Efficient Intrusion Detection and Prevention Model in Cloud Environment Using Sgd-LSTM and C2HA

Ponnuviji NAMAKKAL PONNUSAMY*, Vigilson Prem MONICKARAJ, Ezhumalai PERIYATHAMBI
Department of Computer Science and Engineering, R.M.D. Engineering College, Kavaraipettai, India
ponnuviji@gmail.com (**Corresponding author*), vigiprem@gmail.com, ezhumalai.es@gmail.com

Abstract: Cloud computing is an attractive technology paradigm that has been widely used as a tool for storing and analyzing the data of different users. Since access to the cloud is achieved through the Internet, data stored in clouds is susceptible to attacks from external as well as internal intruders. Henceforth, cloud service providers (CSPs) need to take action in order to provide a secure framework that would detect intrusion in the cloud and protect and secure customer information against hackers and intruders. This paper proposes a Sgd-LSTM and signature-based access control policy based Intrusion Detection and Prevention System (IDPS) model which is meant to detect and prevent various intrusions in the cloud. The proposed system includes three phases: the user registration phase, intrusion detection phase, and intrusion prevention phase. Initially, user registration is performed based on a unique ID and password, and then, the password is converted into hashcode by using the C2HA algorithm and then stored in the cloud for authentication purposes. In the intrusion detection phase, the status of cloud data is predicted by employing the Sgd-LSTM classifier in order to discard the intruder data packets from the cloud. At last, in the intrusion prevention phase, data access to the cloud environment is controlled by using signature-based user authentication in order to authenticate the legitimate user. The proposed classifier can effectively detect the intruders, which was experimentally proved by comparing it with the existing classifiers.

Keywords: Intrusion Detection and Prevention System (IDPS), Cloud, User authentication, Stochastic Gradient Descent Long Short-Term Memory (Sgd-LSTM) classifier, Color Hidden Hashing Algorithm.

1. Introduction

Cloud computing is the recent growing computational model that provides convenient, on-demand network access for sharing the group of computing resources, i.e. servers, networks, storage, applications, etc. Three-tier intrusion detection and prevention model was created by Ali & Yousaf (2020). Virtualization is one of the key technologies in the cloud environment, which enables the creation of an intelligent abstraction layer, called Virtual Machine Monitor (VMM) or Hypervisor. However, cloud computing is vulnerable to traditional information technology (IT) attacks, i.e. intrusion, because it uses and widens the existing IT infrastructure, operating systems (OSs), and applications. Network intrusion detection is discussed in Mauro et al. (2020).

The process of stealing, modifying, or corrupting other users` information by sending malicious packets through the network is referred to as intrusion (Traore et al., 2012). To identify and protect the cloud user's data, Intrusion Detection Systems (IDS), and Intrusion Prevention Systems (IPS) are important (Xie et al., 2020). The primary reason for any IDS is to detect assaults/attacks and to avoid an assault if possible (Mishra et al., 2020). Most of the current IDSs can be divided into two main types: signature-based and anomaly-based IDS. For an intrusion detection system to be effective, the factors that should be taken into account are speed, self-monitoring, fault tolerance, a user-friendly configuration, and the ability of the system to be cheat-resistant and interruption-free with minimum overhead so that the malicious data may be detected and removed from the network automatically (Mishra et al., 2020).

Just like intrusion detection, intrusion prevention is also important for maintaining the security of the cloud user's data. A two-layer defence scheme application is presented in (Liu et al., 2018) Security has become one of the serious bottleneck problems that need to be resolved. Privacy, confidentiality, integrity, and access control are the common requirements of security (Saxon, Bordbar, & Harrison, 2015). Intrusion management system is presented in (Mauro, Galatro, & Liotta, 2020). Techniques based on access control, such as authentication methods, represent one of the best ways to defend data security based on control and limit unauthorized clients. IoT-based application is discussed in (Hafeez et al., 2020) and security challenges are discussed in (Mishra, & Pandya, 2021).

Defense System against Multi-Type Attacks in Cloud is presented in (Wahab et al., 2021). In Intrusion Prevention and Detection System (IPDS), if the first line of defense fails to prevent attacks, then the second line comes into play to detect any intrusion and remove the affected data from the cloud (Feng et al., 2019) Dynamic intrusion detection in cloud environment is discussed in (Chkirbene et al, 2020). Many supervised learning approaches, e.g., decision tree (DT), deep neural network (DNN), support vector machine (SVM), etc., have been successfully employed for recognizing and detecting possible intrusions. In this paper, an improved algorithm is proposed for intrusion detection and prevention. For intrusion detection, data packets are analyzed and the malicious patterns are identified through data preprocessing, feature extraction and packet classification. For determining the malicious users and protecting the environment from attackers, all users are registered to the system before transmitting the packets. The objective is to obtain a higher accuracy for security attacks detection and improve the privacy of the cloud environment using hash code generation.

This paper is summarized as follows. Section 2 presents the literature review which focuses on the analysis of various research papers. Section 3 presents and discusses the proposed algorithm. Section 4 discusses the experimental results obtained, while section 5 concludes this paper.

2. Literature Survey

Mohana Prabha & Vidhya Saraswathi (2020) introduced a Suppressed K-Anonymity Multi-Factor Authentication Based Schmidt-Samoa Cryptography (SKMA-SC) technique for cloud data security. The SKMA-SC technique comprised three key processes, namely registration, authentication, and data access. In SKMA-SC, the data integrity and confidentiality level in the needed to be improved. By focusing on security model, the clould model was improved.

Dey, Ye & Sampalli (2019) designed a “cognitive system”, a machine learning-based intrusion detection scheme for data fusion in mobile clouds involving heterogeneous client networks. The obtained results indicated that even though the scheme was highly effective it didn`t solve certain security issues and featured data leakage.

Hajimirzaei & Navimipour (2019) designed an IDS based on a combination of a multilayer perceptron (MLP) network, an artificial bee colony (ABC), and fuzzy clustering algorithms. Normal and abnormal network traffic data was detected by the MLP, while the MLP training was performed by the ABC algorithm by optimizing the values of linkage weights and biases. Due to the limitations of the ABC algorithm, the performance of MLP was significantly reduced.

He at al. (2016) constructed a Privacy-Aware authentication (PAA) scheme for Mobile Cloud Computing (MCC) services by using an identity-based signature scheme. The advantage is that, the hash-based authentication process outperformed well. The main drawback of PAA was no appropriate secure service was provided in MCC.

3. Intrusion Detection and Prevention System (IDPS) in Cloud

Cloud computing has grown significantly with regard to: a) profoundly changing the entire Information Technology culture from the construction of data servers, b) the deployment of programs c) dealing with the technology upgrading process. But data security threats still represent the main problem in the cloud. The major security threats are data breach, information theft, and omission of data in the cloud infrastructure. These attacks will only become more sophisticated, so it is important to adapt protection technologies to the respective threats. The proposed system includes 3 phases: user registration phase, intrusion detection phase, and intrusion prevention phase. These phases are explained below in detail. Figure 1 shows the flow of these phases and the working of each block.

The proposed paper contributes to the improvement of cybersecurity performance in cloud model. In the intrusion detection phase, the status of cloud data is predicted using the Sgd-LSTM classifier in order to discard the intruder packets from the cloud.

3.1 Intrusion Detection Phase

In this phase, the intruder data packets and normal data packets are identified based on data packet features using SGD-LSTM. The data packets are initially loaded from the UNSW-NB15 dataset. The collected data mainly contains 47 features including data packet IP address, source port number, destination IP address, protocol type, etc. To accurately detect the data packet status, the dataset undergoes the following stages: preprocessing, feature selection and data packet status classification stage.

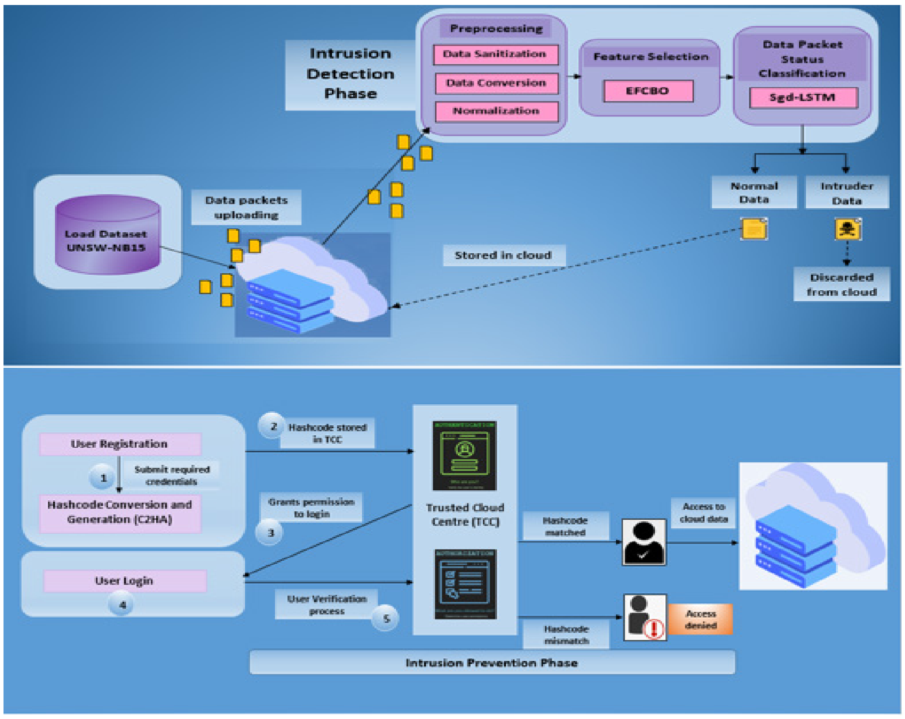


Figure 1. Architecture diagram of Sgd-LSTM and signature-based access control policy

3.1.1 Preprocessing Stage

The collected data contains massive amounts of unnecessary, duplicated values, and outliers. The preprocessing involves three steps: data sanitization, numerical conversion, and normalization. Let us consider the input data as:

$$\left(D_p\right)_i = \left\{D_{p1}, D_{p2}, D_{p3}, \ldots\ldots\ldots, D_{pk}\right\}, \quad (1)$$

where $\left(D_p\right)_i$ represents the input data, and D_{pk} represents the k-number of data.

(a) ***Data Sanitization:*** Data sanitization is used for filtering outliers, replacing missing values, smoothing noisy data, and correcting inconsistent data. Here, the noisy data and duplicated data are removed from the dataset.

(b) ***Numerical Conversion:*** The learning algorithms can only handle numeric data. In the collected dataset, the features like protocol type are in string format. Those string values are converted to numeric data.

(c) ***Normalization:*** This step is used for scaling the feature values into a specific confidence interval. The main benefit of this step is that it removes the bias from the raw data. Based on Z-score normalization, each feature value is reconstructed into a specific range.

$$Z\left(D_p\right) = \frac{\mathrm{X}(D_p)_i - \widetilde{\mathrm{X}}(D_p)_i}{\sigma_{\mathrm{X}(D_p)_i}}, \quad (2)$$

where $\widetilde{\mathrm{X}}(D_p)_i$ denotes the standard deviation of data and $\sigma_{\mathrm{X}(D_p)_i}$ denotes the mean of $\mathrm{X}(D_p)_i$. The preprocessing stage is represented as:

$$\left(D_p\right)_i \xrightarrow{\text{preprocessing}} P\left(D_p\right)_i \quad (3)$$

In equation (3), $P\left(D_p\right)$ denotes the preprocessed data.

3.1.2 Feature Selection Stage

Feature Selection is intended to select the features of data packets and contribute most to the intrusion prediction process. The collected data contains important features that might be used for identifying intrusion data packets; they should be carefully analyzed in order to separate only the relevant information. For a feature, the proposed system uses the Enhance Fragrance Coefficient-based Butterfly Optimization (EFCBO) algorithm. The traditional Butterfly Optimization Algorithm (BOA) is characterised by slow convergence due to its stochastic behaviour and blindness of the fragrance coefficient. To eliminate these behaviours, the proposed algorithm enhanced the fragrance coefficient of BOA using a self-adaption method.

EFCBO algorithm is an inspired meta-heuristic algorithm that mimics the butterflies' natural foraging and mating behaviour. The stimulus intensity for the butterfly is decided by the landscape of the objective function. In the EFCBO algorithm, initially each butterfly's position vector is in $t_n \times k$, where t_n denotes the number of iteration and k denotes the population size. Each butterfly's position vector is represented as:

$$BT_k(t) = \{BT_1, BT_2, ,......., BT_n\}, \quad (4)$$

where $BT_k(t)$ represents k -th butterfly's position at the c urrent iteration t. The main phases involved in EFCBO are global and local search. The next position of a certain butterfly in the global phase is computed as:

$$BT_k(t+1) = BT_k(t) + \left(\tilde{r}^2 \times q - BT_k(t)\right) \times U(F_g), \quad (5)$$

where $BT_k(t+1)$ represents the position of butterfly at iteration $t+1$, q is the best position in search space, $\tilde{r}$ is the random value $\left(\tilde{r} \in [0,1]\right)$, $U(F_g)$ is the fragrance coefficient, which is updated using a self-adaption method as follows and ψ is the fragrance distribution range:

$$U(F_g) = \psi \times \left(1 - \frac{t}{t_n}\right) \quad (6)$$

$$BT_k(t+1) = BT_k(t) + \left(\tilde{r}^2 \times BT_m(t) - BT_n(t)\right) \times U(F_g), \quad (7)$$

where $BT_m(t)$ and $BT_n(t)$ are the positions of the m -th and n -th butterflies in the searching space. Here, the position of the butterfly is updated based on switching probability $s(\rho)$ and a random value $\tilde{r}$, where the switching probability is in the range between 0 and 1 $if \ \left(s(\rho) < \tilde{r}\right)$.

Update the next position of a butterfly using equation (5) $if \ \left(s(\rho) \geq \tilde{r}\right)$.

Update the next position of a butterfly using equation (7).

Record and replace the best solution and the best fitness value if there is a better solution. Repeat the process until the maximum iteration. By applying the EFCBO algorithm, in the feature selection phase, the necessary features are selected from dataset features. The selected features are represented as:

$$\hbar_i = \{\hbar_1, \hbar_2, \hbar_3, ..., \hbar_n\} \quad (8)$$

where $\hbar_i$ is the selected feature set, and $\hbar_n$ is the nth feature in the feature set $\hbar_i$.

3.1.3 Data Packet Status Classification Stage

The selected features are given to the classification phase, which classifies the data packet status as normal data and data related to cyberattacks based on different types of attacks, i.e. DoS attacks, exploits, generics, shellcodes, and so on. The proposed system uses Stochastic Gradient Descent Long Short-Term Memory (Sgd-LSTM) networks. The default behaviour of the LSTMs is based on a novel type of Recurrent Neural Network (RNN) which is capable of knowing long-term dependencies by remembering information for long periods of time The RNN is affected by the exploding gradient problem. This gradient problem can create major issues such as long training time, poor system performance, and bad prediction rate. To overcome this gradient problem, the LSTM network is employed and the weight values of LSTM are optimized using Stochastic Gradient Descent (SGD) parameter to achieve a superior performance. The structure of Sgd-LSTM is illustrated in Figure 2. At the forget gate, input gate, and output gate respectively, a weight value W_{FG}, W_{IG} and W_{OG} is generated; these values are applied to the following equation to obtain an optimized weight value:

$$W'_w = W_w - \mu; \qquad w = \{FG, IG, OG\} \quad (9)$$

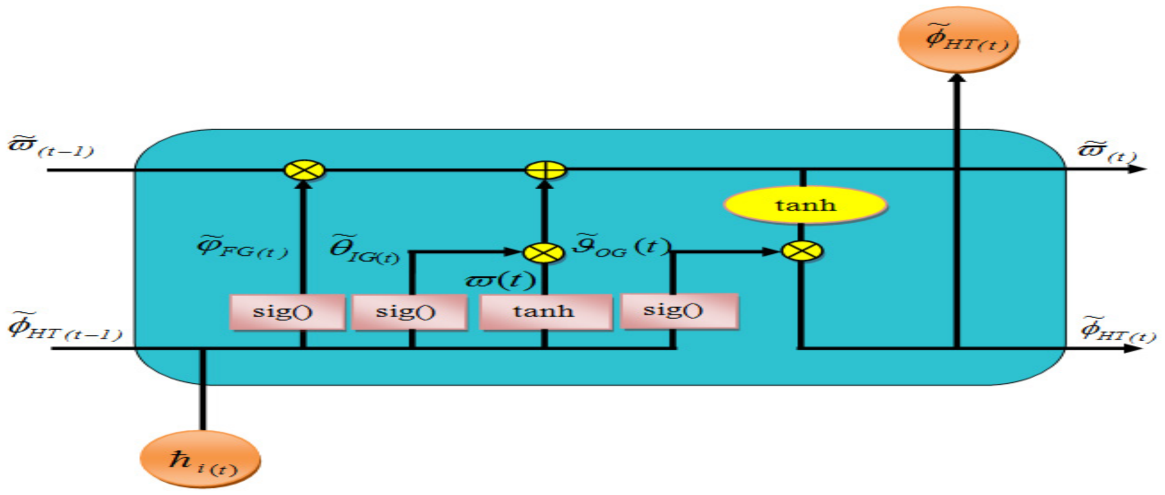
where, W'_w is the optimized weight value, W_w is the old weight value and μ denotes the step size of selected features. The obtained weight values for forget gate, input gate and output gate are W'_{FG}, W'_{IG} and W'_{OG}, respectively. The steps involved in Sgd-LSTM-based data packet status classification are given below.

Forget Gate: The first step in the LSTM is to choose which information should be omitted from the cell for that particular time step. The sigmoid function determines this using the previous hidden state ($\tilde{\Phi}_{HT}(t-1)$) along with the current input $\hbar_i(t)$, where t is the current time step.

$$\tilde{\phi}_{FG}(t) = sig\left(W'_{FG}\left[\tilde{\Phi}_{HT}(t-1), \hbar_i(t)\right] + b_{FG}\right), \quad (10)$$

where $\tilde{\theta}_{IG}(t)$ represents the forget gate output, $sig(\bullet)$ represents the sigmoid function, b_{IG} is the bias value and $\tilde{\Phi}_{HT}(t-1)$ is the previous hidden state.

Input Gate: This gate is used for updating the memory cell state and it involves two functions.



$\widetilde{\phi}_{HT(t)}$
$\widetilde{\varpi}_{(t-1)}$
$\widetilde{\varpi}_{(t)}$
tanh
$\widetilde{\varphi}_{FG(t)}$
$\widetilde{\theta}_{IG(t)}$
$\widetilde{\vartheta}_{OG}(t)$
$\varpi(t)$
$\widetilde{\phi}_{HT(t-1)}$
sig()
sig()
tanh
sig()
$\widetilde{\phi}_{HT(t)}$
$\hbar_{i(t)}$

Figure 2. The structure of the Sgd-LSTM network

One is the sigmoid function, and the other is the tanh function. The sigmoid function decides which values to let through (0 or 1), wheras the *tanh* function gives weightage to the values which are passed, deciding their level of importance (-1 to 1).

$$\tilde{\theta}_{IG}(t) = sig\left(W'_{IG}\left[\tilde{\Phi}_{HT}(t-1), \hbar_i(t)\right] + b_{IG}\right) \quad (11)$$

$$\varpi(t) = tanh\left(\tilde{\varpi}_{MC} \cdot \left[\tilde{\Phi}_{HT}\hbar_i(t)\right] + b_{\varpi}\right), \quad (12)$$

where $\tilde{\theta}_{IG}(t)$ represents the input gate output, $tanh(\bullet)$ denotes the *tanh* function, $\varpi(t)$ and is a vector of new candidate values that could be added to the cell state, and b_{IG} and b_{ϖ} are the bias values.

Cell Gate: In this step, first, the previous cell states are pointwise multiplied by the forget gate output, and then the output from the input gate is multiplied with the current cell state. Then the outputs obtained through the pointwise multiplication are taken and a pointwise addition is performed, which updates the cell state to new values that give the new cell states.

$$\tilde{\varpi}(t) = \tilde{\theta}_{FG} \circ \tilde{\varpi}(t-1) \oplus \tilde{\theta}_{IG} \circ \varpi(t), \quad (13)$$

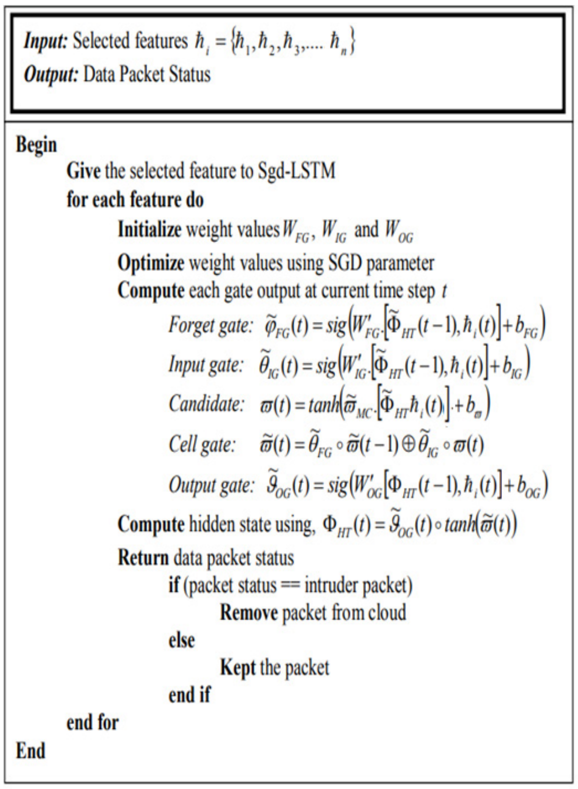
where $\tilde{\varpi}(t)$ represents the new modified cell state, $\tilde{\varpi}(t-1)$ represents the previous cell state and $\circ$ and $\oplus$ represent pointwise multiplication and addition, respectively.

Output Gate: The output gate chooses which should be the next hidden state. First, the previous hidden state and the current input is passed to a sigmoid function, which chooses which parts of the cell state shall make up the output. Then, the cell state is passed to *tanh* function to push the values into the range between -1 and 1 and it is multiplied by the output of the sigmoid gate.

$$\tilde{\vartheta}_{OG}(t) = sig\left(W'_{OG}\left[\Phi_{HT}(t-1), \hbar_i(t)\right] + b_{OG}\right) \quad (14)$$

$$\Phi_{HT}(t) = \tilde{\vartheta}_{OG}(t) \circ tanh\left(\tilde{\varpi}(t)\right), \quad (15)$$

where $\tilde{\vartheta}_{OG}(t)$ represents the output of output gate, b_{OG} is the bias value and $\Phi_{HT}(t)$ is the hidden state which represents classification output, i.e. data packet status. This classification result contains 10 classes: 9 different types of intruder data packets (Analysis, backdoors, DoS, exploits, fuzzers, generics, reconnaissance attacks, shellcodes, worms) and a normal data packet. The pseudocode of Sgd-LSTM is shown in Figure 3.



```
Input: Selected features $\hbar_i = \{\hbar_1, \hbar_2, \hbar_3, .... \hbar_n\}$
Output: Data Packet Status

Begin
    Give the selected feature to Sgd-LSTM
    for each feature do
        Initialize weight values $W_{FG}$, $W_{IG}$ and $W_{OG}$
        Optimize weight values using SGD parameter
        Compute each gate output at current time step $t$
            Forget gate: $\tilde{\varphi}_{FG}(t) = sig\left(W'_{FG}\left[\tilde{\Phi}_{HT}(t-1), \hbar_i(t)\right] + b_{FG}\right)$
            Input gate: $\tilde{\theta}_{IG}(t) = sig\left(W'_{IG}\left[\tilde{\Phi}_{HT}(t-1), \hbar_i(t)\right] + b_{IG}\right)$
            Candidate: $\varpi(t) = tanh\left(\tilde{\varpi}_{MC} \cdot \left[\tilde{\Phi}_{HT}\hbar_i(t)\right] + b_{\varpi}\right)$
            Cell gate: $\tilde{\varpi}(t) = \tilde{\theta}_{FG} \circ \tilde{\varpi}(t-1) \oplus \tilde{\theta}_{IG} \circ \varpi(t)$
            Output gate: $\tilde{\vartheta}_{OG}(t) = sig\left(W'_{OG}\left[\Phi_{HT}(t-1), \hbar_i(t)\right] + b_{OG}\right)$
        Compute hidden state using, $\Phi_{HT}(t) = \tilde{\vartheta}_{OG}(t) \circ tanh\left(\tilde{\varpi}(t)\right)$
        Return data packet status
            if (packet status == intruder packet)
                Remove packet from cloud
            else
                Kept the packet
            end if
    end for
End
```

Figure 3. Pseudocode of Sgd-LSTM

After identifying the status of data packets, the intruder packets are removed from the cloud and only the normal data packets are kept in the cloud. After identifying intruders in the cloud environment, it is essential to provide a secure framework for preserving users' data and secure access.

3.2 Registration Phase

Registration limits access to data by allowing only the approved users. In this phase, the users register their information in the cloud for data access. For this purpose, initially, the user chooses a signature i.e. a unique ID $\ddot{u}_{ID}$ and password $\ddot{u}_{\tilde{p}}$ and then sends a registration request to the Trusted Cloud Centre (TCC) along with $\{\ddot{u}_{ID}, \ddot{u}_{\tilde{p}}\}$.

$$user \xrightarrow{R\_REQ\{\ddot{u}_{ID},\, \ddot{u}_{\tilde{p}}\}} TCC\,, \tag{16}$$

where $R\_REQ\{\cdot\}$ represents the registration request and TCC represents the Trusted Cloud Centre. After receiving the registration request from the user, the TCC computes the hash value for the user password. The hash value of the user password is denoted as $\hat{\mathrm{H}}(u_{\tilde{p}})$. Here, the hash value is computed using the Color Hidden Hash Algorithm (C2HA).

3.2.1 Color Hidden Hash Algorithm (C2HA)

The C2HA is one of the hashing algorithms, which computes the hash value for the given input using RGBA color values. In C2HA, the end-user can choose hash code length, and also makes hash code private between channels, by sharing unique additional data between them. The steps involved in C2HA are explained as follows:

i. Initially, the input text $u_{\tilde{p}}$ is joined with constant unique text $t_{\tilde{c}}$ then the constructed input is converted into a binary array, which is expressed in equations (2) and (3).

$$\tilde{T} \rightarrow join(u_{\tilde{p}}, t_{\tilde{c}}) \tag{17}$$

$$\tilde{T} \xrightarrow{Text\ b\ binary\ conversion} B_y(\tilde{T})\,, \tag{18}$$

where $\tilde{T}$ is the obtained input by joining $u_{\tilde{p}}$ and $t_{\tilde{c}}$, and $B_y(\tilde{T})$ is the obtained byte array corresponding to $\tilde{T}$.

ii. Then, the byte array is grouped as 4-bit sets to obtain a relevant *rgba* value. For example, let us consider 00000610 as the byte array value. This byte array is grouped as $rgba_i\ (0{,}0{,}0{,}0)$ and $rgba_j\ (0{,}6{,}1{,}0)$. Here, the first row is allotted to the Red column, the second row is allotted to the Green one, the third one is allotted to the Blue column, and the last row is allotted to the Alpha column.

iii. Then, the obtained *rgba* values $rgba_i\ (0{,}0{,}0{,}0)$ and $rgba_j\ (0{,}6{,}1{,}0)$ are added by taking their modules of 255. The result is $rgba_R\ (0{,}6{,}1{,}0)$.

iv. If a set of *rgba* values includes more than 2 values, they are grouped two by two and the above process is repeated until obtaining a single *rgba* value. And also if the number of *rgba* values in a value set is an odd number, then the last pixel value is moved as the same value without doing any processing on it.

v. Finally, a corresponding hexadecimal value is computed for the resultant *rgba* value $rgba_R\ (0{,}6{,}1{,}0)$. Using private constant text with input text makes the algorithm secure for communication.

$$C2HA : u_{\tilde{p}} \rightarrow \hat{\mathrm{H}}(u_{\tilde{p}}) \tag{19}$$

After the user registration, the user requests the TCC to access the cloud which is dealt in under Intrusion Prevention Phase in subsection 3.3.

3.3 Intrusion Prevention Phase

Intrusion prevention is a structure of system security which is meant to protect data from the various types of threats. Due to the rapid growth of intrusions, it is necessary to control and protect the data access from unauthorized users in the cloud. For this purpose, an access control policy is proposed. It is designed to ensure that sensitive information cannot be accessed by the wrong person, while only the right person can access it.

3.3.1 Login Phase

Generally, login consists in a set of credentials that help to authenticate a user. If any user wants to access the data from the cloud, they give their unique ID and password to the cloud and send a login request to the TCC for data access.

$$user \xrightarrow{L\_REQ\{\ddot{u}_{ID},\, \ddot{u}_{\tilde{p}}\}} TCC\,, \tag{20}$$

where $L\_REQ\{\cdot\}$ represents the login request. After this user login request, the user authentication is performed.

3.3.2 User Authentication Phase

In the user authentication phase, the TCC verifies whether the user is a legitimate user or not. After receiving the login request from the user, the TCC compares the hash code already generated. If the hash code matches the TCC grants the permission for the user to access the cloud. The generated hash code is employed by cloud users only once at a time. After accessing data, the users generate a new password, and a new hash code is generated for every user, and it protects users' data from unauthorized users.

4. Results and Discussion

In this section, the outcomes of the proposed technique are discussed based on experimental evaluation. This section includes performance metrics and a comparative analysis with graphical plots. For performance evaluation, the proposed system uses the UNSW-NB15 dataset, which is collected via the IXIA PerfectStorm tool in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS). The UNSW-NB15 consists in a mixture of evidence on real normal data packets and synthetic contemporary attack instances in the form of numerous records of users; it includes observations on normal data packets and nine families of attacks.

4.1 Performance Analysis for Sgd-LSTM

Here, experiments are conducted on the UNSW-NB15 dataset to validate the performance of the proposed Sgd-LSTM classifier using 6 quality metrics, namely: precision, sensitivity, F-Score, accuracy, specificity, and training time. These metrics are measured based on four important parameters: true positive ($T(+ve)$), false positive ($F(+ve)$), true negative ($T(-ve)$), and false negative ($F(-ve)$) as follows:

Precision (M_1): It represents the fraction of data packets correctly recognized as intruder packets it concerns all packets, which are detected as intruder packets.

$$M_1 = \frac{T(+ve)}{T(+ve)+F(+ve)} \tag{21}$$

Sensitivity (M_2): It is the fraction of intruder packets correctly detected.

$$M_2 = \frac{T(+ve)}{T(+ve)+F(-ve)} \tag{22}$$

F-Score (M_3): It is the harmonic mean of the precision and the sensitivity.

$$M_3 = 2\times\frac{M_1 * M_2}{M_1+M_2} \tag{23}$$

Accuracy (M_4): It is the fraction of data packets that is correctly predicted.

$$M_4 = \frac{T(+ve)+F(-ve)}{T(+ve)+F(+ve)+T(-ve)+F(-ve)} \tag{24}$$

Specificity (M_5): The fraction of normal packets that is correctly predicted.

$$M_5 = \frac{T(-ve)}{F(+ve)+T(-ve)} \tag{25}$$

Training time (M_6): The time taken by the proposed algorithm for training the network.

Table 1 illustrates the performance of the proposed and existing classifiers in terms of precision, sensitivity, F-Score, accuracy, specificity, and training time. For an accurate prediction, the classifier should attain high precision, sensitivity, F-Score, accuracy, and specificity, and an efficient classifier would need a shorter training time. This performance comparison is graphically plotted in the following figures.

Table 1. Performance Comparison for the Proposed and Existing Classifiers

Techniques	Precision	Sensitivity	F-Score	Accuracy	Specificity	Training Time (in sec)
Proposed Sgd-LSTM	96.89	96.59	95.99	96.98	96.10	322
LSTM	94.55	94.95	94.39	94.58	91.02	416
ANN	93.62	93.54	93.43	93.12	90.99	423
KNN	92.85	92.56	92.86	92.63	90.05	459
ANFIS	91.55	91.23	92.03	91.56	89.36	472

Figure 4 compares the performance of the proposed classifier and existing classifiers such as Long Short-Term Memory (LSTM), Artificial Neural Network (ANN), K-Nearest Neighbour (KNN) and Adaptive Neuro-Fuzzy Inference System (ANFIS) in terms of precision. The graphical comparison shows that the proposed Sgd-LSTM attains the highest precision value of 96.89%. The precision value of existing LSTM, ANN, KNN, and ANFIS is 94.55%, 93.62%, 92.85%, and 91.55%, respectively. These results have revealed that the proposed classifier recognizes the intruder's packets accurately compared to existing classifiers.

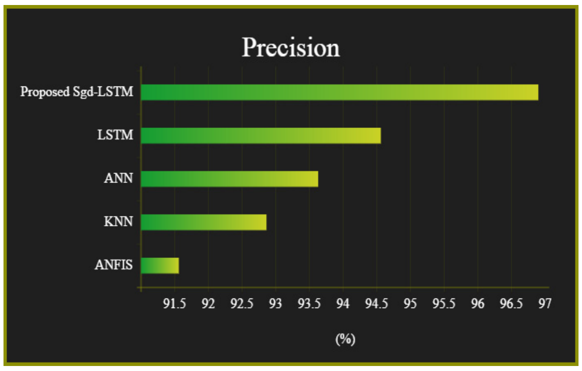


Figure 4. Precision Analysis

The effectiveness of the proposed and existing classifiers is also evaluated based on their attained sensitivity, which is shown in Figure 5. The sensitivity values of existing classifiers ANFIS, KNN, ANN, and LSTM is 91.23 %, 92.56 %, 93.54%, and 94.95% respectively. But the proposed classifier achieves a sensitivity of 96.59 % , which is more than 2% higher than that of the existing classifiers. This comparison proved that the proposed classifier is the most promising for data packet status identification as sensitivity is concerned.

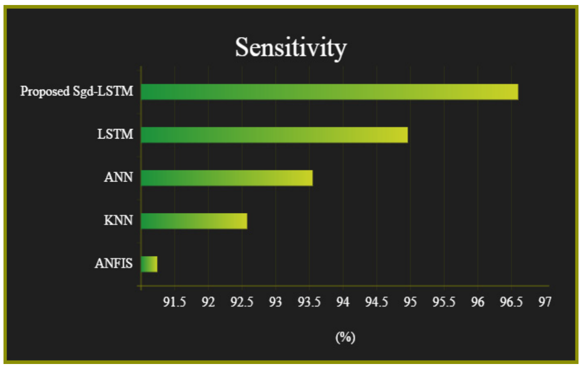


Figure 5. Sensitivity Analysis

Figure 6 illustrates the performance of the proposed and existing classifiers with respect to F-Score value. The F-score value for the proposed classifier is 95.99%, which is 1.60% higher than that of LSTM, 2.56% higher than that of ANN, 3.13% higher than that of KNN, and 3.96% higher than that of the ANFIS classifier. Based on these results, it is very clear that the proposed classifier attains a superior performance compared to other existing classifiers in terms of the F-Score, too.

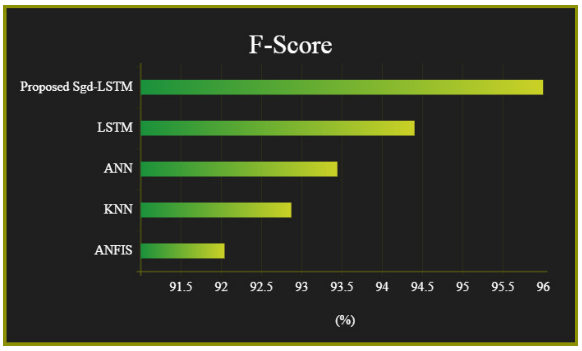


Figure 6. F-Score Analysis

Figure 7 illustrates the accuracy level of the proposed Sgd-LSTM which is weighted against the accuracy level of existing classifiers. The accuracy of the proposed classifier is 96.98%, whereas the accuracy level of the existing classifiers LSTM, ANN, KNN, and ANFIS is 94.58%, 93.12%, 92.63%, and 91.56%, respectively.

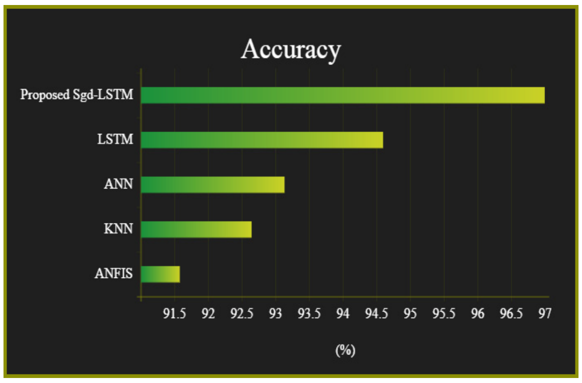


Figure 7. Accuracy Analysis

The specificity of the proposed and existing classifiers is illustrated in Figure 8 for performance comparison.

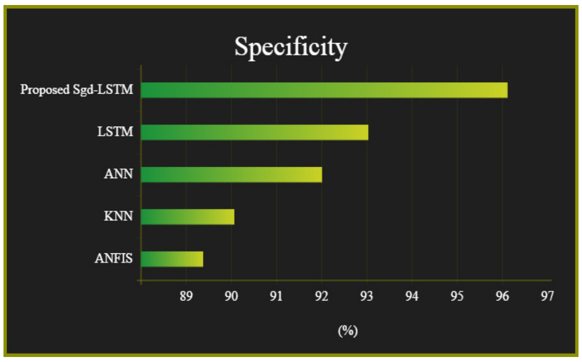


Figure 8. Specificity Analysis

The experimental results demonstrate that the proposed classifier is more promising than others with regard to specificity, too.

Figure 9 illustrates the efficiency level of the proposed and existing classifiers in terms of training time. For an efficient classification, the classifier should be faster and accurate. From all the experimental results, it is clear that the proposed classifier is highly efficient and accurate.

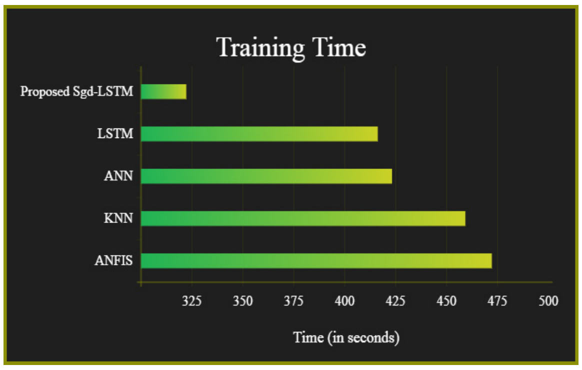


Figure 9. Training Time Analysis

4.2 Performance Analysis of C2HA

Here, the speed level of the proposed C2HA used in hash code generation is validated by the performance comparison for C2HA and the traditional MD5 and SHA512 algorithms in terms of hash code generation time. Figure 10 illustrates the hash code generation time for the proposed C2HA and traditional MD5 and SHA512 algorithms.

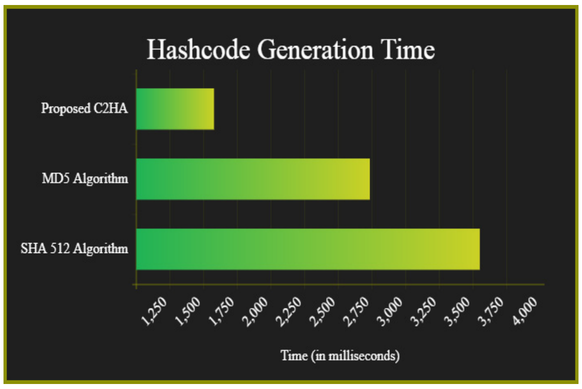


Figure 10. Hash code generation time Analysis

The time required for generating a hash code for a particular text or file is called hash code generation time. By comparing the proposed C2HA approach with the MD5 and SHA512 algorithms, it is clear that the proposed technique requires a shorter time for hash code generation. The hash code generation time for C2HA is 1574 ms, whereas the MD5 and SHA512 algorithms need 2731 ms and 3547 ms, respectively. This comparison proves that the C2HA is faster than the MD5 and SHA512 algorithms. The fitness values obtained by varying the number of iterations are illustrated in Table 2.

Table 2. Fitness Comparison

Techniques	Number of Iterations				
	5	10	15	20	25
Proposed EFCBO	63	79	85	97	110
BOA	54	61	69	82	95
ABC	45	55	62	76	87
CSO	36	48	56	70	79
WOA	31	44	53	62	70

On analyzing Table 2, it is obvious that the fitness level is efficiently increasing as the iteration count increases. The fitness level for the proposed EFCBO algorithm is 63 for 5 iterations, whereas the fitness level for an existing technique, such as Butterfly Optimization Algorithm, Artificial Bee Colony, Cuckoo Search Optimization, and Whale Optimization algorithm is 54, 45, 36, and 31, respectively. Similarly, for all the remaining iteration counts, the proposed EFCBO algorithm obtained the highest fitness level.

5. Conclusion

The security issue in the cloud environment is one of the major barriers to cloud implementation. Various attacks take advantage of the network and protocol susceptibility to damage the cloud users' data and applications. To detect such attacks and protect the cloud user's data from various intrusions, this paper proposed a Sgd-LSTM and signature-based access control policy based intrusion detection and prevention system in the cloud. For the purpose of performance evaluation, the proposed Sgd-LSTM, EFCBO, and C2HA techniques are weighted against several existing techniques. The obtained results proved that the proposed technique is highly efficient and more secure and accurate in comparison with other techniques. In the future, the proposed work could be extended by integrating the cryptography algorithm with the proposed system to prevent data transmission attacks in the cloud environment.

REFERENCES

Ali, A. & Yousaf, M. M. (2020). Novel three-tier intrusion detection and prevention system in software defined network, *IEEE Access*, *8*, 109662-109676. DOI: 10.1109/ACCESS.2020.3002333

Chkirbene, Z., Erbad, A., Hamila, R., Mohamed, A., Guizani, M. & Hamdi, M. (2020) TIDCS: A dynamic intrusion detection and classification system based feature selection, *IEEE Access*, *8*, 95864-95877. DOI: 10.1109/ACCESS.2020.2994931

Dey, S., Ye, Q. & Sampalli, S. (2019). A machine learning based intrusion detection scheme for data fusion in mobile clouds involving heterogeneous client networks, *Information Fusion*, *49*, 205-215. DOI: 10.1016/j.inffus.2019.01.002

Feng, J., Yang, L. T., Dai, G., Wang, W. & Zou, D. (2019). A Secure High-Order Lanczos-Based Orthogonal Tensor SVD for Big Data Reduction in Cloud Environment, *IEEE Transactions on Big Data*, *5*(3), 355-367. DOI: 10.1109/TBDATA.2018.2803841

Hafeez, I., Antikainen, M., Ding, A. Y. & Tarkoma, S. (2020). IoT-KEEPER: Detecting malicious IoT network activity using online traffic analysis at the edge, *IEEE Transactions on Network and Service Management*, *17*(1), 45-59. DOI: 10.1109/TNSM.2020.2966951

Hajimirzaei, B. & Navimipour, N. J. (2019). Intrusion detection for cloud computing using neural networks and artificial bee colony optimization algorithm, *ICT Express*, *5*(1), 56-59.

He, D., Kumar, N., Khan, M. K., Wang, L. & Shen, J. (2016). Efficient privacy-aware authentication scheme for mobile cloud computing services, *IEEE Systems Journal*, *12*(2), 1621-1631. DOI: 10.1109/JSYST.2016.2633809

Liu, J., Yu, J. & Shen, S. (2018). Energy-Efficient Two-Layer Cooperative Defense Scheme to Secure Sensor-Clouds, *IEEE Transactions on Information Forensics and Security*, *13*(2), 408-420.

Mauro, D. M, Galatro, G. & Liotta, A. (2020). Experimental Review of Neural-Based Approaches for Network Intrusion Management, *IEEE Transactions on Network and Service Management*, *17*(4), 2480-2495.

Mishra, N. & Pandya, S. (2021). Internet of Things Applications, Security Challenges, Attacks, Intrusion Detection, and Future Visions: A Systematic Review, *IEEE Access*, *9*, 59353-59377. DOI: 10.1109/ACCESS.2021.3073408

Mishra, P., Varadharajan, V., Pilli, E. S. & Tupakula, U. (2020). VMGuard: A VMI-Based Security Architecture for Intrusion Detection in Cloud Environment, *IEEE Transactions on Cloud Computing*, *8*(3), 957-971. DOI: 10.1109/TCC.2018.2829202

Mohana Prabha, K. & Vidhya Saraswathi, P. (2020). Suppressed K-Anonymity Multi-Factor Authentication Based Schmidt-Samoa Cryptography for privacy preserved data access in cloud computing, *Computer Communications*, *158*, 85-94. DOI: 10.1016/j.comcom.2020.04.057

Saxon, J., Bordbar, B. & Harrison, K. (2015). Introspecting for RSA Key Material to Assist Intrusion Detection, *IEEE Cloud Computing*, *2*(5), 30-38. DOI: 10.1109/MCC.2015.100.

Traore, I., Woungang, I., Nakkabi, Y., Obaidat, M. S., Ahmed, A. A. E. & Khalilian, B. (2012). Dynamic Sample Size Detection in Learning Command Line Sequence for Continuous Authentication, *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *42*(5), 1343-1356. DOI: 10.1109/TSMCB.2012.2191403

Wahab, O. A., Bentahar, J., Otrok, H. & Mourad, A. (2021). Resource-Aware Detection and Defense System against Multi-Type Attacks in the Cloud: Repeated Bayesian Stackelberg Game, *IEEE Transactions on Dependable and Secure Computing*, *18*(2), 605-622. DOI: 10.1109/TDSC.2019.2907946

Xie, Y., Feng, D., Hu, Y., Li, Y., Sample, S. & Long, D. (2020). Pagoda: A Hybrid Approach to Enable Efficient Real-Time Provenance Based Intrusion Detection in Big Data Environments, *IEEE Transactions on Dependable and Secure Computing*, *17*(6), 1283-1296. DOI: 10.1109/TDSC.2018.2867595