they are widely spread. This is mathematically shown in Eq. (2).

$$\mathrm{SD}(\sigma) = \sqrt{\frac{\sum (R^{pre} - \mu)^2}{N}} \tag{2}$$

Where R^{pre} is the input value (pre-processed data, μ is the mean and N is the total number of elements.

Variance Analytically, the variance of a data set is the measure of numerical variation. Variance, in particular, determines how far off each integer in the set is from the mean and, consequently, from the other numbers in the set. This is mathematically shown in Eq. (3).

$$\text{Variance} = \frac{\sum (R^{pre} - \mu)^2}{N} \tag{3}$$

Protocol-Based Features: Packet and Byte Count Per Protocol

Protocol-based features, such as the number of packets and bytes transmitted for each protocol, can be used as inputs for intrusion detection and mitigation systems in sensor networks based on blockchain technology. The number of packets and bytes transmitted for each protocol are some of the protocol-based features that can be used to analyze network traffic. Both the NSL-KDD and CSE-CIC-IDS2018 datasets contain protocol-based features among others. By analyzing these protocol-based features, network intrusion detection systems can identify anomalies in network traffic that may indicate potential security threats. By incorporating protocol-based features into intrusion detection and mitigation systems in sensor networks based on blockchain technology, organizations can improve the security of their network infrastructure and protect against potential cyber-attacks.

Higher-Order Statistical Features (HOS)

Higher-order statistical features (HOS) are statistical measurements that provide additional information about the distribution of a dataset beyond the mean and standard deviation. Skewness, Kurtosis, and Correlation are three examples of HOS features.

Skewness Skewness is used to describe the distribution of data and to identify any asymmetries in the data as shown in Eq. (4).

$$skewness = \frac{3(Mean - Median)}{StandardDeviation} \tag{4}$$

Kurtosis Kurtosis is a statistical measure that describes the shape of the distribution of a set of values. It is a measure of the "peakedness" or "flatness" of the data compared to a normal distribution. A normal distribution has a kurtosis of zero, while positive kurtosis indicates a more peaked distribution and negative kurtosis indicates a flatter distribution. This is mathematically shown in Eq. (5).

$$Kurtosis(highorder) = \frac{4^{th}Moment}{4^{th}Moment^2} \tag{5}$$

Correlation The correlation feature demonstrates the co-occurrence matrix's grey level values' linear dependence. It shows how closely linked a reference pixel is to its neighbor; 0 indicates no relationship and 1 indicates a perfect one as per Eq. (6).

$$g_5 = \sum_{a=0}^{N_e-1} \sum_{b=0}^{N_e-1} P_c, \theta(a,b) \frac{(a - \mu_x) - (b - \mu_y)}{\sigma_x \sigma_y} \tag{6}$$

The extracted features are passed as an input to the feature selection phase.

Feature Encryption

In this research work, the new Advanced Encryption Standard (AES) approach is used for feature selection.

Advanced Encryption Standard

Encryption/Decryption Using AES

The Advanced Encryption Standard (AES) is a widely used encryption algorithm that is used to secure sensitive data such as credit card numbers, passwords, and other personal information. AES is a symmetric-key encryption algorithm, which means that the same key is used for both encryption and decryption of data. The algorithm uses a block cipher, which operates on fixed-size blocks of data. The standard AES block size is 128 bits, but it can also be 192 or 256 bits. AES uses a series of mathematical operations, including substitution, permutation, and bitwise operations, to transform the input data and key into encrypted output data. The strength of AES lies in the complexity of these operations, which make it very difficult for attackers to decipher the encrypted data without the correct key. The structure of AES pseudocode is shown in Algorithm 2.

Blockchain Based Data Storage

Blockchain is a decentralized database system that facilitates peer-to-peer transactions by storing data in distinct structures called blocks. The blockchain structure in network model is described in Fig. 2. The block's primary role is to store, via cryptographic hashing, a record of transactions that have already been confirmed. Each block represents a compilation of transactions that have been executed during a specific time frame. The number of transactions contained within each block may vary. Within the blockchain, the connection between blocks is established using a cryptographic mechanism. This involves the computation of a hash value by combining the signature of the current block with the hash value of the preceding block. This feature makes the hash function efficient and date integrity will be maintained effectively.

The blockchain is established and the references to the preceding block are maintained through the repetition of this procedure for every block. The model then utilizes the data acquired through the activities of a blockchain to identify irregularities in the fundamental operation of the network.

Algorithm 2 AES

```
Input: input file $S^i$
Output: plain text $U^i$
Initialize
Key← $p$
Encrypt $S^i$ using AES public key
Encrypted data are transmitted to blockchain
Output (cipher text)
Decrypt the encrypted data using public key
Return original data $U^i$
Output (plaintext)
```

In this work, blockchain is used for storage. When utilizing blockchain solely for storage, its decentralized and tamper-resistant characteristics guarantee data integrity. The blockchain's immutability and transparency ensure secure and reliable data storage without additional functionalities. The gateway node plays a crucial role in this process by encrypting the data before storing it on the blockchain network, ensuring that the data is protected from unauthorized access.

Deep Reinforcement Learning-Based Intrusion Detection

In this research work, two deep learning algorithms, namely Multi-Layer Perceptron (MLP) and Recurrent Neural Network (RNN), are used for Deep Reinforcement Learning-Based Intrusion Detection. This involves combining the strengths of both MLP and RNN to enhance the accuracy and effectiveness of intrusion detection. The DRL framework will be trained and tested using these neural network structures to improve security measures and detect anomalies or malicious activities within a network.

Multi-Layer Perceptron's (MLP)

MLP stands for Multi-layer Perceptron, which is a type of artificial neural network (ANN). An MLP consists of an input layer, one or more hidden layers, and an output layer. Each layer contains a number of artificial neurons (also known as nodes) that are connected to the neurons in the previous and next layers via weighted connections. One feed-forward neural network classification technique is the multilayer perceptron. There are several layers in it. The linearly separable issues in Single-Layer Perception (SLP) can be solved, but the nonlinear problems cannot. MLP is utilized to solve these difficult issues.

The network's weights are randomly set before training. The neurons then take in information from the training set, which in this case is a set of tuples (n_1, n_2, e). The input to the network is n_1 and n_2, and its expected output is e. If u represents the actual output, the neuron's output is relied on the weighted total of all of its neurons and is shown as Eq. (7).

$$u = n_1 w_1 + n_2 w_2 \tag{7}$$

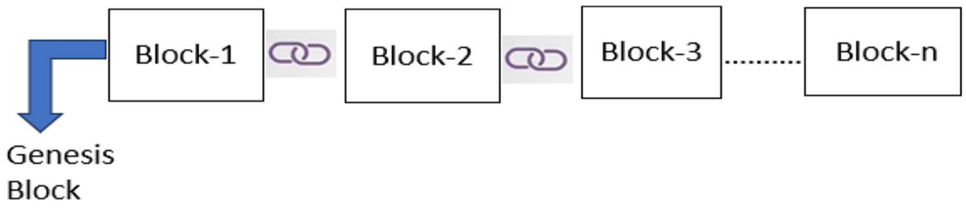


Fig. 2 Blockchain architecture in network model [34]

A single hidden layer with a nonlinear activation function makes up the network. The network's output can be stated as Eq. (8).

$$O = f(v) = M \div \emptyset(Rv + l) + s \tag{8}$$

where s represents the input is vector and O represents the output vector. The weight matrix and first layer's bias vector are R and l, respectively. M stands for the weight matrix, and s for the second layer's bias vector. The nonlinear element is $\emptyset$. The output is not visible in the output but is related to the inputs of additional neurons in the hidden layer.

Recurrent Neural Network (RNN)

RNN is a type of neural network that is designed to process sequential data, such as time series data or natural language. RNNs have a "memory" component, called a hidden state, that allows them to remember information from previous time steps, which is useful for tasks such as language modelling or speech recognition. RNNs can be unrolled over time to show the connections between the hidden state at each time step, forming a structure called a recurrent neural network diagram. Common types of RNNs include the simple RNN, the long short-term memory (LSTM) network, and the Gated Recurrent Unit (GRU) network. Figure 3 depicts the RNN.

In recurrent neural networks (RNN), the input consists of selected optimal features, while the output is the detected outcome, indicating whether the input data is classified as normal or malicious.

The traditional feed forward neural network is expanded upon by an RNN. The reason the RNN is referred to as "recurrent" is because it completes the same task for each element of a sequence, with the output depending on the results of the earlier calculations. The hidden states of the RNN are calculated mathematically using Eq. (9),

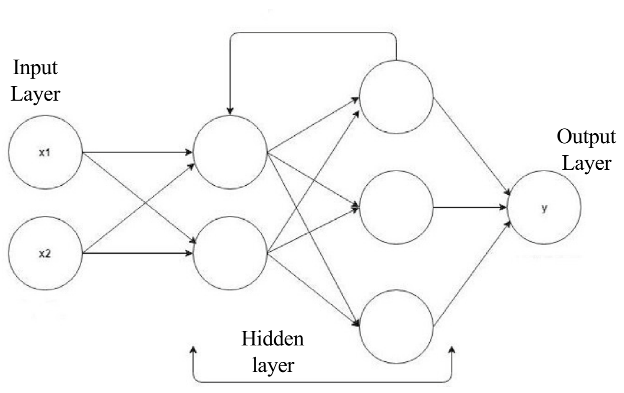


Fig. 3 RNN [36]

$$s_y = \sigma\left(K_{z_y} + Js_{y-1} + L_s\right), y = 1, 2, \ldots.., M \tag{9}$$

where $\sigma(\bullet)$ is an equation for nonlinearity, z_y is an input row vector at time y, s_y is a hidden state row vector at time y, K is an input to hidden weight matrix, J is a hidden-to-hidden weight matrix, and L_s is a row vector bias term. Figure 4 depicts the DRL based intrusion detection.

Adaptive Spiral Seagull Optimization (ASSO)

Seagulls are a type of marine bird that exhibit a wide range of physical characteristics, including variations in size and weight. What sets them apart from many other creatures is their ability to consume both freshwater and saltwater. The most notable behavioural traits of seagulls are their migration and hunting habits. Seagulls travel in flocks during migration, assuming various positions to avoid collisions. They follow the direction that offers the best chances of survival, or the one with the lowest level of risk. Some seagulls alter their positions based on the behaviour of the strongest bird in the group. During migration over water, seagulls often engage in spiral-shaped attacks on other birds. Researchers have studied the mathematical principles underlying seagull migration and their methods for hunting prey.

Initialization The process begins by initializing several variables. These include the population size, which determines the number of candidate solutions that will be evaluated during each iteration. The current iteration t and the maximal iteration t_{max} are also initialized to keep track of the progress of the algorithm. The dimension of the

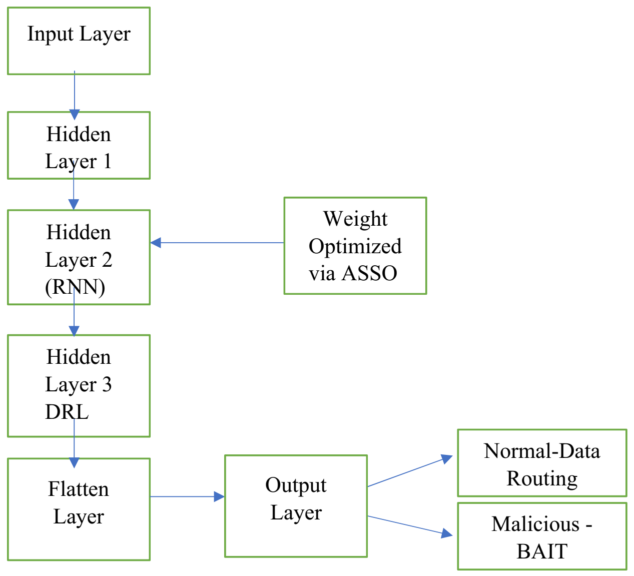


Fig. 4 Deep reinforcement learning-based intrusion detection

search space d is another important variable that needs to be set before running the algorithm. Finally, the position x_i of each candidate solution is also initialized, which represents a point in the search space that will be evaluated during the optimization process.

Fitness Computation During the optimization process, the fitness of each individual in the population is computed [35]. In this particular case, the fitness is calculated as per Eq. (10), the minimum attack detection error, denoted by d_{error}. The lower the value of d_{error}, the better the performance of the intrusion detection system. Therefore, the fitness value of an individual is equal to the minimum value of d_{error} achieved during the evaluation of the intrusion detection system using the corresponding feature subset. This fitness value is used to determine the selection probability of each individual for the next generation.

$$fit = \min(d_{error}) \tag{10}$$

Sorting The population is sorted based on the fitness, with the individuals having the lowest fitness values (i.e., the best fitness) at the beginning of the list. This sorting process allows for selecting the best individuals for further optimization and helps to increase the overall performance of the algorithm.

Proposed Migration (Exploration) The algorithm mimics the movement of a flock of seagulls as they fly from one spot to another. At this point, a seagull would be able to fulfil all three of their requirements.

(a) Avoiding the collisions

During the process of identifying the new search agent's position and used to stop collisions from happening with nearby seagull as per Eq. (11).

$$X_i^{t+1} = X_i^t + A * rand1() * \|X_j^t - X_i^t\| \tag{11}$$

where, X_i^t is the position of seagull i at time step t, A represents the variable that controls the step size, $rand1()$ is the random value between 0 to 1, and $\|X_j^t - X_i^t\|$ represents the distance between seagull i and its nearest neighbour j.

(b) Moving in the direction of the best neighbour

After avoiding the collisions, the seagull moves towards the direction of its best neighbour k as per Eq. (12).

$$X_i^{t+1} = X_i^t + A * rand2(0,1) * (X_k^t - X_i^t) \tag{12}$$

(c) Stay in close to the top search agent

After avoiding collisions and moving towards the direction of the best neighbor, the seagull can choose to stay close to the top search agent, G. This is done to explore the search space around the best solution found so far.

Proposed Attacking Phase (Exploitation)

(a) Memory Pool Update: In the attacking phase, the algorithm updates the memory pool with the best solution obtained so far. This is done to ensure that the algorithm retains the best solution found during the previous iterations. The updated memory pool is then used to guide the search process in the next iteration.

(b) Based on the memory of the previous attack, the seagull adjusts its attack strategy by changing its angle and speed while maintaining its altitude. The spiral movement behaviour occurs in the air as the seagull approaches its prey as per Eq. (13). This spiral attack pattern allows the seagull to confuse and disorient its prey, making it easier to catch. The algorithm simulates this behaviour by adjusting the position of the search agents in a spiral pattern around the current best position.

$$X_i^{t+1} = X_i^t + r * \sin(b) * m \tag{13}$$

where r is the distance from current position, a is the angle of attack, and m is the memory.

$$a_i^{t+1} = a_i^t + rand3(0,1) * \frac{X_{best}^t - X_i^t}{\|X_{best}^t - X_i^t\|} \tag{14}$$

The algorithm saves the solutions acquired so far and returns the best solution acquired. If the maximum number of iterations has been reached, the algorithm terminates. After detecting the intrusion, it moves to the mitigation phase.

The structure of ASSO pseudocode is shown in Algorithm 3.

Algorithm 3 ASSO

```
Input: seagull population
Output: optimal search agent
Procedure ASSO
Initialize parameters $t$, $t_{max}$, $x_i$, $d$
Calculate fitness as per Eq. (10)
Sort the seagull population based on fitness values
  i.  Proposed Migration – Exploration
        a. Avoid collisions using Eq. (11)and Eq.(12)
        b. Move towards the direction of best neighbours
        c. Close to top search agent
  ii. Proposed Attacking - Exploitation
        a. Enhancing iterative attacks with updated memory pool optimization
        b. Adapting angle and speed for confusing prey-spiral attack using Eq. (13) and
           Eq. (14)
Return value
End procedure
```

Intrusion Mitigation

If an intrusion is detected, the blockchain network takes necessary actions to mitigate the attack (BAIT).

BAIT

The proposed scheme consists of three steps, which are the Bait Step, Reverse Trace, and Reactive Defense. Each step is explained below:

Bait Step: In this step, the source node randomly selects an adjacent node within its one-hop neighbourhood and cooperates with this node to attract malicious nodes. The address of the adjacent node is taken as the destination address of the bait RREQ' packet. When a malicious node receives the RREQ', it replies with a false RREP without referring to its routing table. The next step is initiated as soon as the malicious node replies with a false RREP.

Reverse Trace: This step is used to detect the behaviour of malicious nodes through the route reply to the RREQ' message. The reverse tracing operation is conducted for nodes that receive the RREP to detect the suspicious path and the temporarily trusted path. The complete path list is stored in the RREP header, and the address of the malicious node is stored in the record address field of the RREP. To confirm that the malicious node is in the suspicious path set, the source node sends test packets to this route, and the result is fed back to the source node. The source node then stores the malicious node reported in a list of black holes and broadcasts the alarm packets through the network to inform all the other nodes to terminate their operation with the malicious node. Steps 1 and 2 are proactive defense steps. The third step is a reactive defense step.

Reactive Defense: In this step, the AODV route discovery process is initiated. Data transmission is started, and after some time, if the packet delivery ratio (PDR) significantly falls below a set threshold, the bait detection scheme is triggered again. The threshold is initially set to *Th* from 80 to 90% depending upon the network efficiency. As the network starts transmitting packets, the PDR is checked at regular intervals. If it falls below the threshold, step 1 of the Bait Step is activated again to detect any malicious nodes.

Shortest Path Calculation

The A* algorithm is a popular technique used for pathfinding and is commonly used to find the shortest path between two points in a graph. It uses a heuristic function to estimate the distance between a given node and the destination node, and then chooses the path with the lowest estimated cost.

A* Algorithm

A* algorithm is a searching algorithm used to find the shortest path between the initial and final state in various applications, such as maps. It is popularly used for pathfinding and graph traversals. The A* algorithm is commonly used for pathfinding and can be applied in various fields, including computer networking. In the context of data transmission, the gateway node can utilize the A* algorithm to determine the most efficient route between the source and destination nodes, optimizing factors such as distance, speed, and potential obstacles. This helps to minimize the time and resources required for the transmission process. The algorithm has three main parameters:

$\varepsilon g\varepsilon$ represents the cost of moving from the initial cell to the current cell. This value is the sum of all the cells visited since leaving the first cell.

$\varepsilon h\varepsilon$ is the heuristic value, which estimates the cost of moving from the current cell to the final cell. It is essential to ensure that there is never an overestimation of the cost.

$\varepsilon f\varepsilon$ represents the sum of $\varepsilon g\varepsilon$ and $\varepsilon h\varepsilon$. Therefore, $f = g + h$.

The A* algorithm makes decisions based on the $f - value$. It selects the cell with the smallest f-value and moves to that cell. The process continues until the algorithm reaches the goal cell.

A* algorithm is useful for graph traversals and finding the shortest path in maps. Suppose you have a graph and apply the A* algorithm to it, with the initial node $\varepsilon A\varepsilon$ and the goal node $\varepsilon E\varepsilon$. The algorithm uses the f-value to move towards the goal state.

Data Transmission

After the gateway node has determined the shortest path for data transmission using the A* algorithm, it will transmit the encrypted data to the destination node through the selected path. The transmission may be performed using various communication protocols, such as TCP/IP, UDP, or others depending on the requirements of the application. During transmission, the encrypted data is protected by cryptographic measures to ensure confidentiality, integrity, and authenticity. Once the data reaches the destination node, it will be decrypted using the same key that was used for encryption and processed as needed.

Decryption and Processing

Once the destination node receives the encrypted data, it first needs to decrypt it before it can be used for any further processing. The decryption process requires the use of the same key that was used for encryption at the source node. The decryption technique used depends on the encryption technique employed during data transmission. AES is a commonly used symmetric encryption algorithm that can be used to encrypt and decrypt data. The destination node can use the appropriate decryption technique, such as AES, to decrypt the data received from the gateway node.

Result and Discussion

The proposed model was implemented using PYTHON. The analysis and comparison of the suggested method's performance with the existing algorithms like Seagull Optimization Algorithm (SOA), Recurrent Neural Network (RNN), Multi-Layer Perceptron's (MLP).

Performance Metrics

The performance is compared using the confusion matrix like accuracy, precision, sensitivity, specificity, f-measure, NPV, FPR, FNR, AND MCC. The formula for calculating the metrics is discussed in this section.

(i) Accuracy

Accuracy is calculated as the fraction of correctly predicted cases to all examples.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

(ii) Precision

Precision is a valuable indication of how exactly the positive compounds are expected since it measures the percentage of properly anticipated positive instances to all test findings.

$$Precision = \frac{TP}{TP + FP}$$

(iii) Sensitivity

The sensitivity value may be obtained by disunion the total positives by the proportion of true positive forecasts.

$$Sensitivity = \frac{TP}{TP + FN}$$

(iv) Specificity

Specificity is defined as the proportion of accurately anticipated negative outcomes over all negative outcomes.

$$Specificity = \frac{TN}{TN + FP}$$

(v) F_Measure

The F-Measure number strikes a balance between ensuring that each class only includes a single type of data item and fully identifying all data bits.

$$F\_Score = \frac{Presision.Recall}{Presision + Recall}$$

(vi) Matthew's correlation coefficient (MCC)

MCC is a two-by-two binary variable association measure, which is represented below,

$$MCC = \frac{(TP \times TN - FP \times FN)}{\sqrt{(TP + FN)(TN + FP)(TN + FN)(TP + FP)}}$$

(vii) Negative Prediction Value (NPV)

A diagnostic test's or another quantitative metric's performance is described by NPV.

$$NPV = \frac{TN}{TN + FN}$$

(viii) False Positive Ratio (FPR)

The false positive rate is deliberate by segmentation the total number of negative events by the number of negative events that were wrongly labelled as positive (false positives).

$$FPR = \frac{FP}{FP + TN}$$

(ix) False Negative Ratio (FNR)

The false-negative rate, often known as the "miss rate," is the probability that the test may fail to detect a real positive".

$$FNR = \frac{FN}{FN + TP}$$

Table 2 Testing metrices-dataset 1: DRL based intrusion detection

Performance metrics	SOA	MLP	RNN	DRL – ASSO
Accuracy	0.933714	0.946824	0.903533	0.976242
Precision	0.979844	0.939041	0.865479	0.968206
Sensitivity	0.884490	0.954797	0.949703	0.984451
Specificity	0.982287	0.938957	0.857975	0.968141
F-Measure	0.902402	0.917333	0.876370	0.945824
MCC	0.880199	0.941019	0.865816	0.801440
NPV	0.853100	0.908822	0.903445	0.960463
FPR	0.021450	0.027524	0.025805	0.020882
FNR	0.004441	0.005894	0.006913	0.003660

Table 3 Testing metrices-dataset 2: DRL based intrusion detection

Performance metrics	SOA	MLP	RNN	DRL – ASSO
Accuracy	0.929720	0.930561	0.958824	0.988614
Precision	0.890563	0.976535	0.950942	0.980477
Sensitivity	0.977228	0.881503	0.966898	0.996928
Specificity	0.882842	0.978969	0.950857	0.980411
F-Measure	0.901770	0.899354	0.928959	0.957811
MCC	0.890910	0.877227	0.952945	0.814132
NPV	0.929630	0.850219	0.920340	0.972636
FPR	0.026553	0.021377	0.027873	0.021147
FNR	0.007113	0.004426	0.005969	0.003706

Overall Performance Analysis

Table 2 shows the testing metrics for four different DRL-based intrusion detection models, namely, SOA, MLP, RNN, and proposed DRL—ASSO, on Dataset 1. The proposed model achieved the highest accuracy of 0.976242, followed by the MLP model with an accuracy of 0.946824. The SOA and RNN models achieved accuracies of 0.933714 and 0.903533, respectively. In terms of precision, the SOA model achieved the highest precision of 0.979844, while the proposed model achieved a precision of 0.968206. The MLP and RNN models achieved precision values of 0.939041 and 0.865479, respectively. For sensitivity, the proposed model achieved the highest value of 0.984451, while the MLP and RNN models achieved sensitivity values of 0.954797 and 0.949703, respectively. The SOA model achieved a sensitivity value of 0.884490. The proposed model also achieved the highest specificity value of 0.968141, followed by the SOA model with a value of 0.982287. The MLP and RNN models achieved specificity values of 0.938957 and 0.857975, respectively. For the F-measure, the proposed model achieved the highest value of 0.945824, followed by the MLP model with a value of 0.917333. The SOA and RNN models achieved F-measure values of 0.902402 and 0.876370, respectively. The MCC values ranged from 0.801440 to 0.941019, with the proposed and MLP models achieving the lowest and highest values, respectively. The NPV values ranged from 0.853100 to 0.960463, with the proposed model achieving the highest value. Finally, the FPR and FNR values ranged from 0.020882 to 0.027524 and from 0.003660 to 0.006913, respectively. The proposed model achieved the lowest FNR value of 0.003660, indicating that it has the lowest false negative rate among all the models.

Table 3 compares the performance of four different models, SOA, MLP, RNN, and proposed DRL—ASSO, for DRL-based intrusion detection using dataset 2. The proposed model achieves the highest accuracy of 0.988614, which is significantly better than the other models. The precision of the proposed model is 0.980477, which is slightly lower than the MLP model but higher than the SOA and RNN models. The sensitivity of the proposed model is 0.996928, which is significantly higher than the SOA and MLP models and slightly higher than the RNN model. The specificity of the proposed model is 0.980411, which is higher than the SOA and RNN models and slightly lower than the MLP model. The F-Measure of the proposed model is 0.957811, which is higher than the SOA and MLP models but slightly lower than the RNN model. The MCC of the proposed model is 0.814132, which is higher than the SOA and MLP models but slightly lower than the RNN model. The NPV of the proposed model is 0.972636, which is higher than the SOA and RNN models but lower than the MLP model. The FPR of the proposed model is 0.021147, which is lower than all the other models, indicating that the proposed model has the lowest false positive rate. The FNR of the proposed model is 0.003706, which is the lowest among all models, indicating that the proposed model has the lowest false negative rate.

Table 4 compares the performance of the proposed method with two other baseline methods, SDWSN [20] and GWOSVM-IDS [18], on Dataset 1. The proposed system

Table 4 Testing metrices-dataset 1: base paper comparison

Performance metrics	SDWSN [20]	GWOSVM-IDS [18]	DRL—ASSO
Accuracy	0.918085	0.962073	0.976242
Precision	0.879417	0.954165	0.968206
Sensitivity	0.964998	0.970174	0.984451
Specificity	0.871793	0.954079	0.968141
F-Measure	0.890484	0.932107	0.945824
MCC	0.879760	0.956174	0.801440
NPV	0.917995	0.923459	0.960463
FPR	0.026220	0.027968	0.020882
FNR	0.007024	0.005989	0.003660

achieves the highest accuracy of 0.976242, followed by GWOSVM-IDS with 0.962073, and SDWSN with 0.918085. The precision of the proposed system is 0.968206, which is second only to GWOSVM-IDS with 0.954165, and much better than SDWSN with 0.879417. Similarly, the sensitivity and specificity of the proposed system are 0.984451 and 0.968141, respectively, which are higher than the other two systems. The F-measure of the proposed system is 0.945824, which is again higher than SDWSN but slightly lower than GWOSVM-IDS. The MCC of the proposed system is 0.801440, which is lower than the other two systems. The NPV of the proposed system is 0.960463, which is better than SDWSN but slightly worse than GWOSVM-IDS. The FPR and FNR of the proposed system are 0.020882 and 0.003660, respectively, which are better than the other two systems.

Table 5 compares the performance of the proposed intrusion detection system with two other baseline methods, SDWSN [20] and GWOSVM-IDS [18], using Dataset 2. The proposed system outperforms both baseline methods in terms of accuracy, precision, sensitivity, specificity, F-measure, and NPV. The accuracy achieved by the proposed method is 0.988614, which is higher than the accuracy of SDWSN [20] and GWOSVM-IDS [18] by 0.043066 and 0.014348, respectively. Similarly, the proposed method achieved higher precision, sensitivity, specificity, F-measure, and NPV compared to the baseline methods. In terms of false-positive rate (FPR) and false-negative rate (FNR), the proposed method performs better than SDWSN [20] and GWOSVM-IDS [18]. The FPR achieved by the proposed method is 0.021147, which is lower than the FPR of SDWSN [20] and GWOSVM-IDS [18] by 0.000575 and 0.007175, respectively. The FNR achieved by the proposed method is 0.003706, which is lower than the FNR of SDWSN [20] and GWOSVM-IDS [18] by 0.000792 and 0.002359, respectively.

Table 5 Testing metrices-dataset 2: base paper comparison

Performance metrics	SDWSN [20]	GWOSVM-IDS [18]	DRL—ASSO
Accuracy	0.945548	0.974266	0.988614
Precision	0.992262	0.966258	0.980477
Sensitivity	0.895700	0.982470	0.996928
Specificity	0.994736	0.966171	0.980411
F-Measure	0.913838	0.943921	0.957811
MCC	0.891355	0.968292	0.814132
NPV	0.863912	0.935163	0.972636
FPR	0.021722	0.028322	0.021147
FNR	0.004498	0.006065	0.003706

Overall Graphical Representation of Performance Analysis

Figure 5 is a graphical representation of the performance comparison between the existing models and the proposed model for dataset 1 and dataset 2. The x-axis represents the different models while the y-axis represents the values of the performance metrics. The graph has two subplots for each metric, one for dataset 1 and the other for dataset 2. From the graph, it can be observed that the proposed model outperforms the existing models for all the performance metrics for both datasets. The accuracy and F-Measure of the proposed model are higher than the existing models for both datasets. The FNR and FPR of the proposed model are lower than the existing models for both datasets. The MCC and NPV of the proposed model are higher than the existing models for dataset 1, while for dataset 2, the proposed model has lower MCC and higher NPV than the existing models. The Precision, Sensitivity, and Specificity of the proposed model are higher than the existing models for both datasets. Figure 6 shows the confusion matrix.

Conclusion

An intrusion detection and mitigation system based on sensor networks and deep reinforcement learning has been created in this work. The solution utilised the decentralised and transparent characteristics of blockchain technology to address the issues of security risks and data privacy concerns in sensor networks. Using data cleaning and transformation techniques, the strategy for intrusion detection in wireless sensor networks includes gathering and preprocessing datasets (NSL-KDD, CSE-CIC-IDS2018). The Pearson Correlation Based Principal Component Analysis (PC-PCA), statistical features, protocol-based features, and higher-order statistical features (HOS) were chosen as pertinent characteristics for the intrusion detection task. To prevent unauthorised access, the best features were encrypted with the latest AES. The integrity, immutability, and transparency of the data were subsequently secured by storing the encrypted data in the blockchain network. The chosen ideal features were fed into the multi-layer perceptron's (MLP) recurrent neural network (RNN) during the intrusion detection phase. To increase detection accuracy, ASSO was used to fine-tune the weight function of the RNN. The blockchain network performed the required steps to mitigate the attack (BAIT) if an incursion was discovered. The gateway node transferred the encrypted data to the destination node through the determined path after computing the shortest path for data transmission using the A* algorithm. The destination node handled the encrypted data for various

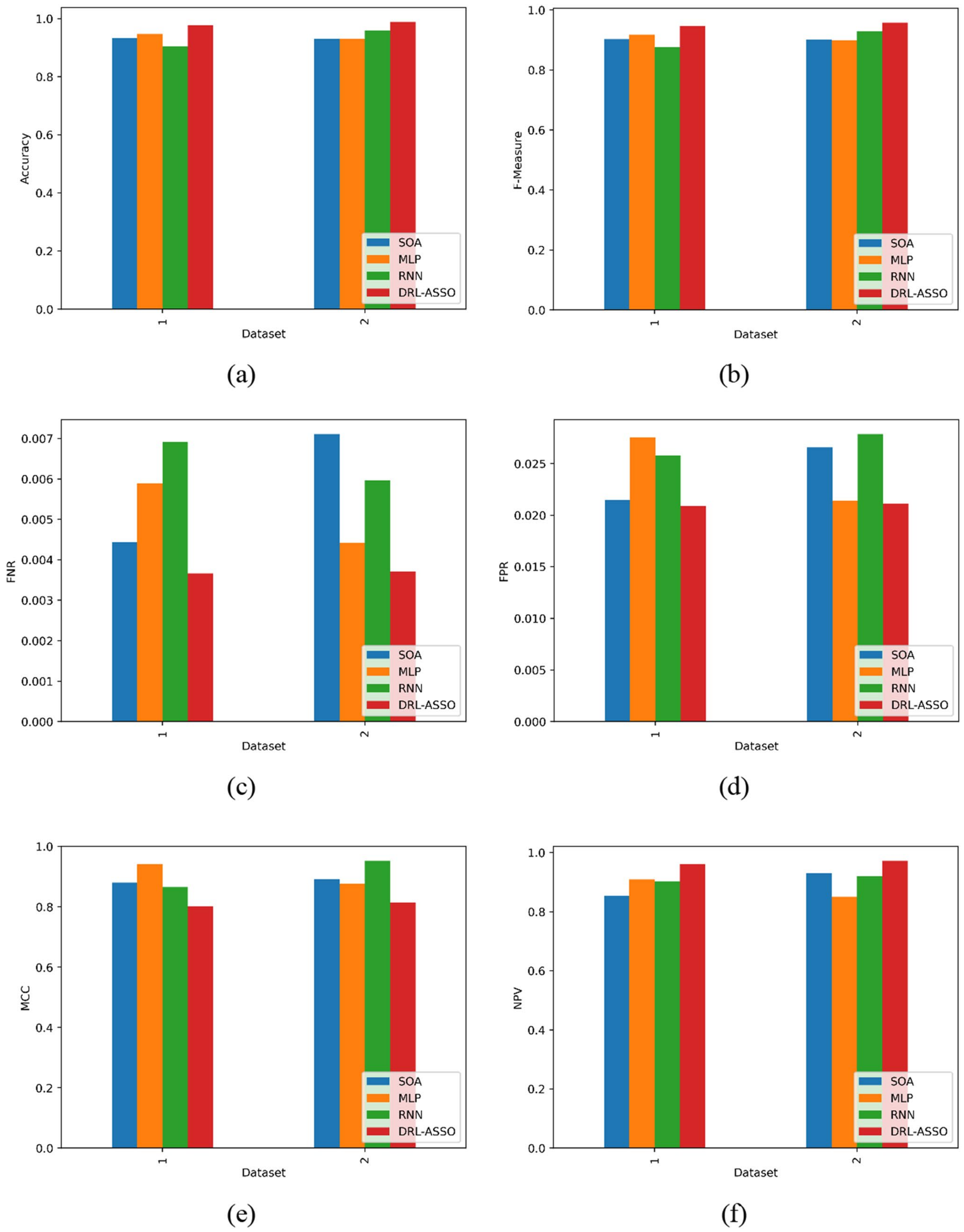
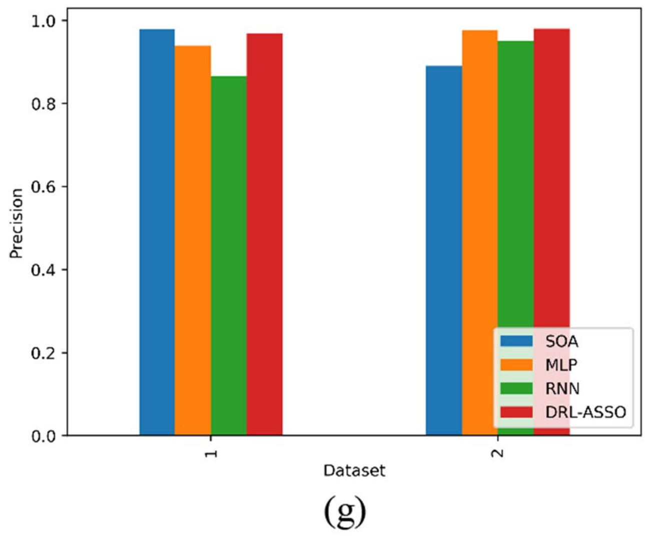
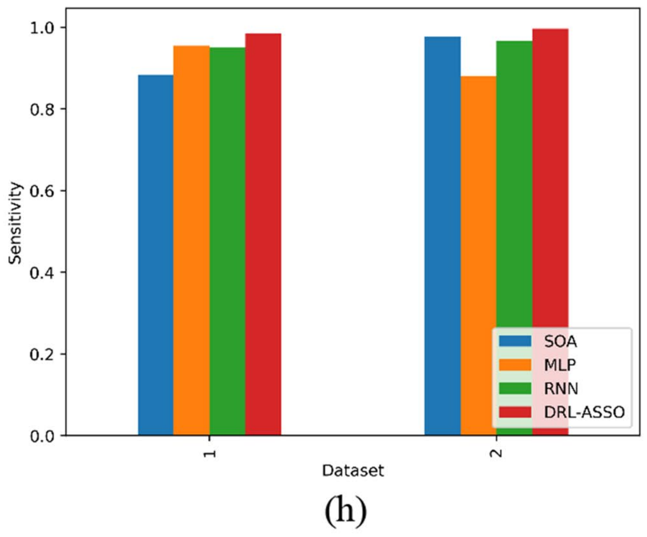


Fig. 5 Overall Graphical Representation of existing and proposed model for dataset 1 and dataset 2 **a** Accuracy, **b** F-Measure, **c** FNR, **d** FPR, **e** MCC, **f** NPV, **g** Precision, **h** Sensitivity, **i** Specificity





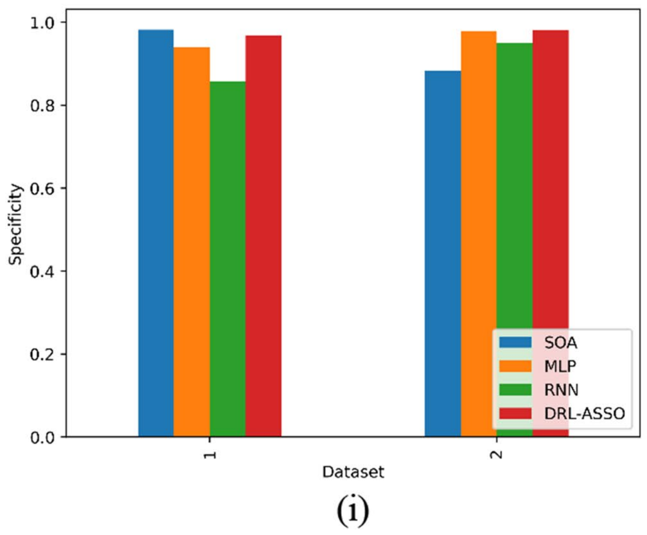


Fig. 5 (continued)

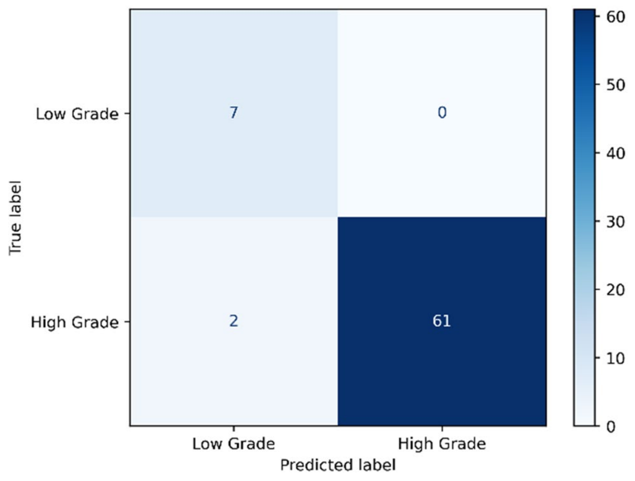


Fig. 6 Confusion matrix

applications after decrypting it using the proper decryption method. Python was used to implement the suggested model.

In summary, there are a number of advantages to the suggested blockchain-based solution for intrusion detection and sensor network security. The accuracy of intrusion detection is improved by combining deep reinforcement learning with blockchain. Preprocessing the data guarantees consistency, while feature extraction and encryption help create strong security. But there are few significant drawbacks and restrictions. The computational intricacy of encryption with deep reinforcement learning could provide difficulties. The effectiveness of the system may also be influenced by the network configuration and datasets chosen. Taking these aspects into account carefully, it is necessary to ensure optimal performance. Subsequent investigations can be developed for optimisation strategies to tackle computational complexities and assess the system in various network scenarios. Furthermore, there is still room to explore how to make the suggested model more flexible and scalable to various intrusion circumstances in cybersecurity domain.

Data availability Data links are available in References [1] and [2].

Declarations

Conflict of Interest The author declare that they have no conflict of interest.

References

1. Canadian Institute of cybersecurity, university of new brunswick, ISCX dataset https://www.unb.ca/cic/datasets/nsl.html (Accessed on 09 May 2023)
2. Canadian Institute of cybersecurity, university of new brunswick, IDS dataset https://www.unb.ca/cic/datasets/ids-2018.html (Accessed on 09 May 2023)
3. Khan MA, Kim J. Toward developing efficient conv-AE-based intrusion detection system using heterogeneous dataset. Electronics. 2020;9:1771. https://doi.org/10.3390/electronics9111771.
4. Al-Daweri MS, Zainol Ariffin KA, Abdullah S, Md. Senan MFE. An analysis of the KDD99 and UNSW-NB15 datasets for the intrusion detection system. Symmetry. 2020;12:1666. https://doi.org/10.3390/sym12101666.
5. Kumar R, Kumar P, Tripathi R, Gupta GP, Garg S, Hassan MM. A distributed intrusion detection system to detect DDoS attacks in blockchain-enabled IoT network. J Parallel Distrib Comput. 2022;164:55–68.
6. Banerjee M, Lee J, Choo KKR. A blockchain future for internet of things security: a position paper. Digital Commun Netw. 2018;4(3):149–60.
7. Liang C, Shanmugam B, Azam S, Karim A, Islam A, Zamani M, Kavianpour S, Idris NB. Intrusion detection system for the internet of things based on blockchain and multi-agent systems. Electronics. 2020;9(7):1120.
8. Mubarakali A. An efficient authentication scheme using blockchain technology for wireless sensor networks. Wirel Person Commun. 2021. https://doi.org/10.1007/s11277-021-08212-w.
9. Khalaf OI, Abdulsahib GM. Optimized dynamic storage of data (ODSD) in IoT based on blockchain for wireless sensor networks. Peer-to-Peer Netw Appl. 2021;14:2858–73.
10. Li W, Wang Y, Li J, Au MH. Toward a blockchain-based framework for challenge-based collaborative intrusion detection. Int J Inf Secur. 2021;20:127–39.
11. Yang J, He S, Xu Y, Chen L, Ren J. A trusted routing scheme using blockchain and reinforcement learning for wireless sensor networks. Sensors. 2019;19(4):970.
12. Khan AA, Khan MM, Khan KM, Arshad J, Ahmad F. A blockchain-based decentralized machine learning framework for collaborative intrusion detection within UAVs. Comput Netw. 2021;196: 108217.
13. Rathore S, Kwon BW, Park JH. BlockSecIoTNet: blockchain-based decentralized security architecture for IoT network. J Netw Comput Appl. 2019;143:167–77.
14. He D, Chan S, Ni X, Guizani M. Software-defined-networking-enabled traffic anomaly detection and mitigation. IEEE Internet Things J. 2017;4(6):1890–8. https://doi.org/10.1109/JIOT.2017.2694702.
15. Rahman A, Islam MJ, Montieri A, Nasir MK, Reza MM, Band SS, Pescape A, Hasan M, Sookhak M, Mosavi A. Smartblock-sdn: an optimized blockchain-sdn framework for resource management in IoT. IEEE Access. 2021;9:28361–76.
16. Chaabouni N, Mosbah M, Zemmari A, Sauvignac C, Faruki P. Network intrusion detection for IoT security based on learning techniques. IEEE Commun Surv Tutor. 2019;21(3):2671–701. https://doi.org/10.1109/COMST.2019.2896380.
17. Cao S, Dang S, Zhang Y, Wang W, Cheng N. A blockchain-based access control and intrusion detection framework for satellite communication systems. Comput Commun. 2021;172:216–25.
18. Ferrag MA, Maglaras L. DeepCoin: a novel deep learning and blockchain-based energy exchange framework for smart grids. IEEE Trans Eng Manage. 2019;67(4):1285–97.
19. Goyat R, Kumar G, Rai MK, Saha R, Thomas R, Kim TH. Blockchain powered secure range-free localization in wireless sensor networks. Arab J Sci Eng. 2020;45:6139–55.
20. Sun Z, Xu Y, Liang G, Zhou Z. An intrusion detection model for wireless sensor networks with an improved V-detector algorithm. IEEE Sens J. 2018;18(5):1971–84. https://doi.org/10.1109/JSEN.2017.2787997.
21. Qu H, Qiu Z, Tang X, Xiang M, Wang P. Incorporating unsupervised learning into intrusion detection for wireless sensor networks with structural co-evolvability. Appl Soft Comput. 2018;71:939–51.
22. Safaldin M, Otair M, Abualigah L. Improved binary gray wolf optimizer and SVM for intrusion detection system in wireless sensor networks. J Ambient Intell Humaniz Comput. 2021;12:1559–76.
23. Jin X, Liang J, Tong W, Lu L, Li Z. Multi-agent trust-based intrusion detection scheme for wireless sensor networks. Comput Electr Eng. 2017;59:262–73.
24. Miranda C, Kaddoum G, Bou-Harb E, Garg S, Kaur K. A collaborative security framework for software-defined wireless sensor networks. IEEE Trans Inf Forensics Secur. 2020;15:2602–15.
25. Wang J, Jiang S, Fapojuwo AO. A protocol layer trust-based intrusion detection scheme for wireless sensor networks. Sensors. 2017;17(6):1227.
26. Han L, Zhou M, Jia W, Dalil Z, Xu X. Intrusion detection model of wireless sensor networks based on game theory and an autoregressive model. Inf Sci. 2019;476:491–504.
27. Butun I, Ra IH, Sankar R. An intrusion detection system based on multi-level clustering for hierarchical wireless sensor networks. Sensors. 2015;15(11):28960–78.
28. Alqahtani M, Gumaei A, Mathkour H, Ismail MMB. A genetic-based extreme gradient boosting model for detecting intrusions in wireless sensor networks. Sensors. 2019;19(20):4383.
29. Almomani I, Alromi A. Integrating software engineering processes in the development of efficient intrusion detection systems in wireless sensor networks. Sensors. 2020;20(5):1375.
30. Rajasoundaran S, Kumar SS, Selvi M, Ganapathy S, Rakesh R, Kannan A. Machine learning based volatile block chain construction for secure routing in decentralized military sensor networks. Wireless Netw. 2021;27(7):4513–34.
31. Li W, Wang Y, Li J. Enhancing blockchain-based filtration mechanism via IPFS for collaborative intrusion detection in IoT networks. J Syst Architect. 2022;127: 102510.
32. Sundararajan RK, Arumugam U. Intrusion detection algorithm for mitigating sinkhole attack on LEACH protocol in wireless sensor networks. J Sens. 2015. https://doi.org/10.1155/2015/203814.
33. Mbarek B, Ge M, Pitner T. An adaptive anti-jamming system in HyperLedger-based wireless sensor networks. Wireless Netw. 2022;28(2):691–703.

34. Babu ES, SrinivasaRao BKN, Nayak SR, Verma A, Alqahtani F, Tolba A, Mukherjee A. Blockchain-based Intrusion Detection System of IoT urban data with device authentication against DDoS attacks. Comput Electr Eng. 2022;103: 108287.
35. Mansour RF. Artificial intelligence based optimization with deep learning model for blockchain enabled intrusion detection in CPS environment. Sci Rep. 2022;12:12937. https://doi.org/10.1038/s41598-022-17043-z.
36. Yin C, Zhu Y, Fei J, He X. A deep learning approach for intrusion detection using recurrent neural networks. IEEE Access. 2017;5:21954–61. https://doi.org/10.1109/ACCESS.2017.2762418.