Intrusion Detection and Attack Mitigation for Cloud Blade Servers via Optimized GRU Classification and Kerberos Cryptography

Waleed Khalid Alzubaidi[1,*]
[1]Biomedical Informatics College, University of Information Technology and Communication, Iraq
Email: dr.waleed.khalid@uoitc.edu.iq

Abstract

Cloud communication faces numerous disruptive cybersecurity threats. Various issues related to such disruption have been the subject of previous research, but detection attacks in the blade server (BS) in the cloud have not been studied. Therefore, this paper proposes an efficient intrusion detection system (IDS) framework for BS in the cloud. This framework uses Kerberos authentication-based exponential Mestre-Brainstrass curve cryptography, Sechsoftwave and sparsely centric gated recurrent unit (SSGRU). In this framework, cloud users are firstly registered to the network, and then incoming data are encrypted. The BS is then used to balance the incoming loads, and IDS is applied to detect attacks in the BS, with the data being pre-processed firstly and the big data being handled in the IDS. Afterwards, the features are extracted, from which optimal features are selected. Attacked and normal blades are classified by using the SSGRU classifier and then differentiated by generating a Sankey diagram. The attacked blades are then isolated, and the normal blades are used for load balancing on the cloud. Results indicate that this model achieved 99.43% accuracy, thus demonstrating superior performance to other models.

Keywords: Cloud computing, Blade server; Cybersecurity threat detection; Set-union combiner-based Hadoop MapReduce; Grid-greedy initialization-based shark smell optimization; Lorentzy membership-based fuzzy logic system; Exploratory data analysis

1. Introduction

With the increasingly vital role of the Internet in every aspect of people's lives nowadays [1], network infrastructure is growing, thus prompting organizations to adopt cloud computing (CC) [2,3]. CC provides people with on-demand access to cloud resources and enables them to use such resources efficiently over the Internet [4]. However, such development is accompanied by cybersecurity threats, which are intensifying and becoming widespread [5]. Attackers interfere with network operations to disrupt data communication within the cloud [6]. Such attacks could potentially result in financial losses, privacy issues and data losses for the organizations that are being targeted [7]. Having an efficient intrusion detection system (IDS) in place is necessary to mitigate such problems, allowing the attacker's sign to be dynamically detected and sending out alerts to authorities [8].

The traditional methods used in IDSs include gated recurrent unit (GRU), deep belief network, deep neural network (DNN), long short-term memory (LSTM) and recurrent neural network [9,10]. However, the problem with these methods is that they do not detect attacks to the blade server (BS) in the cloud. Therefore, this paper proposes robust IDS for BS, which utilizes Kerberos authentication-based exponential Mestre-Brainstrass curve cryptography (KEMBCC) and sparsely centric gated recurrent unit (SSGRU).

DOI: https://doi.org/10.54216/FPA.210207
Received: May 08, 2025 Revised: July 27, 2025 Accepted: September 01, 2025

1.1 Problem Statement

Numerous studies have examined cybersecurity issues in the cloud, but none of them aimed to detect and diminish attacks to the BS. In [11], a relevant model was proposed, but it could not cope with the sheer scale of cloud environments, while [12] did not analyse how attacks spread across cloud components. In the proposed method in [13], the shared cloud traffic had low integrity. In addition, traditional models have a high false positive rate (FPR).

1.2 Objectives

This paper firstly aims to implement its proposed IDS for BS on the cloud. Big data of the cloud are then handled by using set-union combiner-based Hadoop (SUCH) MapReduce. Afterwards, a Lorentzy membership-based fuzzy logic system (LFLS)-centric Sankey diagram is generated to determine the attack-spreading pattern in the cloud. Incoming cloud data are then encrypted by using KEMBCC. FPR is mitigated by selecting the optimal features with the use of grid-greedy initialization-centric shark smell optimization (GGSSO).

This paper is organized as follows: Section 2 presents the literature review, Section 3 details the proposed method, Section 4 discusses the results and Section 5 provides the conclusions and recommendations.

2. Literature Review

In [11], an advanced persistent threat detection model in CC was established by using an auto encoder and softmax regression algorithm. This model attained high detection accuracy but could not cope with the sheer scale of the cloud environment.

In [12], an efficient IDS for securing data on CC was developed by using a support vector machine and genetic algorithm. The performance of this model was superior to that of popular models, providing a high level of attack symmetries. Yet this model could not determine the attack-spreading pattern.

In [13], an intelligent behavior-centric malware detection system on CC was developed. Experimental results showed that this model obtained maximum detection rates, proving its superiority. However, this model had low integrity.

In [14], a radial basis function neural network and random forest-centric intelligent IDS was introduced. This model strengthened the overall security mechanisms of CC. However, it had a high FPR.

In [15], IDS based on Facebook Prophet and a collaborative feature selection model was implemented for CC. This model proved effective in ensuring CC security, but it faced overfitting problems because it used fewer features for attack detection.

3. Proposed IDS Methodology for BS in the Cloud Using KEMBCC and SSGRU

The proposed IDF framework, whose architecture is shown in Figure 1, detects cybersecurity attacks to the BS by using KEMBCC and SSGRU.

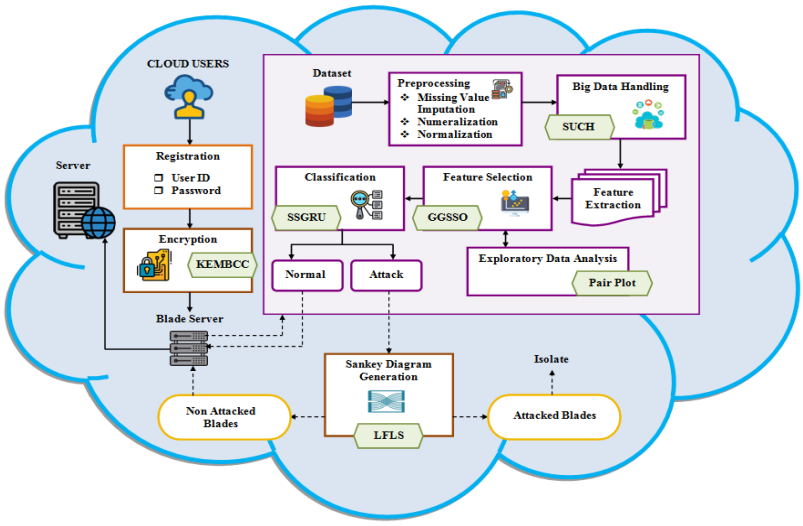


Figure 1. Architecture of the proposed framework

DOI: https://doi.org/10.54216/FPA.210207
Received: May 08, 2025 Revised: July 27, 2025 Accepted: September 01, 2025

3.1 Registration

Cloud users (C) register to the network by using their user ID (C_1) and password (C_2). The number of registered C is denoted by c and represented as

$$C = \{C_1, C_2, ..., C_c\} \quad (1)$$

The user can then access the cloud resources.

3.2 Encryption

KEMBCC is used to secure the data $(\hbar)$ of registered C. Elliptic curve cryptography (ECC) can improve security and use shorter key lengths, but it is less efficient because of elliptic curve parameters. Hence, the exponential Mestre-Brainstrass curve is used. Moreover, a secret key is generated by using the Kerberos function. Based on the exponential Mestre-Brainstrass curve, the keys are generated, as indicated by

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

where x and y denote the coordinates of the curve; are v1 and v2 are the constant parameters; and ζ is the prime number. A base point (β) is then selected on the curve. Then, the public key (λ) is created according to the following expression:

$$\lambda = \Im \cdot \beta \quad (3)$$

where $\Im$ denotes the random point on the curve (i.e. private key). Along with the public and private keys, a secret key (s) is generated by using the Kerberos function. This key is expressed as

$$s = \aleph^{KDF}(\iota + C_2) \quad (4)$$

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

$$\gamma_1 = rand * s \quad (5)$$

$$\gamma_2 = \hbar + rand \cdot \lambda \quad (6)$$

where y1 and y2 denote ciphertext 1 and ciphertext 2, respectively, and $rand$ represents the random number. Afterwards, the decryption process is performed.

$$\hbar = \gamma_2 - \Im * \gamma_1 - s \quad (7)$$

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

3.3 Blade Server

The BS can handle high-density workloads and maximise the utilisation of data centre resources. Hence, it is used by the cloud for load balancing when the incoming $\hbar^*$ is high. However, a robust IDS is needed because the BS is vulnerable to cybersecurity threats.

DOI: https://doi.org/10.54216/FPA.210207
Received: May 08, 2025 Revised: July 27, 2025 Accepted: September 01, 2025

3.3.1 Pre-processing

Firstly, the missing values in the rows and columns of the dataset are used as input. Thereafter, the data are converted into numbers and are then normalised. ϑ denotes the pre-processed data.

3.3.2 Big Data Handling

The pre-processed ϑ undergoes SUCH-centric big data handling. Hadoop MapReduce (HMR) is highly flexible and scalable for processing large amounts of data. However, data shuffling from the mapper to the reducer is a highly complex process. Therefore, a set union combiner is implemented, in which ϑ is firstly divided into smaller portions (ϑ_{ch}) and is expressed as

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

Subsequently, the mapper reads ϑ_{ch} and generates an intermediate key-value pair as follows:

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

where $map(\vartheta_{ch})$ denotes the mapper output; ξ_1 *and* ξ_2 denote the input and intermediate identifier keys, respectively; and Θ_1 *and* Θ_2 denote the input and intermediate values, respectively. Then, the set union combiner (ξ_{com}) function is utilised for the keys on $map(\vartheta_{ch})$ before shuffling it to the reducer phase. Hence, ξ_{com} is expressed as

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

ξ_i denotes the individual sets produced by the mapper for the keys, and j represents the number of sets for the keys produced from $map(\vartheta_{ch})$. Next, the output of ξ_{com} is shuffled and sorted by SUCH. Each group of ξ_{com} is reduced in the reduction phase, thereby generating a set of final key-value pairs (ξ_3) expressed as

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

ϑ_{big} refers to the handled big data.

3.3.3 Feature Extraction

Protocol type, service, flag, duration, traffic volume, outbound traffic, resource utilization and failed login attempts, among other features, are extracted from ϑ_{big} and are denoted by ϑ_{ext}.

3.3.4 Feature Selection

The optimal features are selected from ϑ_{ext} by using GGSSO. The search space is explored by using the SSO algorithm, and the initial position of the shark is randomly selected. Hence, grid-greedy initialization is used to generate the initial position of the shark. The shark (S) population is represented as

DOI: https://doi.org/10.54216/FPA.210207
Received: May 08, 2025 Revised: July 27, 2025 Accepted: September 01, 2025

$$S = \{S_1, S_2, ..., S_h\} \quad (12)$$

where h denotes the number of sharks in the population. The objective function (χ) of GGSSO aims to attain maximum $(\max)$ classification accuracy (υ) and is expressed as

$$\chi = \max\{\upsilon_S\} \quad (13)$$

Then, a visual representation of the selected χ is generated and assessed by using exploratory data analysis that employs a pair plot to ensure the strength of χ . The initial position (S_a) of the shark is then expressed as

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

where A and B denote the tuning intervals; i' denotes the number of grid points (T); $\breve{\Psi}$ *and* $\hat{\wp}$ represent the weight parameter and the learning weight, respectively; and $\nabla\chi$ denotes the gradient of χ . The initial velocity vector (Z) of the shark is determined as

$$Z = \{Z_1, Z_2, ..., Z_h\} \quad (15)$$

Sharks exhibit inertia as they swim. Therefore, the velocity of each single dimension (E) is dependent on the previous velocity and is given as

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

where $Z_{a,b}^{w}$ denotes the velocity of the a^{th} shark in the b^{th} dimension of E at iteration (w); ϖ_w denotes the gradient coefficient; α_w is the weight coefficient; and τ_1 *and* τ_2 represent random numbers between $[0,1]$.

The shark has limited speed, which is expressed as

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

where $\tilde{\varphi}_w$ denotes the speed limit factor at the w^{th} iteration. Afterwards, the position of the shark is updated by using its previous speed and position. The new position $(S_{a,b}^{w+1})$ is determined as

$$S_a^{w+1} = S_a^{w} + Z_a^{w} \cdot \Delta^{w} \quad (18)$$

where Δ^{w} is the time interval at the w^{th} iteration. The local search position $(L_{a,b}^{w+1,k})$ of the shark is updated on the basis of the rotation motion and is expressed as

DOI: https://doi.org/10.54216/FPA.210207
Received: May 08, 2025 Revised: July 27, 2025 Accepted: September 01, 2025

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

where τ_3 is the random number, and k is the number of points in the local search. The shark moves towards the location of a prey if it detects a strong odour as it rotates. Hence, the search behaviour is expressed as

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

Therefore, the shark selects the candidate solution, focusing on the highest χ values. The cycle of forward movement and rotation continues until w reaches maximum iteration $\left(w_{\max}\right)$. The optimal features $\left(\vartheta_{opt}\right)$ are selected once $w_{\max}$ terminates. The pseudocode for GGSSO is presented below.

Input: Extracted features $\left(\vartheta_{ext}\right)$
Output: Optimal features $\left(\vartheta_{opt}\right)$
Begin
Initialise 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
Initialise 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

DOI: https://doi.org/10.54216/FPA.210207
Received: May 08, 2025 Revised: July 27, 2025 Accepted: September 01, 2025

Return optimal features
End

The attack to the BS is further detected on the basis of ϑ_{opt}.

3.3.5 Classifications

ϑ_{opt} is input to SSGRU for attack detection. GRU uses gating mechanisms to control information flow. However, GRU has a low learning efficiency and slow convergence rate. Thus, the Sechsoftwave activation function and sparse initialization are applied. Figure 2 displays the structure of SSGRU.

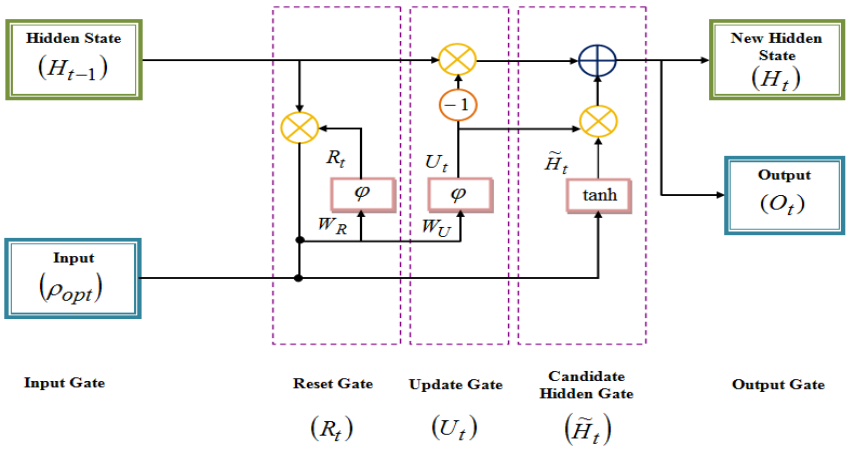


Figure 2. Structure of the proposed SSGRU

The input gate contains the input (ϑ_{opt}), which is then transferred to the reset gate (R) and is expressed as

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

where t is the current state.

Then, R determines how much of the previous information must be forgotten from the gates in t.

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

where H_{t-1} denotes the previous hidden state (H_t), and WR represents the weights of R. WR is determined by using the sparse initialization function

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

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

DOI: https://doi.org/10.54216/FPA.210207
Received: May 08, 2025 Revised: July 27, 2025 Accepted: September 01, 2025

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

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

Subsequently, the update gate (U) determines the amount of the new input to be utilised for updating the hidden gate at t.

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

where W_U signifies the weights of U, which is initialised by the sparse initialisation function.

The candidate's hidden state $\left(\vec{H}_t\right)$ is denoted by

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

The hidden state (H_t) is then determined as

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

Thereafter, the loss function $(\Diamond)$ is assessed. Lastly, the output gate (O_t) is obtained with the attack (Γ_1) and normal (Γ_2) classes.

$$O_t \rightarrow \left|\Gamma_1, \Gamma_2\right| \quad (28)$$

The pseudocode for SSGRU is presented below.

Input: Optimal features (ϑ_{opt})

Output: Classified output (O_t)

Begin

Initialise R_t, U_t H_t, W_R and W_U

For each ϑ_{opt} **do**

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

Initialise weights

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

Activate

Evaluate update gate

DOI: https://doi.org/10.54216/FPA.210207
Received: May 08, 2025 Revised: July 27, 2025 Accepted: September 01, 2025

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 $(\Diamond = satisfied)$ **{**

Terminate

} Else {

Tune the parameters

}

End If

End For

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

End

Γ_2 indicates that the loads are efficiently balanced by the BS and further processed in the server. By contrast, Γ_1 means that the BS should be processed further according to the method detailed in the next subsection.

3.4 Sankey Diagram Generation

If Γ_1 is detected from SSGRU, then a Sankey diagram is generated on the basis of LFLS to detect attacked blades in the BS. Uncertainties can be efficiently handled by using a fuzzy logic system. However, designing fuzzy rules can be a complex process that requires domain expertise and numerous comprehensive trials. Thus, a well-defined Lorentzy membership function is used to reduce the complexity. The conditions (ϕ) for generating the Sankey diagram (δ) are set, which are expressed as follows:

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

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

where $\delta_1\ and\ \delta_2$ denote the attacked blades and non-attacked blades, respectively; and $\phi_1, \phi_2, \phi_3, \phi_4$ denote the traffic volume, failed login attempt, outbound traffic and resource utilisation, respectively. The input $(\aleph)$ is mapped into a fuzzy value $(\hat{\aleph})$ by using the fuzzification (ς) block of LFLS, which is expressed as

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

Next, the inference engine is used in decision-making with the use of the Lorentzy membership function (ψ), which is expressed as

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

DOI: https://doi.org/10.54216/FPA.210207
Received: May 08, 2025 Revised: July 27, 2025 Accepted: September 01, 2025

where $\Xi \, and \, \tilde{\omega}$ denote the scale and centre parameters, respectively. The last defuzzification $(\tilde{\varsigma})$ functional block converts the fuzzy quantity into crisp values and is expressed as

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

The output is expressed as

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

δ_1 indicates that the blade is isolated from the BS. Otherwise, the blades continue running in the cloud server. The performance of the proposed framework is discussed in subsequent sections.

4. Results and Discussions

The proposed framework is implemented in the Python environment to prove its robustness for attack detection in the BS.

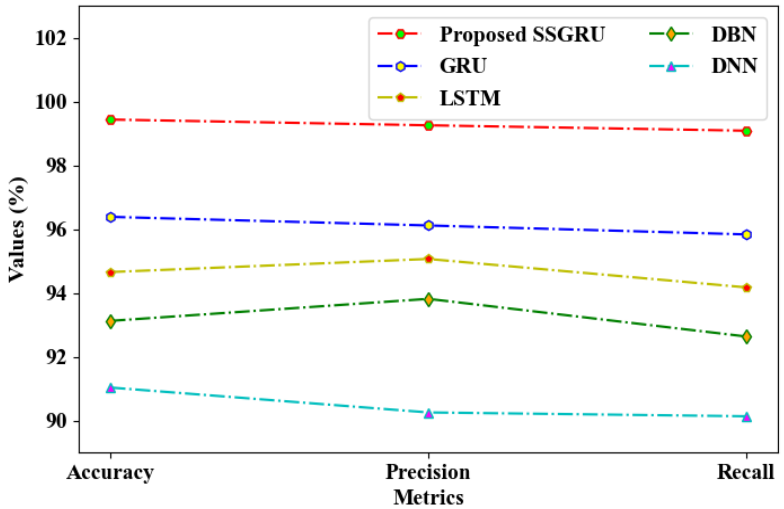
4.1 Dataset Description

The Network Security Laboratory's Knowledge Discovery and Data Mining (NSL-KDD) anomaly detection dataset is used to test the performance of the proposed framework. Out of the entire dataset, 80% and 20% are used for training and testing, respectively.

4.2 Performance Analysis

In this subsection, the proposed model's performance is compared with that of other widely used techniques to prove its reliability.

Figure 3 presents the comparative analysis of the proposed SSGRU and the existing models. The learning efficiency of the proposed model was enhanced by sparse initialization. As a result, 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%, a false negative rate of 0.023 and FPR of 0.029. By contrast, the existing models exhibited low learning efficiency, thus performing more poorly than the proposed method did. These results show that the proposed SSGRU is less error-prone than the existing models.



DOI: https://doi.org/10.54216/FPA.210207
Received: May 08, 2025 Revised: July 27, 2025 Accepted: September 01, 2025

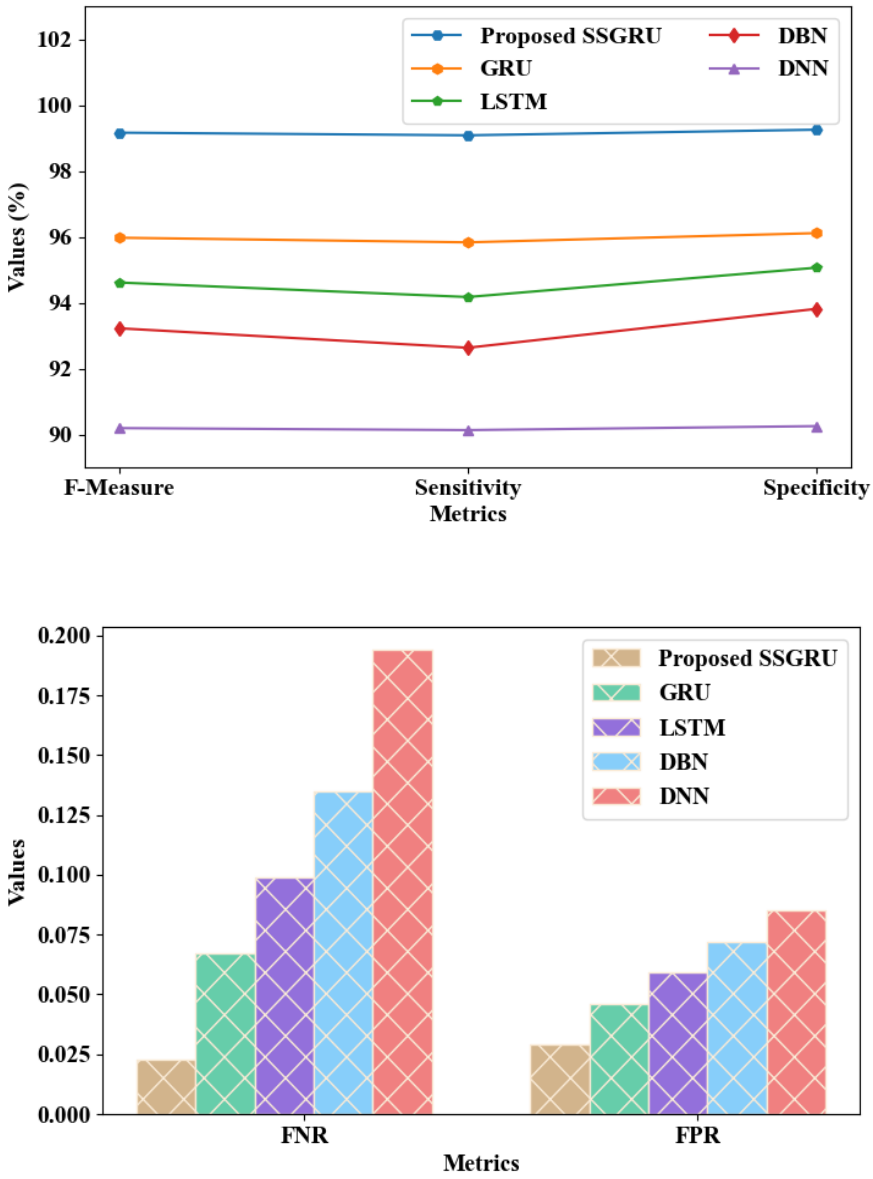


Figure 3. Comparative analysis of the proposed SSGRU

Table 1: Performance analysis of the 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

DOI: https://doi.org/10.54216/FPA.210207
Received: May 08, 2025 Revised: July 27, 2025 Accepted: September 01, 2025

Table 1 presents the performance analysis of the proposed and existing models. ECC achieved an encryption time of 2134 ms, a decryption time of 2166 ms and a security level of 95.26%. Meanwhile, the proposed model generated a secret key to improve security, achieving an encryption time, decryption time and security level of 1349 ms, 1385 ms and 98.59%, respectively.

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
Ant colony optimisation	94.42	2753
Red panda optimisation	91.31	4629
Artificial bee colony optimisation	89.52	5758

Table 2 presents the performance analysis of the proposed GGSSO, the traditional SSO, ant colony optimisation, red panda optimisation and artificial bee colony optimisation (ABCO). ABCO achieved a low average fitness of 89.52% and a long feature selection time of 5758 ms. By contrast, the proposed GGSSO obtained an average fitness and feature selection time of 98.41% and 1292 ms, respectively, thus being able to select the optimal features efficiently.

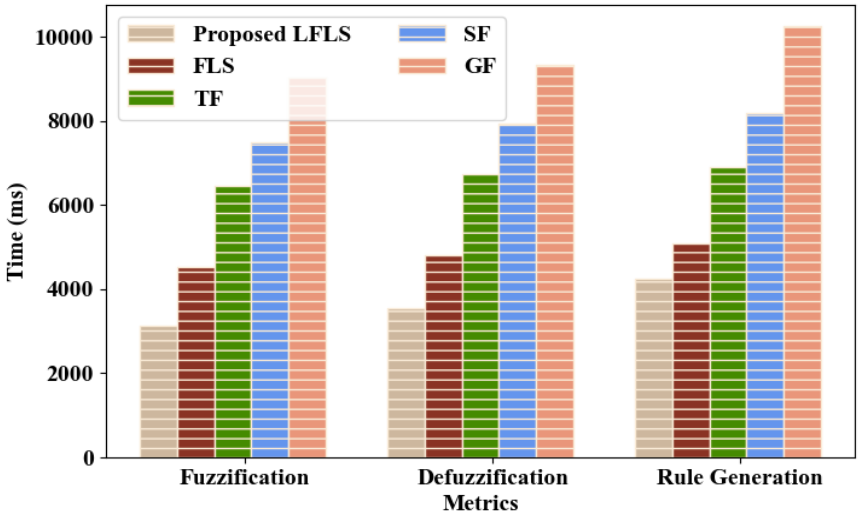


Figure 4. Performance analysis of proposed LFLS

Figure 4 presents the performance analysis of the proposed LFLS and the existing FLS, triangular fuzzy, sigmoid fuzzy and Gaussian fuzzy. The traditional models took an average time of 6883.5, 7211.75 and 7617.5 ms for fuzzification, defuzzification and rule generation, respectively. By contrast, the proposed LFLS took a minimum time of 3142, 3540 and 4247 ms for fuzzification, defuzzification and rule generation, respectively. The proposed LFLS achieved dynamic performance because it used a well-defined membership function.

DOI: https://doi.org/10.54216/FPA.210207
Received: May 08, 2025 Revised: July 27, 2025 Accepted: September 01, 2025

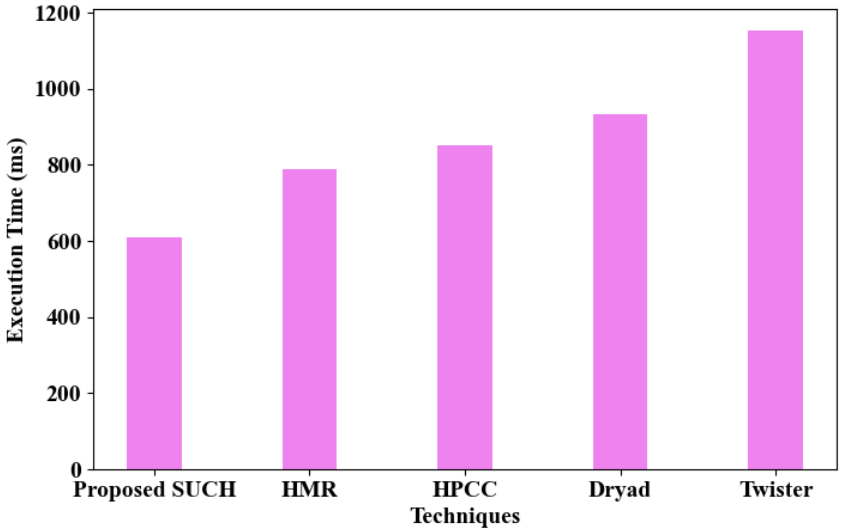


Figure 5. Execution time analysis

Figure 5 compares the execution times of the proposed SUCH and the existing HMR, high-performance computing clustering methods, Dryad and Twister. The average execution time of the existing methods was 931 ms, whereas that of the proposed method was much shorter at 610 ms. The proposed SUCH also reduced the data shuffling complexity, thus achieving robust performance.

4.3 Comparative Analysis with Related Works

The proposed model is compared with related works to verify its efficiency.

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 optimisation algorithm-based DNN	-	95.35
[18]	Gradient hybrid leader optimisation	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

Table 3 presents the comparative investigation of the proposed model with related works. Existing works developed various IDS frameworks by using ensemble learning, CNN and LSTM techniques, yet they achieved low accuracy because of their low learning efficiency. By contrast, the proposed model achieved an accuracy of 99.43% because of improved convergence rates. These results prove the efficacy of the proposed system.

DOI: https://doi.org/10.54216/FPA.210207
Received: May 08, 2025 Revised: July 27, 2025 Accepted: September 01, 2025

5. Conclusion

This paper proposes an IDS framework for BS in the cloud, which uses SSGRU and GGSSO. Firstly, user data were encrypted. Then, the proposed IDS was tested by investigating the dataset features, followed by attack classification. The experimental investigation demonstrated that this model achieved 99.43% detection accuracy, a security level of 98.59% and an execution time of 610 ms. Moreover, the average fitness of the proposed GGSSO was 98.41%. Analysis results indicate that this model was less error-prone than other models, proving its robustness. This work will be improved further in the future by developing attack mitigation strategies for BS that focus 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," *J. Cybersecurity Privacy*, vol. 2, no. 3, pp. 527–555, 2022, doi: 10.3390/jcp2030027.

[2] Z. Alzubaidi, M. A. Alshahrani, M. A. Alzahrani, and R. A. Alhussein, "A comprehensive survey on cybersecurity threats and countermeasures in smart home environments," *J. Netw. Comput. Appl.*, vol. 204, pp. 1–20, Jan. 2023, doi: 10.1016/j.jnca.2023.103307.

[3] Alouffi *et al.*, "A systematic literature review on cloud computing security: Threats and mitigation strategies," *IEEE Access*, vol. 9, pp. 57792–57807, 2021, doi: 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*, vol. 11, no. 1, pp. 1–34, 2022, doi: 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 Comput. Methods Eng.*, vol. 29, pp. 223–246, 2022, doi: 10.1007/s11831-021-09573-y.

[6] M. Abdullahi *et al.*, "Detecting cybersecurity attacks in internet of things using artificial intelligence methods: A systematic literature review," *Electronics*, vol. 11, no. 2, pp. 1–27, 2022, doi: 10.3390/electronics11020198.

[7] V. Chang *et al.*, "A survey on intrusion detection systems for fog and cloud computing," *Future Internet*, vol. 14, no. 3, pp. 1–27, 2022, doi: 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 Things*, vol. 23, pp. 1–33, 2023, doi: 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, doi: 10.3390/sym15050988.

[10] Alshammari and A. Aldribi, "Apply machine learning techniques to detect malicious network traffic in cloud computing," *J. Big Data*, vol. 8, no. 1, pp. 1–24, 2021, doi: 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] 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, doi: 10.3390/sym13122306.

[13] O. 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*, vol. 12, no. 6, pp. 1–42, 2023, doi: 10.3390/electronics12061333.

[14] R. Al-Ghuwairi *et al.*, "Intrusion detection in cloud computing based on time series anomalies utilizing machine learning," *J. Cloud Comput.*, vol. 12, no. 1, pp. 1–17, 2023, doi: 10.1186/s13677-023-00491-x.

[15] H. Attou *et al.*, "Towards an intelligent intrusion detection system to detect malicious activities in cloud computing," *Appl. Sci.*, vol. 13, no. 17, pp. 1–19, 2023, doi: 10.3390/app13179588.

[16] P. Singh and V. Ranga, "Attack and intrusion detection in cloud computing using an ensemble learning approach," *Int. J. Inf. Technol.*, vol. 13, no. 2, pp. 565–571, 2021, doi: 10.1007/s41870-020-00583-w.

DOI: https://doi.org/10.54216/FPA.210207
Received: May 08, 2025 Revised: July 27, 2025 Accepted: September 01, 2025

[17] Agarwal, M. Khari, and R. Singh, "Detection of DDOS attack using deep learning model in cloud storage application," *Wireless Pers. Commun*, vol. 127, no. 1, pp. 419–439, 2022, doi: 10.1007/s11277-021-08271-z.

[18] S. Balasubramaniam *et al.*, "Optimization enabled deep learning-based DDoS attack detection in cloud computing," *Int. J. Intell. Syst.*, vol. 2023, pp. 1–16, 2023, doi: 10.1155/2023/2039217.

[19] S. S. Hameed *et al.*, "Security enhancement and attack detection using optimized hybrid deep learning and improved encryption algorithm over Internet of Things," *Meas.: Sensors*, vol. 30, pp. 1–8, 2023, doi: 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, doi: 10.3390/s22134685.

DOI: https://doi.org/10.54216/FPA.210207
Received: May 08, 2025 Revised: July 27, 2025 Accepted: September 01, 2025