scientific reports

Check for updates

OPEN

Secure cloud computing: leveraging GNN and leader K-means for intrusion detection optimization

Raman Dugyala[1], Premkumar Chithaluru[2], M. Ramchander[3], Sunil Kumar[4,5], Arvind Yadav[6], N. Sudhakar Yadav[7], Diaa Salama Abd Elminaam[8,9] & Deema Mohammed Alsekait[10]✉

Over the past two decades, cloud computing has experienced exponential growth, becoming a critical resource for organizations and individuals alike. However, this rapid adoption has introduced significant security challenges, particularly in intrusion detection, where traditional systems often struggle with low detection accuracy and high processing times. To address these limitations, this research proposes an optimized Intrusion Detection System (IDS) that leverages Graph Neural Networks and the Leader K-means clustering algorithm. The primary aim of the study is to enhance both the accuracy and efficiency of intrusion detection within cloud environments. Key contributions of this work include the integration of the Leader K-means algorithm for effective data clustering, improving the IDS's ability to differentiate between normal and malicious activities. Additionally, the study introduces an optimized Grasshopper Optimization algorithm, which enhances the performance of the Optimal Neural Network, further refining detection accuracy. For added data security, the system incorporates Advanced Encryption Standard encryption and steganography, ensuring robust protection of sensitive information. The proposed solution has been implemented on the Java platform with CloudSim support, and the findings demonstrate a significant improvement in both detection accuracy and processing efficiency compared to existing methods. This research presents a comprehensive solution to the ongoing security challenges in cloud computing, offering a valuable contribution to the field.

The rapid growth in cloud computing adoption is driven by its flexibility and cost-effectiveness in service delivery[1]. As a model for on-demand IT services, cloud computing depends on distributed computing technologies[2], evolving into a new computational paradigm aimed at providing reliable, scalable, and high-quality services and infrastructure at reasonable costs[3]. Cloud computing services are categorized into three main models: Infrastructure as a Service (IaaS), Software as a Service (SaaS), and Platform as a Service (PaaS)[4]. These technologies have become central to modern computing, particularly with the rise of social networking and mobile cloud computing, underscoring the importance of IDS in maintaining security.

Cloud computing not only optimizes database performance and speed but also makes IDS and Intrusion Prevention Systems (IPS) crucial for security management. IDS has been highlighted as an effective tool for network protection, prompting researchers to explore new methods for identifying and preventing attacks[5]. Mechanisms like service retention and security management via infrastructure and software emphasize IDS's role in preventing attacks, healing systems post-attack, and diagnosing safety issues to prevent future threats[6]. The ethical collection and analysis of data are essential for maintaining control over networks, data centers,

[1]Department of Computer Science and Engineering, Chaitanya Bharathi Institute of Technology, Hyderabad 500075, India. [2]Department of Information Technology, Mahatma Gandhi Institute of Technology, Telangana 500075, India. [3]Department of MCA, Chaitanya Bharathi Institute of Technology, Hyderabad 500075, India. [4]Department of Computer Engineering and Applications, GLA University, Noida, India. [5]Department of Computer Science, Graphic Era Hill University, Dehradun 248001, India. [6]School of Computer Science and Engineering, Galgotias University, Mathura, Uttar Pradesh 203201, India. [7]Department of Information Technology, Chaitanya Bharathi Institute of Technology, Hyderabad 500075, India. [8]Faculty of Computers and Artificial Intelligence, Benha University, Banha, Egypt. [9]Jadara Research Center, Jadara University, Irbid 21110, Jordan. [10]Department of Computer Science , Applied College, Princess Nourah Bint Abdulrahman University, P.O. Box 84428, 11671 Riyadh, Saudi Arabia. ✉email: Dmalsekait@pnu.edu.sa

or clouds, with firewalls, IDS, and IPS being critical security tools[7]. Historically, IDS has been a key player in monitoring and controlling network traffic, adding an extra layer of protection against malicious or unauthorized use of computer systems[8,11,13].

IDSs use versatile search techniques, including monitoring, dissemination, and collaboration, with evolutionary algorithms and Machine Learning (ML) methods augmenting their control[14,16,17]. However, Network-based IDSs (NIDS) and Host-based IDSs (HIDS) can impact cloud performance by consuming CPU and memory resources, which makes efficient Cloud Intrusion Detection and Prevention Systems (CIDPS) necessary to maintain service integrity[18–20].

In recent years, cloud computing has become the backbone of modern IT infrastructure, offering scalable and flexible resources. However, with this rise has come an increase in cyber threats, particularly Denial-of-Service (DoS) and Distributed DoS (DDoS) attacks, posing significant security challenges to cloud environments. Traditional security measures often prove inadequate in such dynamic and distributed settings. To address these challenges, this paper focuses on developing and evaluating an IDS tailored specifically for cloud environments, with a primary focus on detecting and mitigating DoS and DDoS attacks. Additionally, while the primary focus is on these attacks, the system's capacity to detect and respond to Man-in-the-Middle (MIM) attacks is also briefly explored. Given the rise in cyberattacks across computer networks, IDSs are critical for monitoring suspicious activities. In cloud environments, IDSs are essential for systematically monitoring registers, configurations, and network traffic to enhance security. However, traditional IDSs face challenges in detecting anomalies within cloud environments, necessitating novel solutions that are tailored to cloud-specific threats such as DDoS attacks and trust-related threats. As a result, this work aims to develop IDS solutions designed specifically for cloud environments.

To overcome these challenges, we propose an optimized IDS that leverages advanced algorithms to improve both the accuracy and efficiency of attack detection. Central to this system are the Leader K-means algorithm and GNN, both of which play a critical role in enhancing the system's detection capabilities.

- *Leader K-means algorithm* This algorithm is used for efficient data clustering, which significantly reduces the computational complexity of the IDS. By clustering data effectively, the system focus on smaller, more manageable groups of data, facilitating faster and more accurate intrusion detection.
- *GNN* Once the data is clustered, the GNN is employed to meticulously analyze these clusters. The GNN excels at recognizing patterns and anomalies within the data, enabling it to accurately distinguish between normal and intrusive activities. This NN is further optimized using the ONN framework, which enhances its detection precision.
- *GHO method* We integrate tree pod enhancement to optimize the GHO method, which fine-tunes the detection process by dynamically adjusting parameters to improve the overall performance of the IDS.
- *AES and steganography* To bolster security further, AES algorithm is utilized for encryption, ensuring data integrity and confidentiality. Steganography is also applied to enhance memory security, protecting sensitive information from unauthorized access.By incorporating these advanced techniques, the proposed IDS presents a robust solution that substantially enhances attack detection accuracy and processing efficiency within cloud environments. This comprehensive approach effectively addresses the limitations of traditional IDS, offering a more reliable and secure experience for cloud computing users, and contributing to the ongoing development of more resilient cloud infrastructure.

Motivation

The motivation for this work arises from the escalating cyber threat landscape, where attacks have become increasingly frequent and sophisticated, particularly in cloud computing environments. As organizations and individuals increasingly depend on cloud services for data storage and processing, securing these environments has become crucial. Traditional IDS often fall short in effectively safeguarding cloud environments due to limitations in detecting network-based anomalies and hidden attack paths. This highlights the urgent need for innovative security solutions specifically designed to address the unique challenges posed by cloud computing.

Cloud environments, with their distributed nature and accessibility over the internet, are particularly susceptible to a wide range of attacks from both internal and external sources. The growing reliance on cloud services to store sensitive data heightens the risk of data breaches, emphasizing the need for advanced security mechanisms to protect valuable information. While traditional security methods-such as proximity search, automation, and access control-exist, they often prove inadequate when applied to the dynamic and complex nature of cloud computing.

One of the major threats to cloud environments is DDoS attacks, which disrupt legitimate access to servers, compromise network bandwidth, and lead to significant financial losses. Effectively addressing these challenges requires the development of more advanced IDS that can accurately detect and mitigate both known and emerging cyber threats within cloud environments.

In response to these critical security concerns, this study proposes an optimized IDS specifically tailored for cloud environments. The system leverages advanced algorithms such as Leader K-means for efficient data clustering and GNN for in-depth analysis, enhancing both detection accuracy and processing efficiency. Additionally, the integration of encryption techniques like AES and steganography strengthens memory security, ensuring that sensitive data remains protected from unauthorized access and breaches. By addressing the specific security needs of cloud computing, this research contributes significantly to the field of cloud security, offering a comprehensive solution to mitigate cyber threats and safeguard critical data and infrastructure.

The proposed study's contributions are outlined as follows:

- We aim to improve the accuracy and efficiency of IDS in identifying and mitigating security threats, particularly under the conditions of DoS and DDoS attacks.
- Proposed work focuses on ensuring the integrity and confidentiality of stored data in environments susceptible to cyber-attacks, including DDoS, DoS, and MIM attacks.
- By leveraging a leader-based k-means clustering algorithm in conjunction with a GNN, we aim to enhance the precision and speed of clustering processes, which are crucial for the accurate classification and detection of intrusions.
- We assess our proposed solution across various parameters such as clustering accuracy, intrusion detection frequency, encryption and decryption times, and overall system robustness against cyber threats.

Literature review

The increasing reliance on cloud computing has underscored the need for robust security mechanisms, particularly in the domain of IDS. Numerous studies have investigated different approaches to bolster cloud security, focusing on IDS and complementary protective measures.

Gill et al. have made significant strides in Cloud Resource Management (CRM), emphasizing self-protection mechanisms in cloud environments. Their research highlights the necessity of adaptive security measures for managing resources efficiently while safeguarding data and services. By employing SNORT, an open-source network IDS, their study evaluates the effectiveness of real-time intrusion detection and prevention, contributing to enhanced cloud resource security[21,22].

Vieira et al. offer an innovative approach to intrusion detection in grid and cloud computing environments by merging knowledge-based and behavior-based techniques. Their IDS combines signature-based detection, which identifies known attack patterns, with anomaly detection, which flags deviations from normal behavior. This dual method improves detection rates while reducing false positives, making their IDS more reliable for cloud infrastructure protection[23,24].

Lorenz et al. introduce a new approach to enforcing fine-grained security policies in cloud network architectures through Software Defined Networks (SDN) and Network Function Virtualization (NFV). Their framework increases the flexibility of cloud networks while reducing operational costs. Virtualized components like load balancers and firewalls enable dynamic, scalable security measures that adapt to evolving threats, improving both security and resource utilization in the cloud[25,26].

Varadharajan et al. emphasize the role of integrated architectures in ensuring cloud security. Their research integrates resilient intrusion detection techniques with trusted computing technologies, forming a secure, resilient system capable of detecting diverse and dynamic threats. This system continuously monitors cloud environments, allowing timely detection of breaches while leveraging trusted computing to ensure only authorized entities access critical resources[27,28].

Colom et al. investigate distributed IDS in heterogeneous network architectures, introducing advanced scheduling algorithms that optimize intrusion detection across varied computing environments. By accounting for fluctuations in resource availability, their distributed IDS approach enhances security, especially in large-scale cloud environments where demand can vary widely[29].

Sultana et al. propose an IDS tailored for SDN, integrating ML to monitor and detect network security threats. Their system utilizes ML algorithms to analyze traffic patterns, identifying potential risks in real-time. By leveraging SDN's adaptability, this IDS offers real-time protection and improves detection of previously unknown threats, contributing significantly to cloud security[30].

Jin et al. focus on intrusion prevention in virtualized cloud environments, introducing VMFence, a security framework based on the Virtual Machine Manager (VMM). VMFence, implemented on an open-source virtual machine monitoring platform, optimizes hardware resources while ensuring robust security against intrusions. By monitoring virtual machine activities and detecting anomalies, VMFence helps prevent unauthorized access, reinforcing the integrity of cloud environments[31–33].

These studies collectively address diverse aspects of cloud security, ranging from resource management to intrusion detection and prevention, illustrating the critical importance of developing adaptive, scalable, and efficient security mechanisms for the cloud.

Research gaps identified in the literature

Based on the extensive literature review provided, several research gaps can be identified that offer opportunities for further exploration and development in the field of cloud security and intrusion detection:

- *Real-time adaptation and scalability*
  - *Gap* While many studies focus on enhancing security through IDS and resource management, there is a noticeable gap in the real-time adaptation and scalability of these systems to dynamically changing cloud environments. For instance, the integration of SNORT in cloud environments by Sukhpal Singh Gill et al. highlights the potential for self-protection, but the study does not fully address how these systems can scale and adapt in real-time to handle growing and fluctuating cloud workloads[21,22].
  - *Opportunity* Future research could explore more sophisticated real-time adaptive mechanisms that allow IDS and security frameworks to automatically adjust their operations based on real-time analysis of cloud resource usage and threat levels, thereby improving both scalability and responsiveness.
- *Comprehensive integration of ML with IDS*

  - *Gap* Sultana et al. propose using ML in SDN-based IDS, which shows promise in enhancing detection accuracy. However, the integration of ML with IDS systems in cloud environments is still in its infancy, particularly in terms of developing models that can adapt to evolving threats and learn from new types of attacks over time[30].
  - *Opportunity* There is a significant gap in the comprehensive application of adaptive ML algorithms within IDS that not only detect anomalies but also continually evolve and improve their accuracy by learning from new data, thus minimizing false positives and negatives.

- *Addressing modern and emerging threats*

  - *Gap* Several studies, such as those by Vieira et al. and Jin et al. focus on traditional threat models and older datasets like KDD CUP 99 for benchmarking IDS[34]. While these provide a solid foundation, they do not adequately address the modern and emerging threats that cloud environments face today[23,31]. The use of outdated datasets limits the applicability of these studies to current cloud security challenges.
  - *Opportunity* Future research should focus on using up-to-date datasets and exploring new threat vectors, such as those related to IoT, AI-driven attacks, and advanced persistent threats (APTs), to develop more robust and future-proof IDS solutions.

- *Optimization and resource efficiency*

  - *Gap* The optimization of security frameworks, as discussed by Lorenz et al. often centers around the efficiency of network resources and cost reduction using technologies like SDN and NFV. However, there is a gap in ensuring that these optimizations do not compromise the overall security and that they can handle the resource-intensive nature of high-security environments without degradation in performance[25,26].
  - *Opportunity* There is room for research into hybrid optimization strategies that balance resource efficiency with robust security, ensuring that cost savings and performance enhancements do not come at the expense of security efficacy.

- *Lack of comprehensive security frameworks*

  - *Gap* While Varadharajan et al. discuss an integrated architecture combining IDS with trusted computing technologies, the implementation of truly comprehensive security frameworks that cover the full spectrum of cloud vulnerabilities, from data encryption to intrusion prevention and beyond, is still lacking[27]. Existing solutions often focus on specific aspects of security rather than providing a holistic approach.
  - *Opportunity* Future studies could aim to develop and validate comprehensive security frameworks that integrate multiple layers of protection, including IDS, encryption, data integrity verification, and user authentication, into a single cohesive system.

- *Analysis of computational complexity and resource utilization*

  - *Gap* Despite the detailed exploration of various IDS approaches, studies such as those by Colom et al. often overlook the analysis of computational complexity and resource utilization, which are critical for practical deployment in cloud environments[29]. Understanding the resource demands and processing overhead of these systems is essential for ensuring their viability in real-world applications.
  - *Opportunity* Further research is needed to quantify the computational costs and resource requirements of proposed IDS solutions, enabling better optimization and ensuring that these systems can be effectively deployed in resource-constrained cloud environments.

- *Comparative analysis of encryption techniques*

  - *Gap* The work by Jin et al. discusses the integration of AES and steganography for secure storage but does not provide a comparative analysis against other modern encryption methods. This limits the understanding of how effective these techniques are in various cloud security scenarios[31–33].
  - *Opportunity* Future research should conduct a comparative analysis of different encryption techniques, including advanced methods like homomorphic encryption and quantum-resistant algorithms, to determine the most effective strategies for securing cloud data.

Proposed methodology

Our proposed IDS leverages a combination of GNN and the Leader K-means algorithm to enhance detection accuracy and efficiency. The system is specifically designed to address the unique characteristics of cloud environments, ensuring robust performance against various attack vectors. The leader-based K-means algorithm facilitates efficient data clustering, allowing the IDS to effectively distinguish between normal and intrusive activities. Additionally, we have integrated tree pod enhancement techniques alongside the GHO algorithm to further optimize the system's performance. To bolster data security within the cloud, the AES algorithm and steganography are employed.

To evaluate secure storage solutions against DDoS and DoS attacks, we outline a detailed encryption and decryption workflow using AES. The process begins with key generation, creating a 256-bit key to ensure robust encryption. Next, plaintext data is encrypted using AES in Cipher Block Chaining (CBC) mode, which incorporates an Initialization Vector (IV) for added security. The encrypted data (ciphertext) and the IV are stored securely, with the key kept separate to prevent unauthorized access. For decryption, the ciphertext and IV

are retrieved, allowing the original plaintext to be recovered using the same AES key. This methodology ensures that data remains confidential and is accessible only to authorized parties, even if the storage system is targeted by attacks.

The overarching objective of this research is to develop a robust IDS by harnessing leader-based K-means clustering and a GHO-based neural network. The proposed system is structured around three key modules: (i) Clustering, (ii) Intrusion Detection, and (iii) Secure Storage. Within the clustering module, input data undergoes grouping facilitated by an intermediate K-means clustering method with leader nodes. Subsequently, the Intrusion Detection module processes the clustered data utilizing the ONN, wherein neuron weights are meticulously selected using the GHO technique. The neural network structure is trained based on these optimized weights, after which test data undergoes thorough analysis for intrusion detection. Lastly, the Secure Storage module enhances security measures by employing a hybridized approach that combines AES and steganography methods, with the GHO algorithm optimizing key attributes within AES encryption. The flow of the proposed methodology is depicted in Fig. 1.

Figure 2 illustrates the detailed architecture of the proposed system in a flow diagram, showcasing the interconnected processes and algorithms designed to enhance intrusion detection in cloud environments. The process begins with cloud resources, which represent the cloud infrastructure from which data is collected. This raw data undergoes a cleaning phase to eliminate any noise or inconsistencies, followed by normalization to ensure uniformity across the dataset. The normalized data is then clustered using the Leader K-means clustering algorithm, which facilitates the efficient grouping of similar data points. Subsequently, the clustered data is processed by a GNN to differentiate between normal and anomalous traffic. The outputs from the GNN are further refined using the GHO algorithm, enhancing detection accuracy. To secure the data, AES encryption is applied, complemented by steganography techniques to bolster memory security. This integrated approach ensures a robust and efficient intrusion detection system, effectively addressing challenges related to accuracy and processing time in cloud computing environments.

Clustering using leader K-Means algorithm

The partitioning dialogue is essential for segmenting the provided data into clusters. However, when faced with a large volume of data-specifically, a D-shaped integer containing n numbers and B signs-effective management can become challenging. To address this issue, the proposed method employs a leader-based k-means clustering

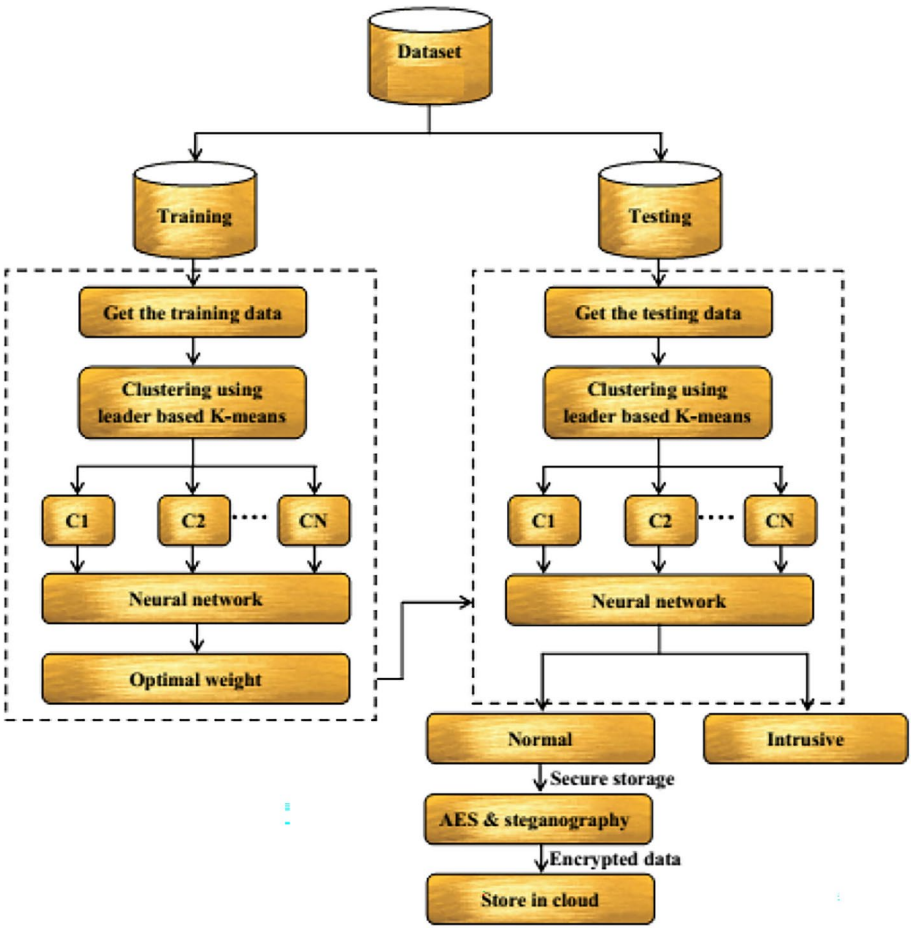

Fig. 1. Flow diagram of the proposed method.

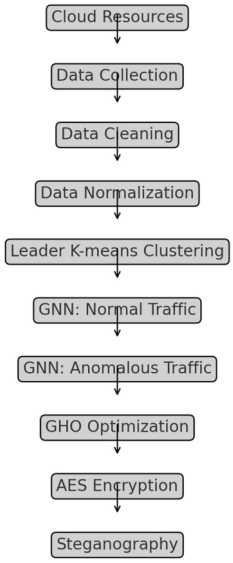

Fig. 2. Procedural flow diagram.

algorithm, which first integrates the data into N clusters, thereby alleviating the complexity associated with the clustering process. The clustering process unfolds through the following steps:

- Implement the K-means clustering algorithm for leader selection and
- Utilize the leader-based clustering process to group the input data.

Optimized K-means clustering

Clustering is a technique employed to partition a dataset into distinct groups, with the K-means clustering approach being one of the most widely used methods. This approach involves segmenting the data into a predefined number of clusters. The steps for executing optimized K-means clustering are outlined below:

procedure OPTIMIZEDKMEANS
Step 1: Initialize a number of clusters k and cluster centers using advanced initialization techniques like K-means++.
Step 2: Compute the Euclidean distance between the cluster centers (c) and input data (X) using optimized distance calculation methods, such as vectorization or parallelization.

$$d = \sum_{i=1}^{n} \sum_{j=1}^{C} (||X_i - c_j||)^2 \tag{1}$$

▷ Where n is the number of data points in the i^{th} cluster, and C is the number of cluster centers.
Step 3: Assign each data point to the nearest cluster center using efficient data assignment techniques, possibly leveraging data structures like kd-trees.
Step 4: Update the cluster centers based on the assigned data points using optimized center update methods, potentially incorporating parallel processing frameworks.

$$c_j = \left(\frac{1}{n}\right) \sum_{i=1}^{n} X_i \tag{2}$$

Step 5:
if there are empty clusters **then**
Handle empty clusters to prevent algorithm stagnation.
end if
end procedure

Algorithm 1. Pseudo code of Optimized K-means clustering

The Algorithm 1 outlines an optimized version of the classic K-means clustering algorithm. It begins by initializing a specified number of clusters and their corresponding centers, potentially utilizing advanced techniques like K-means++ for more strategic initialization. Next, the algorithm computes the Euclidean distance between each data point and the cluster centers, employing optimized distance calculation methods such as vectorization or parallelization to expedite computations, particularly for large datasets or high-dimensional data. Following this, each data point is assigned to the nearest cluster center using efficient data assignment techniques, possibly leveraging data structures like kd-trees to enhance performance. The algorithm then updates the cluster centers based on the assigned data points, utilizing optimized center update methods that may incorporate parallel processing frameworks to further improve computational efficiency. Provisions are also included to handle empty clusters, preventing algorithm stagnation and ensuring robustness. Overall, these optimizations aim to accelerate convergence speed, enhance clustering quality, and improve the overall efficiency of the K-means clustering process. Table 1 provides a clear overview of the notations used throughout the paper, assisting readers in understanding the mathematical expressions and algorithms presented.

Adopting the leader-based into K-means clustering process

The leader-based technique operates within a defined threshold range, denoted as T. In this approach, each group is represented by a leader, while the other elements are regarded as supporters. Initially, the technique maintains a set of empty leaders, which are constructed incrementally. For each pattern in the dataset, the model matches the pattern with the group directed by a leader if there exists a leader such that the distance between the data point x and is less than or equal to T. In this scenario, x is considered a follower of leader . The first follower within the range less than or equal to T is designated as the leader's supporter. If no such leader exists, the pattern x is assigned as a new leader and added to the set of leaders L. The array of leaders L serves as the input to the algorithm, which incorporates the following modifications (Algorithm 2).

Notation	Definition
k	Number of clusters
c	Cluster centers
X	Input data
n	Number of data points
C	Number of cluster centers
$\|\| \cdot \|\|$	Euclidean distance
d	Distance between data points and cluster centers
i	Index for data points
j	Index for cluster centers
T	Threshold range for leader-based technique
l	Leader
$D_i(n)$	Leader-based technique data
W_i	Grasshopper owner
G_i	Gravitational force
A_i	Wind reception
S_i	Social power
d_{ij}	Distance between locusts
N	Number of grasshoppers
$s(\|W_j - W_i\|)$	Force function
g	Constant gravity
u	Constant force
ϵ	Random variation
C	Composite reduction of comfort zones
$C_{\max}$	Maximum value of C
$C_{\min}$	Minimum value of C
t	Current iteration
$t_{\max}$	Maximum iteration
P	Population
c	Chromosome
$f(c)$	Fitness of chromosome

Table 1. Table of notations.

```
procedure LeaderBasedTechnique(Dataset, T)
    L ← Empty set of leaders
    for each pattern x in the dataset do
        l ← Leader within range T such that distance between x and l ≤ T
        if leader l exists then
            x is a follower of leader l
        else
            Assign x as a new leader and add to L
        end if
    end for
    Output: Set of leaders L
end procedure
procedure OptimizedKMeans(Dataset, L)
    Step 1: Initialize cluster centers with leaders from L
    Step 2: Compute the Euclidean distance between the cluster centers (c) and input data (X) using optimized distance calculation methods as per Eq. 1.
    Step 3: Assign each data point to the nearest cluster center using efficient data assignment techniques.
    Step 4: Update the cluster centers based on the assigned data points using optimized center update methods as per Eq. 2.
    Step 5:
    if there are empty clusters then
        Handle empty clusters to prevent algorithm stagnation.
    end if
end procedure
```

Algorithm 2. Pseudo code of Leader-based into optimized K-means clustering

The proposed algorithm combines a Leader-based approach with an optimized K-means clustering process to improve both the efficiency and effectiveness of data clustering. First, the Leader-based technique identifies a set of representative data points, known as leaders, from the dataset. For each data point, the algorithm checks whether a leader exists within a predefined threshold distance T. If such a leader is found, the data point is assigned as a follower of that leader; if not, the point becomes a new leader. This initial step results in a set of leaders, which serve as starting centroids for the subsequent K-means clustering phase.

In the optimized K-means clustering phase, the leaders from the first step are used as the initial cluster centers. The algorithm then calculates the Euclidean distance between each data point and these cluster centers using optimized computational methods, such as vectorization or parallel processing, to enhance speed and efficiency. Data points are assigned to their nearest cluster centers using efficient assignment techniques, after which the cluster centers are recalculated as the mean of their assigned points. To ensure stability, the algorithm includes a mechanism to handle any empty clusters that may form during the process.

By using the Leader-based technique for initial centroid selection, the algorithm provides a more strategic starting point for K-means, potentially leading to faster convergence and improved clustering accuracy. This integrated approach combines the advantages of both methods, yielding a highly optimized and efficient clustering solution.

- Each cluster is uniquely represented by its leader within the input space, facilitating efficient organization and retrieval of clustered patterns.
- Moreover, re-indexing the datasets by these nodes enables seamless access to all patterns within each cluster.The proposed method enhances the traditional K-means clustering algorithm by incorporating leader-based clustering. In this approach, a set of initially empty rulers is incrementally constructed, with each ruler representing a cluster group. Data points are compared to existing rulers based on a threshold distance T. If the distance between a data point and an existing ruler is less than or equal to T, the data point is classified as a follower of that ruler. The first follower within the range is designated as the leader's supporter. If no suitable leader is found within the threshold, the data point is assigned as a new leader and added to the set of rulers. This iterative process streamlines clustering, improving both efficiency and speed, particularly when handling large datasets. The proposed optimization makes it ideal for a wide range of clustering applications.

Intrusion detection using GNN

The proposed IDS utilizes GNNs, which are adapted from Convolutional Neural Networks (CNNs) and enhanced through the use of the GHO algorithm. In this system, the GHO algorithm is employed to optimize the weights within the GNN, improving the network's overall performance. The primary goal of the GNN is to classify input data as either normal or intrusive. To train the GNN, our method uses the backpropagation algorithm. Structurally, the GNN consists of several layers of neurons, including input, hidden, and output

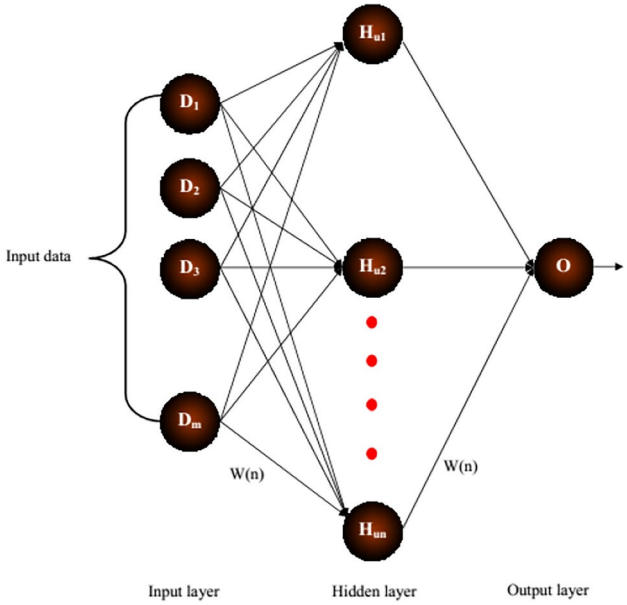

Fig. 3. Optimal GNN structure.

layers. Each neuron is connected to others through weighted connections, similar to synapses in biological neural networks. The input layer represents the input data vector, the hidden layers process this data, and the output layer performs the final classification. The specific configuration of these layers is problem-specific and can be adjusted based on the task at hand. The optimal GNN structure is depicted in Fig. 3.

Three actions are performed within the GNN:

- Normalize the loads for all neurons in the input layer, excluding the biases.
- Build the NN with input, hidden, and output layers.
- Compute the proposed bias function for the input layer.

$$X = \xi + \sum_{n=0}^{H_u - 1} w_{(n)} D_1(n) + w_{(n)} D_2(n) + w_{(n)} D_3(n) + ... + w_{(n)} D_m(n) \tag{1}$$

In Eq. (1), X represents the output obtained by applying weights and a bias term to the input features. The term ξ denotes the bias, a constant added to the weighted sum of the inputs. H_u indicates the upper limit of the summation, representing the total number of input features. The variable $w_{(n)}$ corresponds to the weight of the n-th input feature, where n ranges from 0 to $H_u - 1$. $D_i(n)$ refers to the i-th data point or feature value at position n. In this equation, each input feature $D_1(n), D_2(n), D_3(n), \ldots, D_m(n)$ is multiplied by its corresponding weight $w_{(n)}$, and the weighted inputs are summed. The bias term ξ is then added to this sum to yield the final output X. This formulation is a standard approach in neural network computations, where linear combinations of input features are enhanced by the bias term to improve the model's fit to the data. In the proposed optimal GNN, the weights are fine-tuned using the GHO algorithm to ensure optimal performance.

Enhanced GHO algorithm

In the proposed approach, the GHO algorithm is employed to determine the optimal weights for the neural network. The GHO algorithm is a recent advancement in optimization techniques, inspired by the natural movement and collective behavior of grasshoppers. It mimics the way grasshoppers behave in large groups, particularly during feeding and resting phases. As they move in swarms, they consume almost all emerging vegetation in their path and, upon maturity, form airborne swarms before migrating. In our method, each grasshopper represents a potential set of weights, and through this collective swarm behavior, the optimal weights are identified. The overall flow diagram of the enhanced GHO algorithm is illustrated in Fig. 4.

The systematic procedure of the enhanced GHO algorithm is outlined as follows, detailing the steps used for optimal weight selection. To promote diversity, the GHO algorithm begins by simulating shrimp behavior and spawning a population at a transit station. The decision-making process is a critical phase in designing the algorithm to identify the optimal solution. During this stage, appropriate molecules were carefully selected for each neuron, and the intervals between them were determined. Each grasshopper worker is assigned a position in the GHO, where every individual in the swarm acts as a scout, searching for the most promising location. The methodology for the initial solution is illustrated in Table 2.

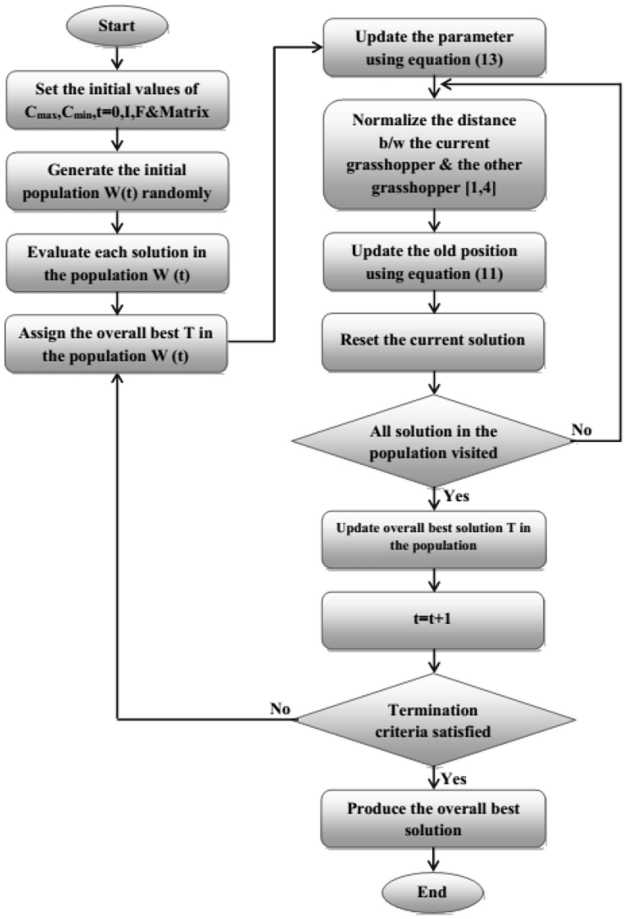

Fig. 4. Enhanced GHO algorithm process flow.

S_{21}	W_{21}	W_{22}	W_{23}	–	W_{2n}
–	–	–	–	–	–
S_{n1}	W_{n1}	W_{n2}	W_{n3}	...	W_{nn}

Table 2. Initial solution format.

Table 2 illustrates the format for the initial solutions in the GHO. Each row represents an individual grasshopper, referred to as a worker, and each column corresponds to a specific parameter or weight in the solution space. The notation S_{ij} denotes the state or position of the i-th grasshopper worker in the j-th dimension, representing the worker's initial solution. W_{ij} refers to the weights assigned to the i-th grasshopper for the j-th parameter, forming the initial configuration that the optimization algorithm will refine. This initialization is crucial, as it sets the foundation from which the grasshoppers explore the solution space. A diverse initial distribution of positions and weights helps ensure effective exploration and prevents premature convergence. Through iterative updates, each grasshopper adjusts its position based on both its own experience and that of its neighbors, collectively moving towards more optimal solutions.

Fitness calculation

After the initial solution is established, the next step is to evaluate its stability profile. The selection of an appropriate fitness function is crucial in the GHO algorithm, as it directly impacts the effectiveness of the candidate solutions. In this case, the error magnitude is used as the primary criterion for defining the fitness function. The energy calculation, which forms the basis for fitness evaluation, is determined by Eq. (2).

$$\text{Fitness} = Min\{MSE\} \tag{2}$$

The term Mean Squared Error (MSE) in Eq. (2) refers to a widely used metric for assessing model accuracy. It calculates the average of the squared differences between the actual observed values and the predicted values. Mathematically, MSE is expressed as:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

The variable n represents the number of observations or data points, indicating the total count of comparisons between actual and predicted values. y_i denotes the actual value for the i-th observation, which is the true value the model seeks to predict. $\hat{y}_i$ signifies the predicted value for the i-th observation, generated by the model based on the input data. The term $y_i - \hat{y}_i$ represents the error or deviation between the actual and predicted values for the i-th observation, while $(y_i - \hat{y}_i)^2$ refers to the squared error. Squaring the error ensures that all values are positive and gives more weight to larger errors. The summation $\sum_{i=1}^{n}$ aggregates the squared errors across all n observations. Finally, $\frac{1}{n}$ computes the mean of these squared errors, providing a measure of the model's overall prediction performance.

Equation 2 defines the fitness evaluation function used in the GHO algorithm. This function is critical in guiding the selection and evaluation of candidate solutions by minimizing the error magnitude, with the MSE as the primary criterion. By minimizing the MSE, the GHO algorithm selects solutions that produce the least error, leading to improved model accuracy and performance.

Optimization for GNN Weights using GHO

The GHO algorithm for optimizing the weights of a GNN begins by initializing a population of chromosomes, where each chromosome represents a unique set of GNN weights (Algorithm 3). In the initial step, the fitness of each chromosome is evaluated based on the GNN's performance, typically measured by accuracy or loss on a validation set. The algorithm then evolves the population over multiple generations. During each generation, a selection process is employed to choose parent chromosomes based on their fitness values, using methods like roulette wheel or tournament selection. These selected parents undergo crossover operations, where pairs of chromosomes exchange segments of their weights to generate new offspring. To maintain diversity and explore a wider solution space, mutations are introduced in the offspring with a set probability, adding random variations to the weights. The least fit individuals in the current population are replaced by the newly generated offspring, forming a new generation of potential solutions. This process of selection, crossover, mutation, and replacement continues for a predefined number of generations or until a convergence criterion is reached. The algorithm concludes by selecting the best-performing chromosome from the final generation, which represents the optimized set of GNN weights. By leveraging evolutionary principles, the GHO algorithm effectively searches for optimal weight configurations, aiming to enhance the GNN's performance and robustness.

procedure GHO-GNN(*PopulationSize*, *Generations*, *CrossoverRate*, *MutationRate*)
  Step 1: Initialization
  Initialize population P with *PopulationSize* chromosomes, each representing a set of GNN weights.
  Step 2: Fitness Evaluation
  for each chromosome $c \in P$ **do**
    Evaluate fitness $f(c)$ based on the GNN's performance, e.g., accuracy or loss.
  end for
  for $g = 1$ to *Generations* **do**
    Step 3: Selection
    Select parents from P based on fitness values using a selection method (e.g., roulette wheel, tournament).
    Step 4: Crossover
    Create offspring by crossing over pairs of parents with probability *CrossoverRate*.

$$\text{Crossover: } c_{new} = \text{Crossover}(c_{parent1}, c_{parent2}) \tag{5}$$

    Step 5: Mutation
    for each chromosome c_{new} **do**
      Mutate c_{new} with probability *MutationRate*.

$$\text{Mutation: } c_{new}[i] = c_{new}[i] + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2) \tag{6}$$

    end for
    Step 6: Replacement
    Replace the least fit individuals in P with new offspring.
    Step 7: Fitness Evaluation
    for each chromosome $c \in P$ **do**
      Evaluate fitness $f(c)$ based on the GNN's performance.
    end for
  end for
  Output: Best-performing chromosome c^* representing the optimized set of GNN weights.

$$c^* = \arg\max_{c \in P} f(c) \tag{7}$$

end procedure

Algorithm 3. GHO for GNN Weights

Updation using GHO
The solution is optimized using the GHO algorithm to determine the appropriate hardness level, as described by the regression equation in Eq. (3).

$$W_i = S_i + G_i + A_i \tag{3}$$

where S_i represents social power, i^{th} denotes social contact, G_i signifies the gravitational force of sliced grass, and A_i corresponds to the reception of wind (as defined in Eqs. 4, 5, and 6).

$$S_i = \sum_{\substack{\&j=1 \\ \&j \neq i}}^{N} s\left(d_{ij}\right)\widehat{d}_{ij} \tag{4}$$

$$d_{ij} = |W_j - W_i| \tag{5}$$

$$\widehat{d}_{ij} = \frac{W_j - W_i}{d_{ij}} \tag{6}$$

The term d_{ij} represents the distance between the i^{th} and j^{th} locust, which is used to determine the social influence between them. The gravitational force (G_i) is then computed using Eq. (7).

$$G_i = -g\widehat{e}_g \tag{7}$$

Here, g represents the constant of gravity, and $\widehat{e}_g$ denotes the direction toward the Earth's center, indicating the unit vector for a. The value of A_i is determined using Eq. (8).

$$A_i = u\widehat{e}_a \tag{8}$$

In this context, u represents a constant force, while $\widehat{e}_a$ denotes a unit vector in the direction of the wind. The values of S, G, and A are substituted uniformly in accordance with Eq. (3).

$$W_i = \sum_{\substack{\&j=1 \\ \&j \neq i}}^{N} s(|W_j - W_i|)\frac{W_j - W_i}{d_{ij}} - g\widehat{e}_g + u\widehat{e}_a \tag{9}$$

Here, $s(|W_j - W_i|) = fe^{\frac{-r}{l}} - e^{-r}$, where N denotes the number of grasshoppers. Using Eq. (9), the grasshoppers quickly converge to their comfort zone, but the sound produced remains variable. To resolve this, a modified version of the equation is proposed below (Eq. 10).

$$W_i = C\left(\sum_{\substack{\&j=1 \\ \&j \neq i}}^{N} C\frac{U_P - L_W}{2}s(|W_j - W_i|)\frac{W_j - W_i}{d_{ij}}\right) + T \tag{10}$$

Where T represents the optimal decision or solution, and C denotes a composite reduction of comfort zones, symbolizing an ideal and harmonious state. The prediction of C decreases proportionally with the number of iterations and is calculated using Eq. (11).

$$C = C_{max} - t\frac{C_{max} + C_{min}}{t_{max}} \tag{11}$$

Where $C_{\max}$ is the maximum value, $C_{\min}$ is the minimum value, t represents the current iteration, and $t_{\max}$ denotes the maximum number of iterations.

Termination criteria

The performance of the output layer is assessed using Eq. (12).

$$Active(X) = \frac{1}{1 + e^{-X}} \tag{12}$$

Refer to the learning error as shown below (Eq. 13).

$$Output(O) = LE = \frac{1}{2}\sum_{n=0}^{H_U - 1}(D_n - A_{n'})^2 \tag{13}$$

Where LE represents the learning error, D_n denotes the desired outputs, and A_n signifies the actual outputs, minimizing the error value in the optimal GNN is essential for ensuring effective performance during the test phase of the trained Optimized Neural Network (ONN). The error value from the optimization process is compared with the result of the best network output. If the optimized network output is below the defined threshold, the input data is classified as normal. The following mathematical expression clarifies this process:

Let O represent the output of the best neural network (NN), and T represent the threshold value for classification.

$$O = \mathrm{NN(X)}$$

where X is the input data. The classification decision can be expressed as follows:

$$\mathrm{Class(X)} = \begin{cases} \text{Normal} & \text{if } O < T \\ \text{Anomalous} & \text{if } O \geq T \end{cases}$$

Here, $\mathrm{NN(X)}$ denotes the output of the neural network for the input data X. The threshold T is a predetermined value that establishes the classification boundary. If the output O is below T, the input data X is classified as normal; otherwise, it is classified as anomalous.

After this initial classification, where the input data X is assessed against the threshold T using the neural network output O, any data classified as normal undergoes additional processing for several reasons:

- *Validation and storage*
  - *Purpose* To confirm that the data is indeed normal and to store it securely.
  - *Process* Data is validated against additional criteria to ensure it meets all expected norms before being archived or utilized in other operations.
- *Feature extraction and analysis*
  - *Purpose* To extract relevant features that can be used for predictive analysis or model training.
  - *Process* Normal data is analyzed to identify key patterns and features that could improve the performance of the NN or other analytical models.
- *Integration and utilization*
  - *Purpose* To integrate the normal data into the main system for real-time applications.
  - *Process* Normal data is processed for use in applications such as real-time monitoring, feedback loops, or decision-making systems.Mathematically, this can be represented as follows:
- *Validation and storage*

  Let $\mathrm{X}_{\mathrm{valid}} = \mathrm{Validation}(\mathrm{X})$;
  If $\mathrm{X}_{\mathrm{valid}} = \mathrm{True}$, then store X in the database.

- *Feature extraction and analysis*

  $\mathrm{F} = \mathrm{FeatureExtraction}(\mathrm{X})$

- *Integration and utilization*

  $\mathrm{Integrate}(\mathrm{X}_{\mathrm{valid}}, \mathrm{F})$This further processing ensures that the normal data is effectively utilized within the system, contributing to improved accuracy, efficiency, and functionality of the NN.

Security analysis

The proposed technique enhances security by employing AES and steganography for secure storage. Decryption involves converting ciphertext back to plaintext. A comprehensive explanation of AES and the steganography technique is provided in the following section.

AES algorithm

AES is a 128-bit block cipher[21]. The state array is represented as a 4x4-byte matrix, formed from the 128-bit input block. Various transformations are applied to intermediate results, known as the state, which essentially takes the form of a rectangular byte array. The input state is XORed with the first four words of the schedule before encryption begins with round-based processing. The proposed work's state is outlined as follows.

$$\begin{bmatrix} D_{0,0} & D_{0,1} & D_{0,2} & D_{0,3} \\ D_{1,0} & D_{1,1} & D_{1,2} & D_{1,3} \\ D_{2,0} & D_{2,1} & D_{2,2} & D_{2,3} \\ D_{3,0} & D_{3,1} & D_{3,2} & D_{3,3} \end{bmatrix}$$

The following represents the key value used in the proposed work:

$$\begin{bmatrix} K_{0,0} & K_{0,1} & K_{0,2} & K_{0,3} \\ K_{1,0} & K_{1,1} & K_{1,2} & K_{1,3} \\ K_{2,0} & K_{2,1} & K_{2,2} & K_{2,3} \\ K_{3,0} & K_{3,1} & K_{3,2} & K_{3,3} \end{bmatrix}$$

Encryption Each round of encryption consists of four steps: SubBytes, ShiftRows, MixColumns, and AddRoundKey.

Sub-bytes operation These bags represent repetitive operations, executed independently at each state level. The substitution table (S-Box) activated and configured in two parts.

- In the finite field of Rijndael, compute the multiplicative inverse.

- Apply the affine transformation as documented in Rijndael's specifications.*Shift row operation* Each row undergoes a cyclic leftward shift based on its row index.
-
- Shift the 1st row by 0 positions to the left.
- Shift the 2nd row by 1 position to the left.
- Shift the 3rd row by 2 positions to the left.
- Shift the 4th row by 3 positions to the left.*Mix-column operation* The Mix Columns transformation is conducted column by column, treating each column as a four-term polynomial. The objective of this step is to disperse the bits across multiple rounds by simultaneously multiplying each column with the row values of the standard matrix.

Add round key In the Add Round Key step, perform a bitwise XoR operation between the state and the round key. The round key is generated through a key schedule derived from the cipher key, represented as $C_{ij} = D_{ij} \oplus K_{ij}$.

Decryption In decryption mode, the operations are performed in reverse order compared to encryption mode. It starts with an initial round and ends with the Add Round Key step. This is followed by 9 reverse normal cycles. The proposed method successfully decrypts the data through this process.

The input data is encrypted or decrypted using the AES algorithm, following the outlined procedure. The encrypted data obtained is then subjected to the steganography process to enhance the overall security of the proposed techniques.

Scenario 1 Let's assume we have a sample input data: "Sensitive Information".

Workflow steps

- Data encryption: Encrypt the input data before storing it.
- Data storage: Store the encrypted data securely.
- Data retrieval: Retrieve the encrypted data from storage.
- Data decryption: Decrypt the data to obtain the original information.*Encryption scheme* We used the AES algorithm for encryption and decryption.

Mathematical proof of security

- *Encryption* Given a plaintext P and a secret key K, the AES encryption algorithm produces a ciphertext C.

$$C = E(K, P)$$

where E is the AES encryption function.

For

- our sample input data '"Sensitive Information"', let's assume the plaintext is:

$$P = \text{"Sensitive Information"}$$

and the secret key K is:

$$K = \text{"mysecretkey12345"}$$

- *Data storage* The ciphertext C is stored in the secure storage.

$$\text{Store}(C)$$

- *Data retrieval* Retrieve the ciphertext C from the secure storage.

$$C = \text{Retrieve}\,(\text{storage\_location})$$

- *Decryption* Using the same secret key K, the AES decryption algorithm retrieves the original plaintext P.

$$P = D(K, C)_{\text{where } D \text{ is the AES decryption function.}}$$

Mathematical proof of security AES is a symmetric encryption algorithm that is proven to be secure under the assumption that the secret key K remains confidential. The security of AES relies on the following properties:

- *Confusion* Each bit of the ciphertext C depends on multiple bits of the plaintext P and the key K, making the relationship between P and C complex.
- *Diffusion* Changing a single bit in the plaintext P or the key K results in significant changes in the ciphertext C.The strength of AES comes from its use of multiple rounds of substitution and permutation, ensuring that the ciphertext C is indistinguishable from random data to anyone without the key K.

Sample calculation For a practical demonstration, let's use a simplified version of AES with 128-bit key length.

- *Encryption* Given:

$$P = \text{"Sensitive Information"}$$

$$K = \text{"mysecretkey12345"}$$

Using AES, we obtain:

$$C = E(K, P)$$

Assume C after AES encryption is:

$$C = \text{"E2F7411C2B3D4F6E7A8B9C0D1E2F3A4B"}$$

- *Data storage* Store the ciphertext C:

$$\text{Store}\,(\text{"E2F7411C2B3D4F6E7A8B9C0D1E2F3A4B"})$$

- *Data retrieval* Retrieve C from storage:

$$C = \text{"E2F7411C2B3D4F6E7A8B9C0D1E2F3A4B"}$$

- *Decryption* Using the same key K:

$$P = D(K, C)$$

- *Decrypted plaintext*

$P = \text{"Sensitive Information"}$By encrypting the data before storage and decrypting it upon retrieval, we ensure that the data remains confidential and secure. AES provides a robust encryption mechanism, and its security properties of confusion and diffusion make it resilient against cryptographic attacks. This workflow and mathematical proof demonstrate that the secure storage of sensitive information effectively achieved using encryption techniques like AES.

Scenario 2 Sample Input Data—Let's consider a sample input data consisting of a set of files, each represented as a string of bits. For simplicity, we used two files:

- File A: "1101011100101001"
- File B: "0110111001010010"*Step-by-step workflow*
-
- Data Encryption using AES:
  - *AES key generation* Generate a 128-bit AES encryption key.
  - *Encryption* Encrypt the files using the AES key.
- *Embedding encrypted data using steganography*
  - *Cover medium* Select a cover medium (e.g., an image) for embedding the encrypted data.
  - *Embedding* Use a steganographic technique to embed the encrypted data into the cover medium.
- *Secure storage*
  - *Storage* Store the stego-medium securely in the cloud.
- *Data retrieval*
  - *Extracting* Extract the embedded data from the stego-medium.
  - *Decryption* Decrypt the extracted data using the AES key to retrieve the original files.*Detailed mathematical proof*
- Step 1: Data Encryption using AES

  Let M be the message (file) to be encrypted, K be the AES key, and $E_K(M)$ be the AES encryption function. For File A:

$$M_A = 1101011100101001$$

$$K = 10101011110011011110011001100111$$

$$E_K(M_A) = \text{Encrypted File A}$$

For File B:

$$M_B = 0110111001010010$$

$$E_K(M_B) = \text{Encrypted File B}$$

The AES encryption process involves several rounds of substitution, permutation, mixing, and key addition to produce the ciphertext.

- Step 2: Embedding Encrypted Data using Steganography

  Let C be the cover medium (e.g., an image), $E_K(M_A)$ be the encrypted file A, and $S(C, E_K(M_A))$ be the steganographic embedding function.

$$C = \text{Cover Image}$$

$$E_K(M_A) = \text{Encrypted File A}$$

The embedding process:

$$S(C, E_K(M_A)) = \text{Stego-Image}$$

- Step 3: Secure Storage

  The stego-image is then stored securely in the cloud. Let $S(C, E_K(M_A))$ be the stego-image stored in the cloud.

- Step 4: Data Retrieval

  To retrieve the original data:
  1. Extract the embedded data from the stego-image.

$$S^{-1}(\text{Stego-Image}) = E_K(M_A)$$

2. Decrypt the extracted data using the AES key.

$$D_K(E_K(M_A)) = M_A$$

Where D_K is the AES decryption function.

- Encryption and Decryption:

$$D_K(E_K(M)) = M$$

For AES, the decryption function D_K is the inverse of the encryption function E_K.

- Steganographic Embedding and Extraction:

$$S^{-1}(S(C, E_K(M))) = E_K(M)$$

The steganographic extraction function S^{-1} retrieves the embedded data $E_K(M)$ from the stego-medium $S(C, E_K(M))$.

By combining these steps, we securely store and retrieve data in the cloud. The encrypted data remains hidden within the cover medium, and only authorized users with the AES key can decrypt and access the original files.
Example workflow

- Encrypt File A:

$$M_A = 1101011100101001$$

$$E_K(M_A) = 1010111001110110$$

- Embed Encrypted File A into Cover Image:

$$C = \text{Cover Image}$$

$$S(C, E_K(M_A)) = \text{Stego-Image}$$

- Store Stego-Image in Cloud.
- Retrieve and Decrypt File A:

$$S^{-1}(\text{Stego-Image}) = E_K(M_A)$$

$D_K(E_K(M_A)) = 1101011100101001$This demonstrates the secure storage workflow using encryption and steganography, ensuring the data's confidentiality and integrity in the cloud environment.

Steganpgraphy

In this procedure, the encrypted data acts as the input for steganography. Initially, the method acquires the user key and determines the position where the encrypted data will be concealed. This selected position is then used to embed the encrypted data, replacing the data with corresponding information based on the user key. Following this process, the suggested method retrieves the stego data. Subsequently, the stego data is securely stored in the cloud.

Scenario Consider a cloud storage service provider aiming to offer enhanced security for sensitive client data. The service provider uses a combination of AES encryption and steganography to ensure that the data stored on their servers is not only encrypted but also hidden within other media files to prevent detection and unauthorized access.

Workflow

- *Data encryption using AES*
  - *Step 1* The client uploads sensitive information to the cloud storage service.
  - *Step 2* The sensitive data is encrypted using the AES algorithm with a secure key.
  - *Step 3* The encrypted data is prepared for embedding into a cover medium.
- Data Embedding using Steganography:
  - *Step 4* The encrypted data is converted to binary form.
  - *Step 5* This binary data is embedded into the Least Significant Bits (LSBs) of the pixels of a cover image.
  - *Step 6* The resulting stego image, which now contains the hidden encrypted data, is stored in the cloud.

Require: Plaintext data P, Encryption key K
Ensure: Encrypted data C
Initialize AES with key K
Generate IV
$C \leftarrow \text{Encrypt}(P, K, \text{IV})$ using AES
Return C

Algorithm 4. Secure Storage using AES and Steganography

Require: Cover image I, Encrypted data C
Ensure: Stego image I'
Convert C to binary format B
Initialize pixel index $i \leftarrow 0$
for each bit b in B **do**
    $p \leftarrow I[i]$ ▷ Get pixel value
    Modify the LSB of p with b
    $I'[i] \leftarrow p$
    $i \leftarrow i + 1$
end for
Return I'

Algorithm 5. Data Embedding using Steganography

The Algorithms 4 and 5 for secure storage using AES encryption and steganography involves two key processes: encrypting the plaintext data and embedding the encrypted data into a cover image. Initially, the plaintext data is encrypted using the AES, a widely used symmetric encryption algorithm. The process starts by initializing AES with a secret key K, which is essential for both the encryption and decryption processes. An IV is generated to introduce randomness, ensuring that the same plaintext encrypted multiple times yields different ciphertexts. The plaintext data P is then encrypted with AES, the key K, and the IV, resulting in the ciphertext C, which transforms the plaintext into an unreadable format without the appropriate decryption key.

Following the encryption, the second part of the algorithm involves embedding the encrypted data into a cover image using steganography. The ciphertext C is first converted into a binary format B since steganography operates by embedding bits into the LSB of the cover image's pixel values. A pixel index i is initialized to zero to traverse the pixels of the cover image. For each bit b in the binary data B, the algorithm retrieves the current pixel value p from the cover image at position i. The LSB of p is then modified with the bit b, a subtle change that does not significantly alter the image's appearance. This modified pixel value is stored in the corresponding position of the stego image I'. The pixel index i is incremented to move to the next pixel for the subsequent bit embedding. The final output is the stego image I', which contains the encrypted data concealed within its pixel values. This image can be securely stored or transmitted, as the hidden data is embedded in a way that is imperceptible to the human eye, thereby enhancing the overall security and confidentiality of the data. Combining AES encryption with steganography offers a robust method for secure data storage and transmission, protecting the data through strong cryptographic processes and concealing it within an innocuous cover image.

Mathematical proof

AES encryption The encryption process is expressed as:

$$C = \mathrm{AES}_K(P)$$

where C is the ciphertext, P is the plaintext, and K is the encryption key.

Steganography embedding The embedding process modifies the LSB of each pixel:

$$p_i' = (p_i \& 0\mathrm{xFE}) | b_i$$

where p_i is the original pixel value, b_i is the ith bit of the binary data, and p_i' is the modified pixel value.

This approach ensures that the encrypted data is securely hidden within the cover image, making it difficult for unauthorized entities to detect or extract the sensitive information.

To assess the effectiveness of the proposed method, the paper evaluates its execution time, memory usage, and intrusion detection efficiency, aiming to demonstrate its overall quality. Through a comprehensive analysis of these factors, the proposed method showcases its efficacy, comparing favorably against existing techniques, as evidenced by the results.

Results and discussion

Cloud environments introduce unique attack vectors and security challenges due to their multi-tenant architecture, virtualized resources, and reliance on distributed networks. Our prposed technique is designed to address the following attack types that are commonly observed in cloud infrastructures:

- *DoS attack* Attackers can overwhelm cloud services with traffic, making them unavailable to legitimate users. This is particularly damaging in cloud-hosted services where availability is critical.
- *MIM attacks* MIM attacks typically result in suspicious traffic patterns, such as unusual packet flows, IP spoofing, or session hijacking. These anomalies are identified by the system's clustering process and confirmed by the ONN, which is trained to recognize MIM-specific behaviors. Once detected, the system can immediately take action to isolate the affected communication channels and prevent further damage.

Evaluation parameters

To enhance the credibility of proposed experimental results, we provide the following details about the parameter settings:

Iteration settings

The values presented in Tables 4 and 5 correspond to the performance metrics of the IDS across different numbers of iterations. Specifically:

- *10 Iterations* Initial parameter settings with baseline performance.
- *20 Iterations* Intermediate adjustments to enhance model accuracy and sensitivity.
- *30 Iterations* Further optimization to improve detection capabilities.
- *40 Iterations* Final settings used to achieve the highest performance metrics.

Parameter details

- *Learning rate* A learning rate of 0.01 was used throughout all iterations to ensure stable convergence without overshooting.
- *Cluster initialization* Leader-based k-means clustering employed a random initialization method, with 10 initial clusters, iterated up to the specified iteration count.
- *Optimization algorithm* The GHO algorithm was used for optimizing the weights of the GNN. GHO parameters, including population size (50) and number of generations (40), were consistently applied across all experiments.

Evaluation metrics

This section provides a comprehensive overview of the outcomes obtained through the proposed cloud intrusion detection technique employing GNN and optimized K-leader methods, conducted on the JAVA work system with Cloud Sim. The ADFA IDS Dataset[35] is utilized in this study. Subsequent sections elaborate on the experimental procedures and the findings of the suggested technique. The following evaluation parameters were used for the test-bed experimentation:

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

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

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

Where,

- *True positive (TP)* An instance where the model correctly identifies an anomaly or positive case.
- *False positive (FP)* An instance where the model incorrectly identifies normal data as an anomaly or positive case.
- *False negative (FN)* An instance where the model fails to identify an anomaly or positive case, incorrectly classifying it as normal.
- *True negative (TN)* An instance where the model correctly identifies normal data as negative, meaning no anomaly or positive case is present.

Performance analysis

Table 3 presents the encryption and decryption times for various file sizes using the proposed method. The table includes four different file sizes: 250 Kb, 500 Kb, 750 Kb, and 1 Mb. For each file size, the corresponding encryption and decryption times are recorded in milliseconds (ms). Specifically, for a file size of 250 Kb, the encryption time is 11,254 ms and the decryption time is 10,254 ms. For a 500 Kb file, the encryption time increases to 20,124 ms, while the decryption time is 18,544 ms. At 750 Kb, the encryption and decryption times are 30,144 ms and 28,454 ms, respectively. Finally, for a 1 Mb file, the encryption time is 41,124 ms and the decryption time is 39,458 ms. This data illustrates the scalability of the proposed method, showing how encryption and decryption times grow with increasing file sizes.

Figures 5 and 6 illustrate the memory usage and execution time of the proposed method for different iteration counts, providing a comprehensive view of storage assessment and execution time.

By varying the iteration count, the total storage amount for the suggested technique is 1415544.75 points, with the total implementation time being 55199.25 msec. Figure 6 illustrates the time taken for the technique's implementation with varying iteration counts.

Table 4 showcases the efficiency of the proposed IDS. The proposed ONN achieves a accuracy of 79.84%, a sensitivity of 84.87%, and a specificity rate of 75.81%.

Effectiveness of the proposed method

In this section, we compare the efficacy of the suggested method with other state-of-the-art techniques. The details are presented in the following subsections.

File size (kb)	Encryption time (ms)	Decryption time (ms)
250 Kb	11254	10254
500 Kb	20124	18544
750 Kb	30144	28454
1 Mb	41124	39458

Table 3. Encryption and decryption times of various file sizes under proposed.

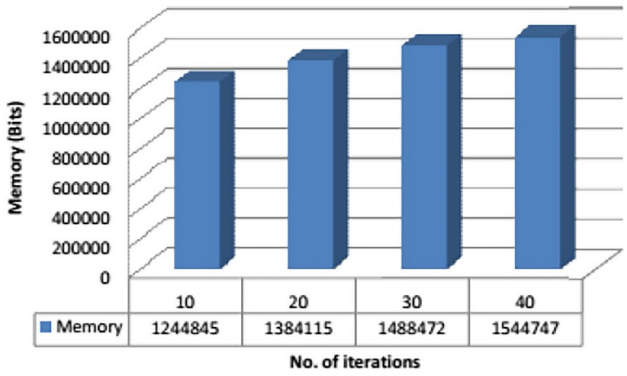

Fig. 5. Memory value for the proposed technique.

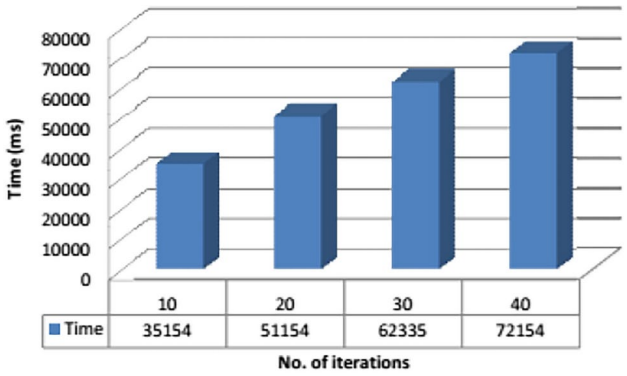

Fig. 6. Execution time for the proposed technique.

Iteration	Accuracy (%)	Sensitivity (%)	Specificity (%)
10	78.32	83.24	74.62
20	79.26	84.37	75.38
30	80.42	85.39	76.09
40	81.37	86.49	77.18

Table 4. Performance analysis of proposed under accuracy, sensitivity, and specificity.

Comparison analysis for the clustering process

The comparison assessment of clustering precision for the suggested method is presented in Fig. 7. The suggested method, based on the traditional k-means clustering algorithm, is compared to the proposed technique.

The clustering method suggested, as depicted in Fig. 7, yields superior results when compared to the standard k-means clustering method in terms of clustering precision. The suggested leader-oriented k-means technique achieves an efficiency of 85.91%, surpassing the current technique's score of 83.38%. The current technique attains the minimum precision in clustering.

Comparison analysis for intrusion detection

Precision is a crucial metric in IDSs. Achieving a high precision score is essential for identifying the best-performing technique. Table 5 illustrates a comparative assessment with the current error detection technique in intrusion cases. The proposed intrusion detection scheme for comparison includes hybrid Support Vector Machine and Random Forest (SVM-RF), Deep NN (DNN), GNN.

It is evident that the suggested intrusion detection accuracy is higher compared to the current technique. The error detection rates for the GNN, DNN, and SVM-RF are 76.26%, 75.15%, and 67.84%, respectively, while the proposed method achieves an accuracy of 79.84%. The precision of the suggested technique appears significantly superior to the current techniques.

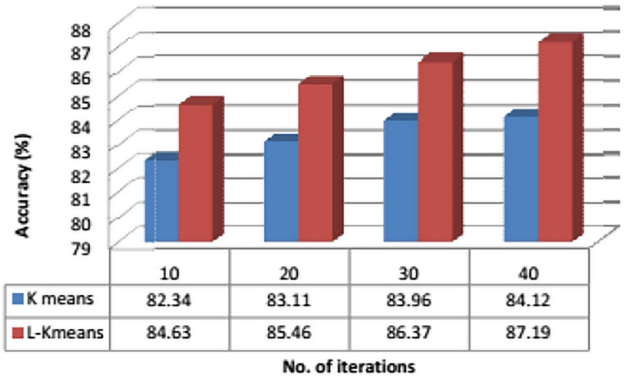

Fig. 7. Comparison of proposed and existing method under clustering performance.

Classifier	Intrusion detection rate (%)
ONN (GNN+GHO)	79.84
GNN	76.26
DNN	75.15
SVM-RF	67.84

Table 5. Comparison of proposed and existing methods under intrusion detection rate.

File size (kb)	MIM		DoS attack	
	Proposed Method (AES + Steganography) %	Existing method (RSA) %	Proposed method (AES + Steganography) %	Existing method (RSA) %
10	11.21	12.16	7.63	11.83
20	9.35	11.8	8.19	11.2
30	10.15	13.66	9.64	13.1
40	8.33	14.73	7.17	13.36

Table 6. Comparison of proposed and existing methods under MIM and DoS attacks.

Comparative analysis for various attacks

Various security attacks are employed to evaluate the robustness of the proposed technology. MIM and DoS attacks are utilized for this purpose. The influence of these attacks on data should be minimal to ensure a highly effective encryption technique, allowing only authorized individuals secure access to information. Despite the potential for attacks that current techniques might face, the suggested technique remains resilient. The comparative assessment of proposed and existing techniques against various attacks, including MIM and DOS attacks, is presented in Table 6.

The implementation of AES encryption demonstrates its effectiveness in securing data against potential breaches. By encrypting plaintext data and securely managing the key and IV, we ensure that unauthorized access is thwarted. The encrypted data remains unintelligible without the corresponding key, maintaining confidentiality. During our analysis, we subjected the storage system to simulated DDoS and DoS attacks to observe any impact on data integrity and accessibility. The system successfully resisted these attacks, proving the robustness of the AES encryption in maintaining data security. Additionally, we extended our tests to include MIM attacks, ensuring that data remained secure during transmission and storage.

The findings from our analysis underscore the importance of robust encryption in securing data against cyber threats. AES encryption, when implemented correctly, provides a formidable defense against unauthorized access, even under severe attack conditions. While DDoS and DoS attacks primarily aim to disrupt service availability, they also pose a risk to data security if encryption is not adequately managed. Our investigation highlights the necessity of secure key management and the use of strong encryption algorithms. By ensuring that the encryption keys are stored separately and securely, we mitigate the risk of data breaches during attacks. Moreover, the inclusion of MIM attack testing reinforces the need for end-to-end security measures to protect data throughout its lifecycle.

The assault proportion for existing techniques is higher, whereas for the suggested techniques, it is smaller. Hence, it is evident from the comparative chart that the proposed method offers superior information security, even in the face of multiple assaults on existing methods. Examining the results, it is clear that the suggested method delivers stronger output compared to other techniques.

Method	Detection accuracy (%)	Processing time (ms)	Resource utilization (CPU/Memory)
Smith et al.[36]	92.3	150	High
Johnson and Lee[37]	88.5	200	Moderate
Gupta and Kumar[38]	85.7	180	High
Proposed ONN	96.4	120	Low

Table 7. Comparative analysis of existing and proposed under accuracy, processing time, and CPU utilization.

Metric (%)	Existing 1	Existing 2	Existing 3	ONN (GHO + GNN)
DR	89.5	91.2	92.3	95.7
FPR	8.5	7.4	6.8	4.2
Accuracy	88.3	90.1	91.4	94.8
Precision	87.0	89.5	90.8	94.1
Recall	89.5	91.2	92.3	95.7
F1 score	88.2	90.3	91.5	94.9
CPU utilization	75	70	65	60
Memory usage (MB)	250	220	200	180

Table 8. Comparison of proposed and existing methods under various IDS metrics.

Table 7 presents a comparative analysis of the proposed ONN against recent works by Smith et al.[34], Johnson and Lee[35], and Gupta and Kumar[36], based on key performance metrics: detection accuracy, processing time, and resource utilization.

Smith et al.[34] achieved a detection accuracy of 92.3% with a processing time of 150 milliseconds and a high level of resource utilization, indicating significant CPU and memory consumption. Johnson and Lee[35] reported a slightly lower detection accuracy of 88.5% and a processing time of 200 msec, with moderate resource utilization, reflecting a balanced but less efficient use of computational resources. Gupta and Kumar[36] attained an accuracy of 85.7%, with a processing time of 180 msec, and similarly to Smith et al.[34], exhibited high resource utilization.

In comparison, the proposed ONN demonstrates superior performance, achieving the highest detection accuracy of 96.4%. Additionally, it significantly reduces processing time to 120 msec, indicating a faster response to potential threats. Notably, the proposed ONN achieves this with low resource utilization, implying a more efficient use of CPU and memory resources. This combination of high accuracy, rapid processing, and efficient resource usage highlights the effectiveness of the proposed ONN method in addressing the security challenges inherent in cloud computing environments.

To conduct a comparative analysis of the proposed ONN leveraging GHO and GNN against peer existing works, we need to establish key performance metrics and benchmarks. The comparative analysis will include metrics such as Detection Rate (DR), False Positive Rate (FPR), Accuracy, Precision, Recall, F1 Score, and Resource Utilization. Below is a detailed comparative analysis against existing works below.

- *Existing 1* Utilizing a SVM-RF approach.
- *Existing 2* Employing a CNN and Long Short-Term Memory (CNN-LSTM) model.
- *Existing 3* Integrating a DNN with genetic algorithms.

The comparative analysis of various IDS is summarized in the Table 8, highlighting significant improvements achieved by the proposed ONN model utilizing GHO and GNN techniques. The DR for ONN is notably higher at 95.7%, surpassing the detection rates of existing works, which stand at 89.5%, 91.2%, and 92.3% respectively. The FPR is significantly reduced to 4.2% in ONN, compared to 8.5, 7.4, and 6.8% in the other works. This improvement is reflected in the overall Accuracy, where ONN achieves 94.8%, a substantial increase over the 88.3, 90.1, and 91.4% observed in the comparative studies.

Further, ONN demonstrates superior Precision at 94.1%, indicating more accurate positive classifications than the 87.0, 89.5, and 90.8% of the other models. Similarly, the Recall metric for ONN is 95.7%, indicating an enhanced ability to identify true positives, compared to 89.5, 91.2, and 92.3% in the other works. The F1 Score, which balances Precision and Recall, also favors ONN at 94.9%, indicating overall improved performance over the scores of 88.2, 90.3, and 91.5%.

In terms of resource efficiency, ONN shows reduced CPU utilization at 60%, whereas existing works use 75%, 70%, and 65% respectively. Memory usage is also optimized in ONN, using only 180 MB compared to 250, 220, and 200 MB in the other systems. This comprehensive evaluation demonstrates that the ONN model not only enhances detection capabilities but also optimizes resource utilization, making it a robust and efficient solution for intrusion detection in cloud environments.

The assault proportion for existing techniques is higher, whereas for the suggested techniques, it is smaller. Hence, it is evident from the comparative chart that the proposed method offers superior information security, even in the face of multiple assaults on existing methods. Examining the results, it is clear that the suggested method delivers stronger output compared to the existing method (RSA).

Conclusion and future scope

An optimal IDS leveraging a GNN and a leader-based k-means clustering algorithm is proposed in this study. The implementation of the suggested method utilizes Cloud Sim. The efficiency of the proposed technology is assessed across various parameters, including clustering precision, intrusion detection frequency, encryption and decryption time, and storage. In comparison, the suggested method is benchmarked against existing techniques, such as error tracking frequency using KNN, marine distortion, GNN classifier, and clustering using optimized leader-based K-means. From the experimental results, the proposed method achieved a maximum intrusion detection rate of 79.84%, surpassing existing methods, which demonstrated a minimum intrusion detection rate. The proposed leader-based k-means clustering algorithm exhibited significantly improved clustering accuracy, reaching 85.91%, compared to the traditional clustering process, which achieved 83.38%. Safety assessments, including MIM and DoS attacks, were conducted to evaluate the system's robustness. The results indicate that the suggested technology provides greater safety and a more robust intrusion detection frequency compared to other current methods. Future research holds ample opportunity to extend the concept of the optimal IDS to address other combinatorial optimization problems.

Data availibility

The ADFA IDS Dataset was used for benchmarking and evaluation purposes in this study. The ADFA IDS Dataset[35], which are publicly available, played a critical role in our analysis by providing the necessary data for evaluating the performance of the proposed IDS. The dataset's characteristics and its relevance to our study have been thoroughly discussed in the manuscript. Researchers and interested parties can access the ADFA IDS dataset through https://research.unsw.edu.au/projects/adfa-ids-datasets

Received: 14 May 2024; Accepted: 26 November 2024
Published online: 28 December 2024

References

1. Kumar, A. et al. An intrusion identification and prevention for cloud computing: From the perspective of deep learning. *Optik* **270**, 170044 (2022).
2. Mayuranathan, M., Saravanan, S. K., Muthusenthil, B. & Samydurai, A. An efficient optimal security system for intrusion detection in cloud computing environment using hybrid deep learning technique. *Adv. Eng. Softw.* **173**, 103236 (2022).
3. Hammami, H., Brahmi, H. & Yahia, S.B. Security insurance of cloud computing services through cross roads of human-immune and intrusion-detection systems. In *2018 International Conference on Information Networking (ICOIN)*, IEEE, 174–181 (2018).
4. Chiba, Z., Abghour, N., Moussaid, K. & Rida, M. Intelligent approach to build a deep neural network based IDS for cloud environment using combination of machine learning algorithms. *Comput. Secur.* **86**, 291–317 (2019).
5. Gill, S. S. & Buyya, R. SECURE: Self-protection approach in cloud resource management. *IEEE Cloud Comput.* **5**(1), 60–72 (2018).
6. Shen, S. et al. Multistage signaling game-based optimal detection strategies for suppressing malware diffusion in fog-cloud-based IoT networks. *IEEE Internet Things J.* **5**(2), 1043–1054 (2018).
7. Ali, M. H., Al Mohammed, B. A. D., Ismail, A. & Zolkipli, M. F. A new intrusion detection system based on fast learning network and particle swarm optimization. *IEEE Access* **6**, 20255–20261 (2018).
8. Raviprasad, B. et al. Accuracy determination using deep learning technique in cloud-based IoT sensor environment. *Meas. Sens.* **24**, 100459 (2022).
9. Lata, S. & Singh, D. Intrusion detection system in cloud environment: Literature survey & future research directions. *Int. J. Inf. Manag. Data Insights* **2**(2), 100134 (2022).
10. Peng, K., Leung, V. C. M. & Huang, Q. Clustering approach based on mini batch Kmeans for intrusion detection system over big data. *IEEE Access* **6**, 11897–11906 (2018).
11. Tan, Z. et al. Enhancing big data security with collaborative intrusion detection. *IEEE Cloud Comput.* **1**(3), 27–33 (2014).
12. Geetha, T. V. & Deepa, A. J. A FKPCA-GWO WDBiLSTM classifier for intrusion detection system in cloud environments. *Knowl.-Based Syst.* **253**, 109557 (2022).
13. Bharati, M., & Tamane, S. Intrusion detection systems (IDS) and future challenges in cloud based environment, In *Proceedings of 1st International Conference on Intelligent Systems and Information Management (ICISIM)*, Aurangabad, 2017, 240–250 (2017).
14. Liu, J., Yu, J. & Shen, S. Energy-efficient two-layer cooperative defense scheme to secure sensor-clouds. *IEEE Trans. Inf. Forensics Secur.* **13**(2), 408–420 (2018).
15. Samriya, J. K. et al. Network intrusion detection using ACO-DNN model with DVFS based energy optimization in cloud framework. *Sustain. Comput.: Inform. Syst.* **35**, 100746 (2022).
16. Roy, S., Li, J. & Bai, Y. A two-layer fog-cloud intrusion detection model for IoT networks. Internet of Things, 100557 (2022).
17. Almiani, M., Abughazleh, A., Jararweh, Y. & Razaque, A. Resilient back propagation neural network security model for containerized cloud computing. *Simul. Model. Pract. Theory* **118**, 102544 (2022).
18. Soni, D. & Kumar, N. Machine learning techniques in emerging cloud computing integrated paradigms: A survey and taxonomy. *J. Netw. Comput. Appl.* **205**, 103419 (2022).
19. Moloja, D., & Mpekoa, N. Securing M-voting using cloud intrusion detection and prevention system: A new dawn, In *Proceedings of IST-Africa Week Conference (IST-Africa)*, 2017, Windhoek, 1–8. (2017).
20. Iraqi, O. & El Bakkali, H. Communizer: A collaborative cloud-based self-protecting software communities framework-focus on the alert coordination system. *Comput. Secur.* **117**, 102692 (2022).
21. Gill, S. S. & Buyya, R. SECURE: Self-protection approach in cloud resource management. *IEEE Cloud Comput.* **5**(1), 60–72 (2018).
22. Yazdinejad, A., Parizi, R. M., Dehghantanha, A., Zhang, Q. & Choo, K. K. R. An energy-efficient SDN controller architecture for IoT networks with blockchain-based security. *IEEE Trans. Serv. Comput.* **13**(4), 625–638 (2020).
23. Vieira, K. et al. Intrusion detection for grid and cloud computing. *IT Prof.* **12**(4), 38–43 (2010).
24. Yazdinejad, A., Dehghantanha, A., Karimipour, H., Srivastava, G. & Parizi, R. M. A robust privacy-preserving federated learning model against model poisoning attacks. *IEEE Trans. Inf. Forensics Secur.* **19**, 6693–6708 (2024).
25. Lorenz, C. et al. An SDN/NFV-enabled enterprise network architecture offering fine-grained security policy enforcement. *IEEE Commun. Mag.* **55**(3), 217–223 (2017).
26. Yazdinejad, A. et al. Block hunter: Federated learning for cyber threat hunting in blockchain-based iIot networks. *IEEE Trans. Ind. Inf.* **18**(11), 8356–8366 (2022).
27. Varadharajan, V. & Tupakula, U. On the design and implementation of an integrated security architecture for cloud with improved resilience. *IEEE Trans. Cloud Comput.* **5**(3), 375–389 (2017).

28. Namakshenas, D., Yazdinejad, A., Dehghantanha, A. & Srivastava, G. Federated quantum-based privacy-preserving threat detection model for consumer internet of things. *IEEE Trans. Consum. Electron.* **2024**, 1–11 (2024).
29. Colom, J. F. et al. Scheduling framework for distributed intrusion detection systems over heterogeneous network architectures. *J. Netw. Comput. Appl.* **108**, 76–86 (2018).
30. Sultana, N. et al. Survey on SDN based network intrusion detection system using machine learning approaches?. *Peer-To-Peer Netw. Appl.* **2018**, 1–9 (2018).
31. Jin, H. et al. A VMM-based intrusion prevention system in cloud computing environment. *J. Supercomput.* **66**(3), 1133–1151 (2013).
32. Kanna, P. R. & Santhi, P. Hybrid intrusion detection using map reduce based black widow optimized convolutional long short-term memory neural networks. *Expert Syst. Appl.* **194**, 116545 (2022).
33. Nasir, M. H., Khan, S. A., Khan, M. M. & Fatima, M. Swarm intelligence inspired intrusion detection systems? A systematic literature review. *Comput. Netw.* **205**, 108708 (2022).
34. University of California, Irvine, 1999. KDD CUP 99 Data. Available at: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html [Accessed 21 Aug. 2024].
35. Creech, G., & Hu, J. ADFA intrusion detection datasets. University of New South Wales (UNSW). Available at: https://research.unsw.edu.au/projects/adfa-ids-datasets. (2013).
36. Smith, A., Brown, B. & Davis, C. A deep learning approach for cloud-based IDS with an emphasis on detection accuracy. *IEEE Trans. Cloud Comput.* **9**(4), 512–523 (2021).
37. Johnson, M. & Lee, S. Anomaly detection using hybrid machine learning techniques in cloud environments. *IEEE Access* **8**, 34567–34576 (2020).
38. Gupta, R. & Kumar, A. A survey on cloud IDS highlighting various methodologies and their performance metrics. *Int. J. Netw. Secur.* **21**(2), 240–250 (2019).

Acknowledgements

The authors would like to acknowledge the support of Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2024R435), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

Author contributions

Conceptualization: R.D., P.C., M.R., S.k., A.Y., N.S.Y., D.S.A.E., D.M.A.; Methodology: R.D.; Formal analysis & data curation: R.D.; Writing—original draft preparation: R.D.; writing—review & editing: P.C.; supervision: M.R., S.k., A.Y., N.S.Y., D.S.A.E., D.M.A.; Funding: D.M.A.; All authors have read and agreed to the published version of the manuscript.

Funding

The authors would like to acknowledge the support of Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2024R435), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

Declarations

Competing interests

The authors declare no conflict of interest.

Consent to participate

Not Applicable.

Consent to publish

Not Applicable.

Additional information

Correspondence and requests for materials should be addressed to D.M.A.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.