Journal of Intelligent Systems and Internet of Things
Vol. 18, No. 01, PP. 48-63, 2026
DOI: https://doi.org/10.54216/JISIoT.180104

Blade Server Attack Detection and Mitigation Framework in Cloud Computing Using SSGRU and GGSSO

Waleed Kh. Hussein[1,*], Ghaith J. Mohammed[1], Ahmed Salih Al-Obaidi[1], Massila Kamalrudin[2], Mustafa Musa[3]

[1]Biomedical Informatics College, University of information Technology and Communication, Iraq

[2]Faculty of Information and Communication Technology, Universiti Teknikal Malaysia Melaka, Malaysia

[3]Center of Research and Innovation Management, Universiti Teknikal Malaysia Melaka, Malaysia

Emails: dr.waleed.khalid@uoitc.edu.iq; dr.ghaith.jaafar@uoitc.edu.iq; dr.ahmed-obaidi@uoitc.edu.iq; massila@utem.edu.my; mustafmusa@utem.edu.my

Abstract

Malicious activities that seek to disrupt cloud communication are cybersecurity threats. Nevertheless, none of the existing works focused on detecting the attacks that happened in the Blade Server (BS) in the cloud. Therefore, this paper proposes an efficient Intrusion Detection System (IDS) framework for BS in the cloud by utilizing Kerberos-based Exponential Mestre-Brainstrass Curve Cryptography (KEMBCC) and Sechsoftwave and Sparsele-centric Gated Recurrent Unit (SSGRU). Primarily, the cloud users are registered into the network. Then, the incoming data are encrypted. Here, to balance the incoming loads, BS is used. To detect attacks in BS, IDS is implemented. Initially, the data are preprocessed. Further, the big data are handled in the IDS. Afterward, the features are extracted and optimal features are chosen from it. Thereafter, to classify the attack and normal BS, the SSGRU classifier is used. After that, by generating a Sankey diagram, the attacked and non-attacked blades in the BS are differentiated. Next, the attacked blades are isolated, whereas the non-attacked blades are further used for load balancing on the cloud. According to the analysis results, this model performed superior to the other models by attaining an accuracy of 99.43%.

Received: March 11, 2025 Revised: June 05, 2025 Accepted: July 12, 2025

Keywords: Cloud Computing; Blade server; Cybersecurity threat detection; Set-Union Combiner-based Hadoop map-reduce; Grid-Greedy initialization-based Shark Smell Optimization

1. Introduction

The internet has played a vital role in each aspect of people's lives in recent times [1]. Organizations are being driven to adopt Cloud Computing (CC) solutions with the rapid growth of network infrastructure [2,3]. The CC enables efficient utilization and on-demand access to cloud resources for humans via the Internet [4]. However, the cybersecurity threats are widespread and become more intense with the continuous advancement of the internet application [5]. To disrupt the data communication within the cloud, the attackers intrude into the network operations [6]. This may lead to financial losses, privacy issues, and data losses for the victims [7]. Hence, to mitigate these problems, an efficient IDS is implemented. The attacker's sign is dynamically detected by the IDS, which also alerts the authorities [8].

Gated Recurrent Unit (GRU), Deep Belief Network (DBN), Deep Neural Network (DNN), Long Short-Term Memory (LSTM), and Recurrent Neural Network (RNN) were some common methods utilized in IDS [9,10]. But, none of these traditional models detected the attacks happening on the BS in the cloud. Therefore, in this paper, a robust IDS for BS utilizing KEMBCC and SSGRU is proposed. The main issues can summarized as follows:

- None of the prevailing works focused on detecting and diminishing attacks occurring on BS.
- [11] was an ineffective model owing to not efficiently handling the sheer scale of cloud environments.
- [12] lacked the potential to analyze how attacks spread across the cloud components.
- In [13], the integrity of the shared cloud traffic was lower.
- The False Positive Rate (FPR) of the prevailing models was higher.

The main objectives of this study are:

- IDS is carried out for BS on the cloud.
- By utilizing Set-Union Combiner-based Hadoop map-reduce (SUCH), the big data of the cloud are handled.
- To determine the attack spreading pattern in the cloud, Lorentchy membership-based Fuzzy Logic System (LFLS)-centric Sankey diagram is generated.
- To encrypt the incoming cloud data, KEMBCC is implemented.
- The FPR is mitigated by optimally selecting the features utilizing Grid-Greedy initialization-centric Shark Smell Optimization (GGSSO).

The paper is structured as: The literature survey is presented in Section 2, the proposed method is explained in Section 3, the results are discussed in Section 4, and lastly, the paper is winded up in Section 5 with future enhancements.

2. Literature Survey

According to [11] established an advanced persistent threat detection model in CC by utilizing an autoencoder and softmax regression algorithm. According to the analysis results, this model attained superior detection accuracy. Nevertheless, this model was ineffective in handling the sheer scale of the cloud. [12] illustrated an efficient IDS for securing data on CC by utilizing a support vector machine and genetic algorithm. As per the assessment outcomes, this model performed superior to the other top-notch models by providing a high level of attack symmetries. But, the attack spreading pattern was not investigated by this model. [13] presented an intelligent behavior-centric malware detection system on CC. The experimental analysis outcomes proved the superiority of this model by obtaining maximum detection rates. However, the integrity of this model was lower. [14] propounded a radial basis function neural network and random forest-centric intelligent IDS on CC. According to the analysis results, this model strengthened the overall security mechanisms of CC. Nevertheless, the false positives of this model were higher. [15] implemented Facebook prophet and collaborative feature selection model-based IDS on CC. The assessment outcomes proved the efficacy of this model in CC security. However, owing to the utilization of fewer features for attack detection, this model had overfitting problems.

3. Proposed IDS Methodology for Bs in the Cloud using KEMBCC and SSGRU

This framework detects the cybersecurity attack occurring within the BS by utilizing KEMBCC and SSGRU. Figure 1 presents the architecture of the proposed framework.

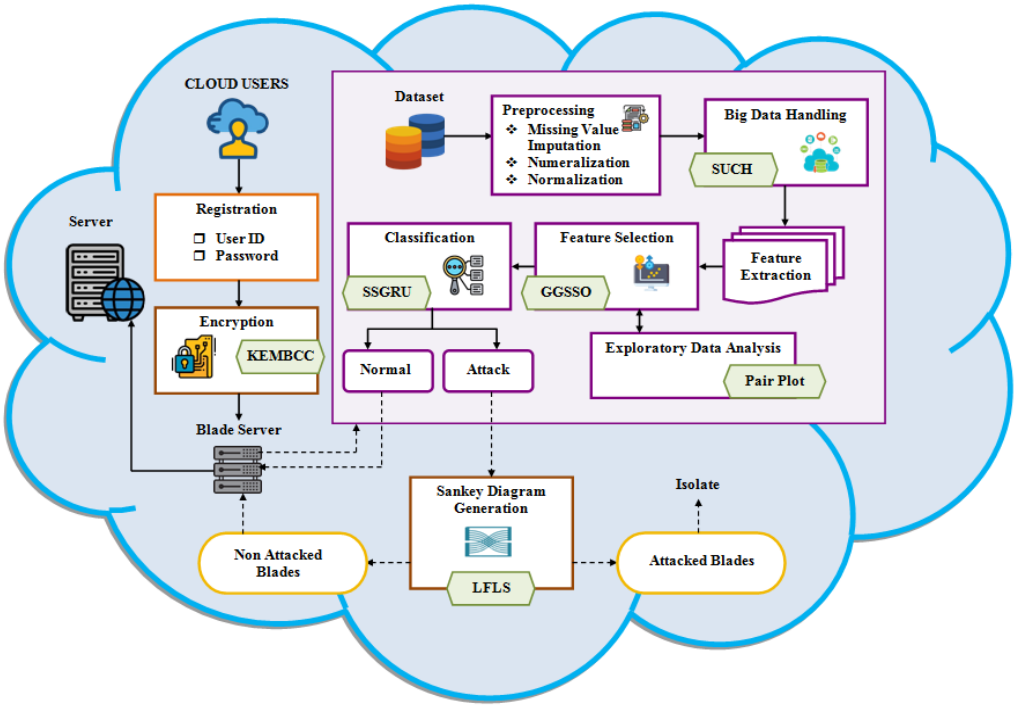

Figure 1. Architecture of the proposed framework

3.1 Registration

The cloud users (C) are registered into the network with their user ID (C_1) and password (C_2). Here, the c number of C are registered and are represented as,

$$C = \{C_1, C_2, ..., C_c\} \tag{1}$$

Afterward, the user can access the cloud resources.

3.2 Encryption

Thereafter, by utilizing KEMBCC, the data $(\hbar)$ of registered C are secured. Elliptic Curve Cryptography (ECC) provides better security with shorter key lengths. Nevertheless, ECC had lower efficiency owing to the elliptic curve parameters. Hence, the Exponential Mestre-Brainstrass curve is used. Moreover, by utilizing the Kerberos function, a secret key is generated. In this, centered on the Exponential Mestre-Brainstrass curve, the keys are generated and are demonstrated as,

$$y^2 = x^3 + \nu_1 x + \nu_2 (\mathrm{mod}\, \zeta) \tag{2}$$

Where, the coordinates of the curve are implied as x *and* y, the constant parameters are notated as ν_1 *and* ν_2, and the prime number is signified as ζ. After that, a base point (β) is chosen on the curve. Then, the public key (λ) is created grounded on the subsequent expression,

$$\lambda = \Im \cdot \beta \tag{3}$$

Here, the random point on the curve (i.e., private key) is exemplified as $\Im$. Together with public and private keys, a secret key (s) is also generated by utilizing the Kerberos function, and it is expressed as,

$$s = \aleph^{KDF}\left(\iota + C_2\right) \tag{4}$$

Where, the key derivative function is notated as $\aleph^{KDF}$, and ι denotes a random value. Subsequently, the input $\hbar$ is encrypted centered on the generated keys and is given as,

$$\gamma_1 = rand * s \tag{5}$$

$$\gamma_2 = \hbar + rand \cdot \lambda \tag{6}$$

Here, ciphertext 1 and ciphertext 2 are exemplified as $\gamma_1 \; and \; \gamma_2$, correspondingly, and the random number is implied as $rand$. Afterward, the decryption process is carried out.

$$\hbar = \gamma_2 - \Im * \gamma_1 - s \tag{7}$$

Therefore, the data $(\hbar)$ is securely encrypted and is signified as $\hbar^*$.

3.3 Blade Server

When the incoming $\hbar^*$ is higher, the BS is used by the cloud for load balancing. BS is designed to handle high-density workloads. It also maximizes the utilization of data center resources. However, the BS is vulnerable to cybersecurity threats. Hence, for BS, a robust IDS is developed.

3.3.1 Preprocessing

Initially, the missing values in the rows and columns of the dataset are imputed. Thereafter, the data are converted into numbers. Next, the numeralized data are normalized. Thus, ϑ denotes the preprocessed data.

3.3.2 Big Data Handling

Afterward, a SUCH-centric big data handling process is undergone by the preprocessed ϑ. Hadoop Map-Reduce (HMR) is highly flexible and scalable for processing vast amounts of data. However, the complexity of data shuffling from mapper to reducer was higher. Therefore, a set union combiner is implemented. Primarily, the input ϑ is divided into smaller chunks (ϑ_{ch}) and is articulated as,

$$\vartheta \xrightarrow{divide} \vartheta_{ch} \tag{8}$$

Subsequently, the mapper reads ϑ_{ch} and generates an intermediate key-value pair, and it is displayed as,

$$map\left(\vartheta_{ch}\right): \left(\xi_1, \Theta_1\right) \rightarrow list\left(\xi_2, \Theta_2\right), \quad where, \left\{\left(\Theta_1, \Theta_2\right) \in \vartheta_{ch}\right\} \tag{9}$$

Where, the mapper output is implied as $map(\vartheta_{ch})$, the input and intermediate identifier keys are notated as $\xi_1 \; and \; \xi_2$, correspondingly, and the input and intermediate values are exemplified as $\Theta_1 \; and \; \Theta_2$, respectively. Then, the set union combiner (ξ_{com}) function is utilized for the keys on $map(\vartheta_{ch})$ before shuffling it to the reducer phase. Hence, the ξ_{com} is demonstrated as,

$$\xi_{com} = \bigcup_{i=1}^{j} \xi_i \tag{10}$$

Here, the individual sets produced by the mapper for the keys are notated as ξ_i, and the number of sets for the keys produced from $map(\vartheta_{ch})$ is implied as j. Next, the output of ξ_{com} is shuffled and sorted by SUCH. Every single group of ξ_{com} is diminished in the reduction phase, thereby producing a set of final key-value pairs (ξ_3), and it is expressed as,

$$\vartheta_{big} : reduce(\xi_{com}) \rightarrow list(\xi_3) \tag{11}$$

Therefore, ϑ_{big} indicates the handled big data.

3.3.3 Feature Extraction

The features, namely protocol type, service, flag, duration, traffic volume, outbound traffic, resource utilization, and failed login attempts are extracted from ϑ_{big} and are signified as ϑ_{ext}.

3.3.4 Feature Selection

Thereafter, by utilizing GGSSO, the optimal features are chosen from ϑ_{ext}. The search space is broadly explored by the Shark Smell Optimization (SSO) algorithm. However, the initial position of the shark is randomly selected. Hence, to generate the initial position of the shark, the Grid-Greedy initialization technique is used. The population of the shark (S) is represented as,

$$S = \{S_1, S_2, ..., S_h\} \tag{12}$$

Where, the number of sharks in the population is notated as h. The objective function (χ) of GGSSO is attaining maximum $(\max)$ classification accuracy (υ) and is displayed as,

$$\chi = \max\{\upsilon_S\} \tag{13}$$

Then, the selected χ is generated as a visual representation, which is assessed under the Exploratory Data Analysis (EDA) process by utilizing a pair plot for ensuring the strength of χ. After that, the initial position (S_a) of the shark is expressed as,

$$S_a = \frac{A + i'(B - A)}{T} + \breve{\Psi} \cdot (\chi + \hat{\wp} \cdot \nabla \chi) \tag{14}$$

Here, the tuning intervals are exemplified as $A and B$, the number of grid points (T) is implied as i', the weight parameter and learning rate are notated as $\breve{\Psi} and \hat{\wp}$, correspondingly, and the gradient of χ is denoted as $\nabla\chi$. The initial velocity vector (Z) of the shark is determined as,

$$Z = \{Z_1, Z_2, ..., Z_h\} \tag{15}$$

The sharks had inertia when they swam. Therefore, the velocity of every single dimension (E) depends on the previous velocity and is given as,

$$Z_{a,b}^{w} = \varpi_w \cdot \tau_1 \cdot \frac{\partial(\chi)}{S_a}\Big|_{S_{a,b}^{w}} + \alpha_w \cdot \tau_2 \cdot Z_{a,b}^{w-1} \tag{16}$$

Here, the velocity of the a^{th} shark in the b^{th} dimension of E at iteration (w) is implied $Z_{a,b}^{w}$, the gradient coefficient is exemplified as ϖ_w, the weight coefficient is notated as α_w, and the random numbers between $[0,1]$ are signified as $\tau_1 and \tau_2$. Next, the speed of the shark is limited, and it is displayed as follows,

$$\left|Z_{a,b}^{w}\right| = \min\left\{Z_{a,b}^{w}, \left|\tilde{\varphi}_w, Z_{a,b}^{w-1}\right|\right\} \tag{17}$$

Where, the speed limit factor in w^{th} iteration is notated as $\tilde{\varphi}_w$. Afterward, by utilizing the previous speed and position, the position of the shark is updated. The new position $\left(S_{a,b}^{w+1}\right)$ is determined as,

$$S_a^{w+1} = S_a^{w} + Z_a^{w} \cdot \Delta^{w} \tag{18}$$

Here, the time interval at w^{th} iteration is implied as Δ^w. The local search position $\left(L_{a,b}^{w+1,k}\right)$ of the shark is updated centered on the rotation motion and is articulated as,

$$L_{a,b}^{w+1,k} = S_a^{w+1} + \tau_3 \cdot S_a^{w+1} \tag{19}$$

Where, the random number is notated as τ_3, and the number of points in the local search is exemplified as k. The shark moves towards the location of that prey if it finds a stronger odor when rotating. Hence, the search behavior is explained as,

$$S_a^{w+1} = \arg\max\left\{\chi\left(S_a^{w+1}\right), \chi\left(L_a^{w+1,1}\right), ..., \chi\left(L_a^{w+1,k}\right)\right\} \tag{20}$$

Therefore, centered on the highest χ values, the shark selects the candidate solution. The cycle of forward and rotation movement continues till w reaches maximum iteration $(w_{\max})$. The optimal features (ϑ_{opt}) are chosen at the end of the $w_{\max}$. The pseudocode for GGSSO is demonstrated below,

Input: Extracted features $\left(\vartheta_{ext}\right)$

Output: Optimal features $\left(\vartheta_{opt}\right)$

Begin

Initialize population of the shark $\left(S\right)$, maximum iteration $\left(w_{\max}\right)$, χ, and φ_w

For each ϑ_{ext} **do**

Compute $\chi = \max\left\{\upsilon_S\right\}$

Evaluate initial position

Initialize velocity vector

Formulate $Z_{a,b}^{w} = \varpi_w \cdot \tau_1 \cdot \left.\frac{\partial(\chi)}{S_a}\right|_{S_{a,b}^{w}} + \alpha_w \cdot \tau_2 \cdot Z_{a,b}^{w-1}$

Limit the speed of the shark

Update position $S_a^{w+1} = S_a^{w} + Z_a^{w} \cdot \Delta^{w}$

Evaluate local search position

If $\left(\Omega_{new} > \Omega\right)${

Change the prey location

} Else {

Moving forward

}

End If

Estimate $S_a^{w+1} = \arg\max\left\{\chi\left(S_a^{w+1}\right), \chi\left(L_a^{w+1,1}\right), ..., \chi\left(L_a^{w+1,k}\right)\right\}$

If $\left(w = w_{\max}\right)${

Stop

} Else {

Continue

}

End If

End For

Return optimal features

End

The attack in BS is further detected grounded on ϑ_{opt}.

3.3.5 Classifications

In this subsection, to detect the attack, the ϑ_{opt} is given to SSGRU. Gating mechanisms are used by GRU for controlling the information flow. Nevertheless, GRU had a low learning efficiency and slow convergence rate. Thus, the Sechsoftwave activation function and Sparsele initialization are implemented. Figure 2 displays the structure of

SSGRU.

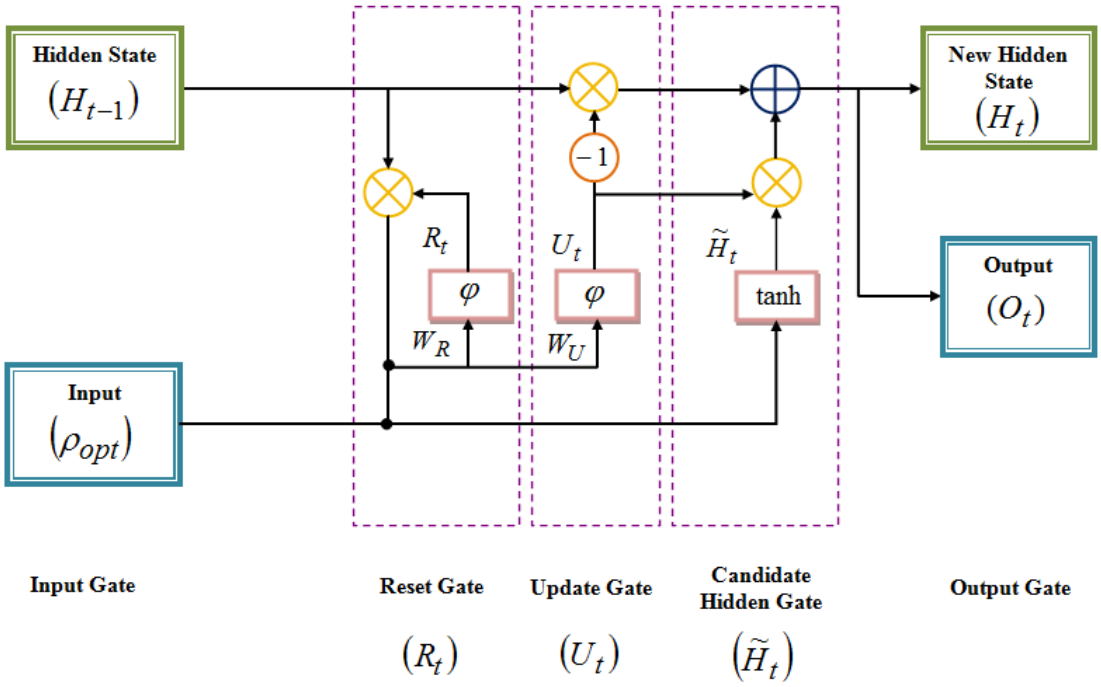

Figure 2. Structure of proposed SSGRU

- The input gate holds the input (ϑ_{opt}). Thereafter, ϑ_{opt} is transferred to the reset gate (R) and is displayed as,

$$\vartheta_{opt} \xrightarrow{transfer} R_t \tag{21}$$

Where, the current state is implied as t.

- Then, R decide how much of the previous information needs to be forgotten from the gates in t.

$$R_t = \varphi \cdot (W_R * H_{t-1}, \vartheta_{opt}) \tag{22}$$

Here, the previous hidden state (H_t) is exemplified as H_{t-1}, and the weights of R are denoted as W_R, which is determined below utilizing the Sparsele initialization function as,

$$W_R = N\left(0, \frac{1}{n(\vartheta_{opt})}\right) \tag{23}$$

Where, the number of neurons in ϑ_{opt} is notated as $n(\vartheta_{opt})$, and the normalization function is implied as N. Moreover, φ indicates the Sechsoftwave activation function, which is demonstrated as,

$$\varphi = I \cdot \frac{\vartheta_{opt}}{1+\left|\vartheta_{opt}\right|} + J \cdot \sin\left(\vartheta_{opt}\right) + \breve{\varepsilon} \cdot \frac{2}{e^{\vartheta_{opt}} + e^{-\vartheta_{opt}}} \tag{24}$$

Here, the controlling parameters are exemplified as $I, J, and\ \breve{\varepsilon}$.

- Subsequently, the update gate $\left(U\right)$ determines how much the new input is utilized for updating the hidden gate at t.

$$U_t = \left(W_U * H_{t-1}, R_t\right) \tag{25}$$

Where, W_U signifies the weights of U, which is initialized by the Sparsele initialization function.

- Afterward, the candidate's hidden state $\left(\vec{H}_t\right)$ is signified as,

$$\vec{H}_t = \varphi \cdot \left|\vartheta_{opt} * U_t + \left|R_t \circ H_{t-1} * W_U\right|\right| \tag{26}$$

- Moreover, the hidden state $\left(H_t\right)$ is determined as,

$$H_t = U_t * H_{t-1} + \left(1 - U_t\right) * \vec{H}_t \tag{27}$$

- Thereafter, the loss function $\left(\Diamond\right)$ is assessed. Lastly, the output gate $\left(O_t\right)$ is obtained with the attack $\left(\Gamma_1\right)$ and normal $\left(\Gamma_2\right)$ classes.

$$O_t \rightarrow \left|\Gamma_1, \Gamma_2\right| \tag{28}$$

The pseudocode for SSGRU is presented as follows,

Input: Optimal features $\left(\vartheta_{opt}\right)$

Output: Classified output $\left(O_t\right)$

Begin

Initialize $R_t, U_t\ H_t$, W_R and W_U

For each ϑ_{opt} **do**

Compute $R_t = \left(W_R * H_{t-1}, \vartheta_{opt}\right)$

Initialize weights

Activate $\varphi = I \cdot \frac{\vartheta_{opt}}{1+\left|\vartheta_{opt}\right|} + J \cdot \sin\left(\vartheta_{opt}\right) + \breve{\varepsilon} \cdot \frac{2}{e^{\vartheta_{opt}} + e^{-\vartheta_{opt}}}$

Evaluate update gate

Perform $\vec{H}_t = \varphi \cdot \left|\vartheta_{opt} * U_t + \left|R_t \circ H_{t-1} * W_U\right|\right|$

Formulate new hidden state

If $\left(\Diamond = satisfied\right)$**{**

Terminate

} Else {

Tune the parameters

}

End If

End For

Return $O_t \rightarrow \left|\Gamma_1, \Gamma_2\right|$

End

If Γ_2 is obtained, then the loads are efficiently balanced by the BS and further processed in the server. If Γ_1 is obtained, then the BS is further processed as explained below.

3.4 Sankey Diagram Generation

If Γ_1 is detected from SSGRU, then a Sankey diagram is generated centered on LFLS for recognizing the attacked blades in the BS. A Fuzzy Logic System (FLS) can efficiently handle the uncertainties. Nevertheless, designing fuzzy rules can become complicated. In addition, it requires domain expertise and extensive trials. Thus, for diminishing the complexity, a well-defined Lorentchy membership function is used. Primarily, the conditions (ϕ) for generating the Sankey diagram (δ) are set and are explained as follows,

$$\delta = \begin{cases} \delta_1, & if\ \phi \\ \delta_2, & otherwise \end{cases} \tag{29}$$

$$\phi \rightarrow \left\{\phi_1 = 200 - 300\%, \phi_2 > 50, \phi_3 = 100 - 200\%, \phi_4 = 90 - 100\%\right\} \tag{30}$$

Where, the attacked blades and non-attacked blades are notated as $\delta_1\ and\ \delta_2$, correspondingly, and the traffic volume, failed login attempt, outbound traffic, and resource utilization in the BS are exemplified as $\phi_1, \phi_2, \phi_3, \phi_4$, respectively. To map the input $(\aleph)$ into a fuzzy value $(\hat{\aleph})$, the fuzzification (ς) block of LFLS is utilized and is articulated as,

$$\delta \xrightarrow{\varsigma} \tilde{\delta} \tag{31}$$

Next, the inference engine is responsible for making decisions by utilizing the Lorentchy membership function (ψ), which is signified as,

$$\psi = \tilde{\delta} \cdot \frac{1}{1+\left(\frac{\tilde{\delta}-\tilde{\omega}}{\Xi}\right)^2} + \left(1-\tilde{\delta}\right)\frac{1}{\pi \cdot \Xi\left[1+\left(\frac{\tilde{\delta}-\tilde{\omega}}{\Xi}\right)^2\right]} \tag{32}$$

Here, the scale and center parameters are implied as $\Xi \, and \, \tilde{\omega}$, correspondingly. The last defuzzification $(\tilde{\varsigma})$ functional block converts the fuzzy quantity into crisp values, and it is articulated as,

$$\psi\left(\tilde{\delta}\right) \xrightarrow{\tilde{\varsigma}} \delta \tag{33}$$

Therefore, the output obtained is signified as,

$$\delta = \left\{\delta_1, \delta_2\right\} \tag{34}$$

The blade is isolated from the BS if δ_1 is obtained. Or else, the blades are further proceeded to run on its application in the cloud server. The performance of this framework is discussed in a further section.

4. Results And Discussions

Here, to prove the robustness of the proposed IDS for BS, the proposed framework is implemented in the working platform of PYTHON.

4.1 Dataset Description

In the proposed system, the Network Security Laboratory-Knowledge Discovery and Data mining (NSL-KDD) anomaly detection dataset is used. This dataset is mentioned in the references. From the whole data, 80% and 20% of data are used for training and testing purposes, correspondingly.

4.2 Performance Analysis

In this subsection, the proposed model's performance is analogized with prevailing techniques for proving its reliability.

The comparative investigation of the proposed SSGRU and the prevailing models is depicted in Figure 3. The Sparsele initialization improves the learning efficiency of the proposed model. Hence, the proposed SSGRU obtained an accuracy of 99.43%, precision of 99.25%, recall of 99.08%, F-measure of 99.16%, sensitivity of 99.08%, specificity of 99.25%, False Negative Rate (FNR) of 0.023, and FPR of 0.029. Nevertheless, the existing models achieved low learning efficiency. Thus, when contrasted with the proposed method, the performances of the prevailing models were lower. Therefore, the proposed SSGRU was proved as less error-prone.

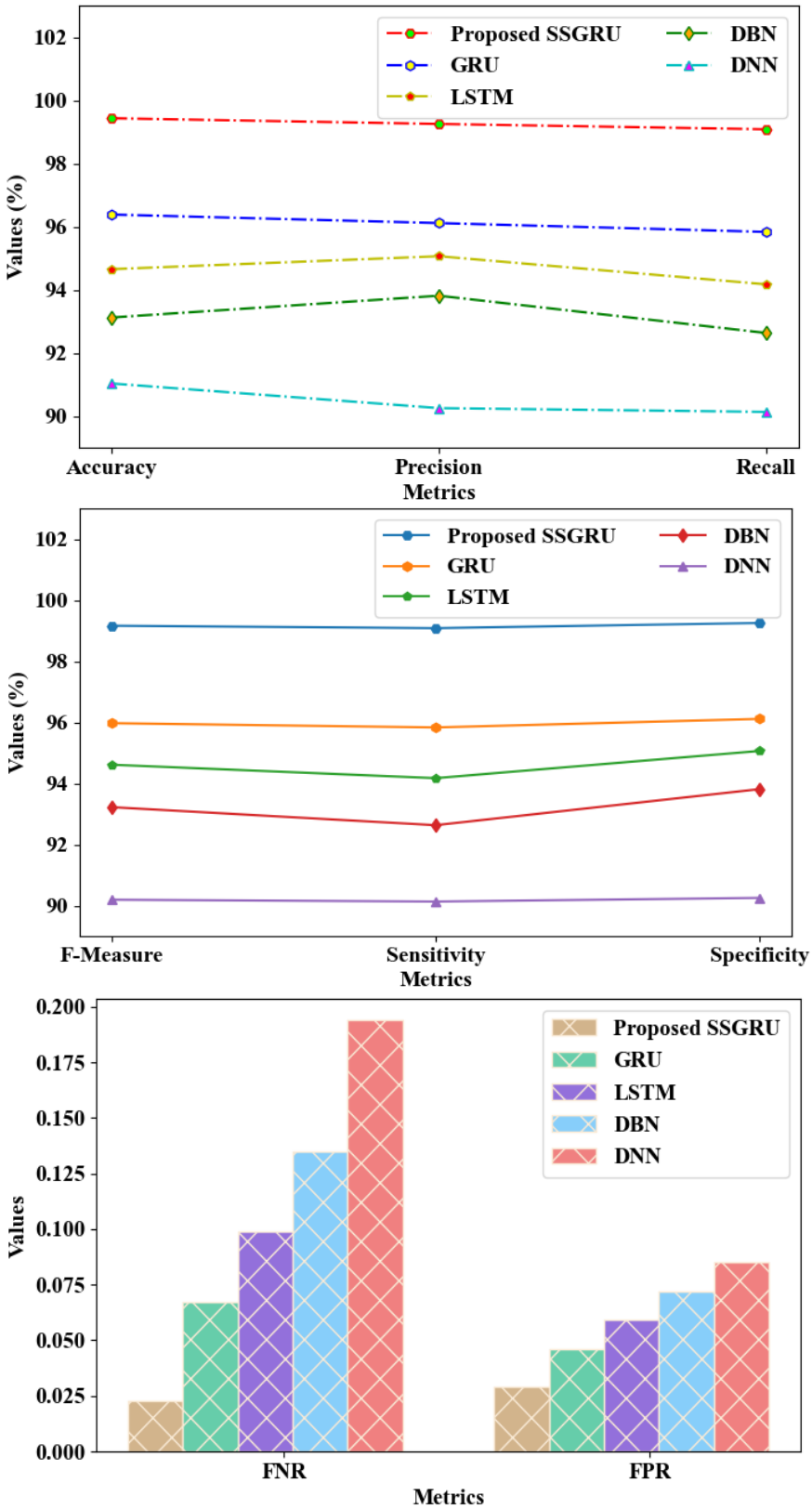

Figure 3. Comparative analysis of proposed SSGRU

Table 1: Performance analysis of proposed KEMBCC

Methods	Encryption time (ms)	Decryption time (ms)	Security level (%)
KEMBCC	1349	1385	98.59
ECC	2134	2166	95.26
Rivest-Shamir-Adleman	3294	3305	92.95
Data Encryption Standard	3985	4109	89.71
ElGamal	5421	5660	88.2

In Table 1, the performance analysis of the proposed and existing models is displayed. The prevailing ECC achieved an encryption time of 2134ms, a decryption time of 2166ms, and a security level of 95.26%. Nevertheless, the proposed model for improving security generates a secret key. Therefore, the proposed model attained encryption time, decryption time, and security level of 1349ms, 1385ms, and 98.59%, correspondingly.

Table 2: Average fitness and feature selection time analysis

Techniques	Average fitness (%)	Feature selection time (ms)
Proposed GGSSO	98.41	1292
SSO	96.39	1954
ACO	94.42	2753
RPO	91.31	4629
ABCO	89.52	5758

The performance analysis of the proposed and the prevailing SSO, Ant Colony Optimization (ACO), Red Panda Optimization (RPO), and Artificial Bee Colony Optimization (ABCO) models is depicted in Table 2. The proposed GGSSO obtained an average fitness and feature selection time of 98.41% and 1292ms, correspondingly. Nevertheless, a low average fitness of 89.52% and a high feature selection time of 5758ms were attained by the existing ABCO. Therefore, the proposed model efficiently selected the optimal features.

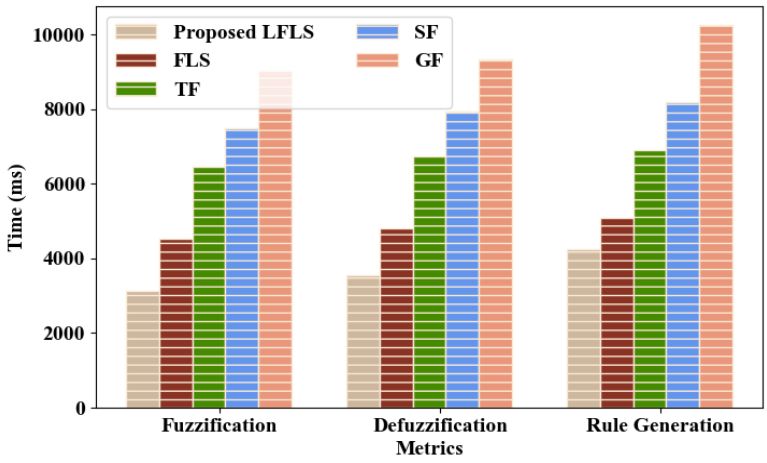

Figure 4. Performance analysis of proposed LFLS

The performance analysis of the proposed LFLS and the existing FLS, Triangular Fuzzy (TF), Sigmoid Fuzzy (SF), and Gaussian Fuzzy (GF) is shown in Figure 4. For fuzzification, defuzzification, and rule generation, the proposed LFLS took a minimum time of 3142ms, 3540ms, and 4247ms, respectively. Nevertheless, the prevailing models took an average time of 6883.5ms for fuzzification, 7211.75ms for defuzzification, and 7617.5ms for rule generation. This dynamic performance of the proposed LFLS was owing to the utilization of a well-defined membership function.

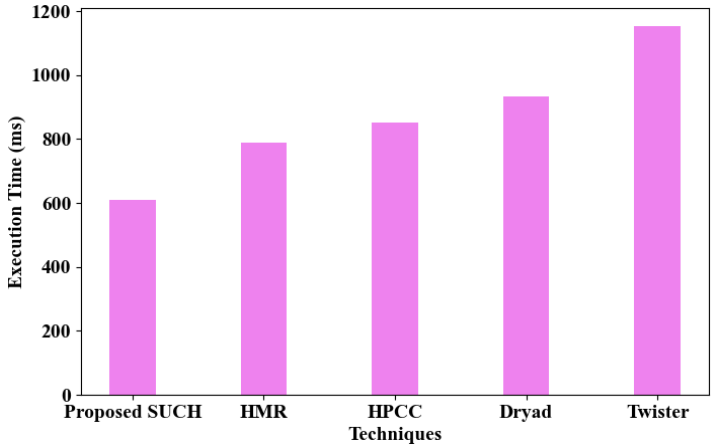

Figure 5. Execution time analysis

Figure 5 illustrates the execution time analysis of the proposed SUCH and the prevailing HMR, High-Performance Computing Cluster (HPCC), Dryad, and Twister. The proposed model took a lesser execution time of 610ms, whereas the existing methods took a huge average execution time of 931ms. Here, the shuffling complexity is diminished by the proposed SUCH, thus obtaining robust performance.

4.3 Comparative Analysis with Related Works

Here, for proving the efficiency of the model, the proposed model is analogized with related works.

Table 3: Comparative analysis

Author's name	Methods	Dataset	Accuracy (%)
Proposed	SSGRU	NSL-KDD	99.43
[16]	Boosted tree, bagged tree, subspace discriminant, along with RUSBooted	CICIDS 2017	97.24
[17]	Whale optimization algorithm-based DNN	-	95.35
[18]	Gradient hybrid leader optimization	NSL-KDD	91.7
[19]	Hybrid Convolutional Neural Network (CNN)	NSL-KDD	92.5
[20]	CNN and LSTM	Cyber range lab at the University of New South Wales	98

The comparative investigation of the proposed model with the related works is presented in Table 3. By utilizing ensemble learning, CNN, and LSTM, the existing works developed diverse IDS techniques. Here, the proposed model attained an accuracy of 99.43%, whereas the prevailing works obtained lower accuracy. This was achieved because of diminished low learning efficiency and improved convergence rates by the proposed model. Hence, the efficacy of the proposed system was proved.

5. Conclusion

This paper proposes the IDS framework for BS in the cloud utilizing SSGRU and GGSSO. The user's data were encrypted. Thereafter, IDS for BS was processed by investigating the dataset features, followed by attack classification. The experimental investigation of this work stated the robustness of this model by obtaining a detection accuracy of 99.43%. Moreover, the proposed GGSSO attained an average fitness of 98.41%. The proposed model had a security level of 98.59%. Moreover, for execution, the proposed model took a minimum time of 610ms. According to the analysis results, this model was less error-prone. Therefore, from the overall analysis, the proposed model's robustness was proved.

Future recommendations: The work will be improved in the future by developing the attack mitigation strategies for BS centered on the severity levels of the attacks.

References

[1] M. Ahsan, K. E. Nygard, R. Gomes, M. M. Chowdhury, N. Rifat, and J. F. Connolly, "Cybersecurity Threats and Their Mitigation Approaches Using Machine Learning—A Review," *Journal of Cybersecurity and Privacy*, vol. 2, no. 3, pp. 527–555, 2022. [Online]. Available: https://doi.org/10.3390/jcp2030027.

[2] Ö. Aslan, S. S. Aktuğ, M. Ozkan-Okay, A. A. Yilmaz, and E. Akin, "A Comprehensive Review of Cyber Security Vulnerabilities, Threats, Attacks, and Solutions," *Electronics (Switzerland)*, vol. 12, no. 6, pp. 1–42, 2023. [Online]. Available: https://doi.org/10.3390/electronics12061333.

[3] B. Alouffi, M. Hasnain, A. Alharbi, W. Alosaimi, H. Alyami, and M. Ayaz, "A Systematic Literature Review on Cloud Computing Security: Threats and Mitigation Strategies," *IEEE Access*, vol. 9, pp. 57792–57807, 2021. [Online]. Available: https://doi.org/10.1109/ACCESS.2021.3073203.

[4] N. Ahmed, T. Baker, Z. Jalil, A. Rasool, and W. Ahmad, "Cyber security in IoT-based cloud computing: A comprehensive survey," *Electronics (Switzerland)*, vol. 11, no. 1, pp. 1–34, 2022. [Online]. Available: https://doi.org/10.3390/electronics11010016.

[5] S. El Kafhali, I. El Mir, and M. Hanini, "Security Threats, Defense Mechanisms, Challenges, and Future Directions in Cloud Computing," *Archives of Computational Methods in Engineering*, vol. 29, pp. 223–246, 2022. [Online]. Available: https://doi.org/10.1007/s11831-021-09573-y.

[6] M. Abdullahi, Y. Baashar, H. Alhussian, A. Alwadain, N. Aziz, L. F. Capretz, and S. J. Abdulkadir, "Detecting Cybersecurity Attacks in Internet of Things Using Artificial Intelligence Methods: A Systematic Literature Review," *Electronics (Switzerland)*, vol. 11, no. 2, pp. 1–27, 2022. [Online]. Available: https://doi.org/10.3390/electronics11020198.

[7] V. Chang, L. Golightly, P. Modesti, Q. A. Xu, L. M. T. Doan, K. Hall, S. Boddu, and A. Kobusińska, "A Survey on Intrusion Detection Systems for Fog and Cloud Computing," *Future Internet*, vol. 14, no. 3, pp. 1–27, 2022. [Online]. Available: https://doi.org/10.3390/fi14030089.

[8] M. L. Hernandez-Jaimes, A. Martinez-Cruz, K. A. Ramírez-Gutiérrez, and C. Feregrino-Uribe, "Artificial intelligence for IoMT security: A review of intrusion detection systems, attacks, datasets and Cloud–Fog–Edge architectures," *Internet of Things (Netherlands)*, vol. 23, pp. 1–33, 2023. [Online]. Available: https://doi.org/10.1016/j.iot.2023.100887.

[9] J. Guo and H. Guo, "Real-Time Risk Detection Method and Protection Strategy for Intelligent Ship Network Security Based on Cloud Computing," *Symmetry*, vol. 15, no. 5, pp. 1–13, 2023. [Online]. Available: https://doi.org/10.3390/sym15050988.

[10] A. Alshammari and A. Aldribi, "Apply machine learning techniques to detect malicious network traffic in cloud computing," *Journal of Big Data*, vol. 8, no. 1, pp. 1–24, 2021. [Online]. Available: https://doi.org/10.1186/s40537-021-00475-1.

[11] F. J. Abdullayeva, "Advanced Persistent Threat attack detection method in cloud computing based on autoencoder and softmax regression algorithm," *Array*, vol. 10, pp. 1–11, 2021.

[12] A. Aldallal and F. Alisa, “Effective intrusion detection system to secure data in cloud using machine learning,” *Symmetry*, vol. 13, no. 12, pp. 1–26, 2021. [Online]. Available: https://doi.org/10.3390/sym13122306.

[13] Ö. Aslan, S. S. Aktuğ, M. Ozkan-Okay, A. A. Yilmaz, and E. Akin, “A Comprehensive Review of Cyber Security Vulnerabilities, Threats, Attacks, and Solutions,” *Electronics (Switzerland)*, vol. 12, no. 6, pp. 1–42, 2023. [Online]. Available: https://doi.org/10.3390/electronics12061333.

[14] A. R. Al-Ghuwairi, Y. Sharrab, D. Al-Fraihat, M. AlElaimat, A. Alsarhan, and A. Algarni, “Intrusion detection in cloud computing based on time series anomalies utilizing machine learning,” *Journal of Cloud Computing*, vol. 12, no. 1, pp. 1–17, 2023. [Online]. Available: https://doi.org/10.1186/s13677-023-00491-x.

[15] H. Attou et al., “Towards an Intelligent Intrusion Detection System to Detect Malicious Activities in Cloud Computing,” *Applied Sciences (Switzerland)*, vol. 13, no. 17, pp. 1–19, 2023. [Online]. Available: https://doi.org/10.3390/app13179588.

[16] P. Singh and V. Ranga, “Attack and intrusion detection in cloud computing using an ensemble learning approach,” *International Journal of Information Technology (Singapore)*, vol. 13, no. 2, pp. 565–571, 2021. [Online]. Available: https://doi.org/10.1007/s41870-020-00583-w.

[17] A. Agarwal, M. Khari, and R. Singh, “Detection of DDOS Attack using Deep Learning Model in Cloud Storage Application,” *Wireless Personal Communications*, vol. 127, no. 1, pp. 419–439, 2022. [Online]. Available: https://doi.org/10.1007/s11277-021-08271-z.

[18] S. Balasubramaniam, C. V. Joe, T. A. Sivakumar, A. Prasanth, K. Satheesh Kumar, V. Kavitha, and R. K. Dhanaraj, “Optimization Enabled Deep Learning-Based DDoS Attack Detection in Cloud Computing,” *International Journal of Intelligent Systems*, vol. 2023, pp. 1–16, 2023. [Online]. Available: https://doi.org/10.1155/2023/2039217.

[19] S. S. Hameed, V. Akshaya, V. Mandala, C. Anilkumar, P. Vishnu Raja, and R. Aarthi, “Security enhancement and attack detection using optimized hybrid deep learning and improved encryption algorithm over Internet of Things,” *Measurement: Sensors*, vol. 30, pp. 1–8, 2023. [Online]. Available: https://doi.org/10.1016/j.measen.2023.100917.

[20] T. H. H. Aldhyani and H. Alkahtani, “Artificial Intelligence Algorithm-Based Economic Denial of Sustainability Attack Detection Systems: Cloud Computing Environments,” *Sensors*, vol. 22, no. 13, pp. 1–24, 2022. [Online]. Available: https://doi.org/10.3390/s22134685.