Effective Routing Attack Detection Analysis in MANET/WSN using a Deep Learning Framework Along with False AES as a Secure Layer

[1]Srinivas Jhade, [2]Premalatha V, [3]T.S. Karthik, [4]V.Pandimurugan, [5]Bejoy.B.J, [6]S.Prabakeran

[1]Associate Professor, Dept. of CSE, KG Reddy College of Engineering and Technology, srjhade@gmail.com
[2]Assistant Professor, Dept. of CSE, Koneru lakshmaiah Education and foundation, premawilliams@gmail.com
[3]Professor, Dept. of ECE, Aditya college of Engineering and Technology, writetotsk@gmail.com
[4]Assistant Professor, Dept of Networking and communications SRMIST, Kattankulathur Campus, Chennai.
[5]Assistant Professor, Dept of CSE, CHRIST(Deemed to be University) Kengeri Campus.
[6]Dept.of Networking and Communications, SRM Institute of Science& Technology (SRMIST)

Abstract

Mobile Ad-Hoc Networks (MANET) are becoming extremely popular as a result of their potential to offer inexpensive solutions to practical communication issues. MANETs are more vulnerable to security attacks due to their characteristics, including node mobility, lack of centralized control and limited bandwidth. Traditional cryptography techniques cannot entirely protect MANETs from new threats and weaknesses to address these security challenges. So, this paper brings an effective integration of cryptography with revolutionized deep learning technology to effectively detect the routing attacks in which following are the stages. a) Data Collection from popular repositories like AWID, DARPA, and UNSW-NB15 containing possible routing attacks in network and b) Preprocessing these with techniques like missing data, redundant data, noisy data and data cleansing c) feature extraction using autoencoder d) feature selection using Particle Swarm Optimization and finally e) classification using LSTM. Also, to secure the whole network along with security features of LST use False Advance Encryption Standard (Faa ES) as a cryptographic method as well. Experimental results were conducted over various state-of-art models under various measures (accuracy:0.97, precision:0.89, detection rate: 0.94).

Keywords: MANET , Particle Swarm Optimization, False Advance Encryption Standard , LST

1 INTRODUCTION

MANETs (mobile ad hoc networks) consist of self-organizing nodes connected wirelessly. Ad hoc networks function as routers where packets are transmitted from node to node. A remote ad hoc network is an enormous and commonly used network. The mobile network management node is not centrally managed since movable nodes are self-managed. It is up to the mobile nodes to decide where to go based on their needs. Assists nodes in joining or leaving the network [1]. Communication between nodes is not limited. During the establishment of the relationship, the nodes can lose data if they are outside the network's radio range. Scientific and military fields use MANET, including rescue operations. In addition to improved connectivity across networks, cyberattacks are also on the rise [2]. Due to their unstable operating environment, limited mobility of resources, and rapidly changing device topologies, wireless ad hoc mobile networks are vulnerable to many security threats.

It is acceptable for everyday actions of a system to interfere with the detection of irregularities. Enumerating standard output is difficult due to the variability of system activity [4]. Anomalies are discovered when attacks are new or unexplained with a high rate of false positives. Rather than identifying new attacks, it recognizes proven attacks. Because MANETs lack fixed infrastructure and have complex topologies, safe connectivity is challenging. Cryptography and access management hold up the balance by detecting intrusions. In response to an attack that has occurred or is currently underway, it is displayed automatically as a root of warning. As shown in Figure 1, deep learning follows a generic flow for attack detection.

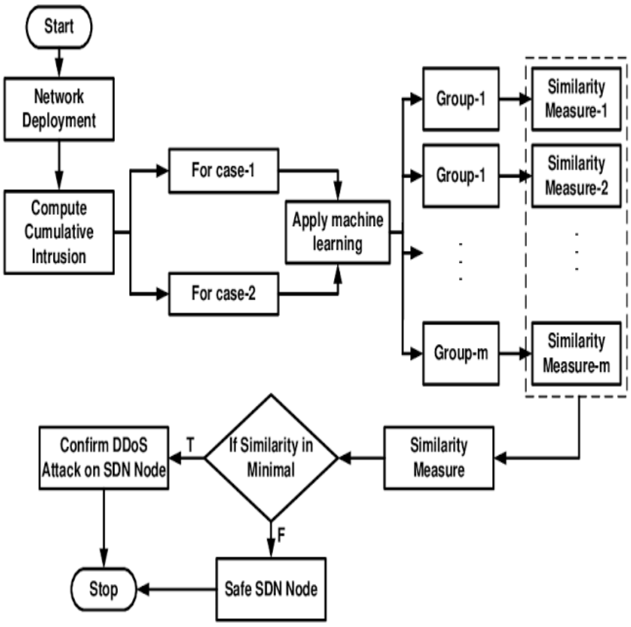


Figure 1. Generic flow for detecting attacks in MANET/WSN

ML was one aspect of artificial intelligence that was created in the late 1950s. The field has evolved into machine-based algorithms that are effective in addressing a variety of medical, engineering, and computer science issues such as sorting, clustering, regression, optimization, and medical image processing [9–14]. Without the involvement of humans, ML architectures dynamically learn and act. It manipulates complex data automatically, appropriately, and effectively to create a model. ML can gain from a generalized structure to have an all-encompassing strategy for enhancing device performance. Many scientific domains can benefit from its use, including manual data entry, spam detection, medical diagnosis, image identification, data cleansing, and noise reduction, among others [9, 17]. In recent years, ML has been used to solve many problems associated with WSNs and MANETs. When ML is used in WSNs/MANETs, complex issues such as reprogramming and manually accessing enormous amounts of data can be avoided. ML techniques frequently result in useful data production and large-scale data collection [17]. Applications of ML techniques for classification and identification are numerous.

1.1 Key Highlights

This paper focuses on bringing an effective deep learning framework for MANET. Following are the objectives of this study:

- Develop an efficient deep learning framework for routing attacks in MANET/WSN.
- Bring a double security layer with help of the LSTM secure inner layer feature and FAES cryptography.
- Boost the performance of classification of attacks using feature extraction and selection procedure.
- The proposed strategy works better under many measures, according to experimental data.

Organization of paper: Since a summary of MANET/WSN attacks was presented in part 1, the remainder of the paper is as follows: Literature review is presented in Part 2, the methodology is presented in Part 3 and Section 4 presents the performance analysis. Part 5 winds the study with the conclusion.

2 RELATED WORKS

Using suggestion filtering and security monitoring, Ponguala and Rao (2019) [18] offered a novel group-based routing system. An unsupervised machine learning algorithm is modified to filter recommendations in the network. Secure Certificate-based Group Formation (SCGF) groups the entire network first. Each group uses the Recommendation Filtering by K-Means algorithm (RF-K means) to compute trust. A hybrid optimization technique called GA-FFA, which combines the FireFly and Genetic Algorithms, is presented for selecting safe and efficient routes. HMAC-AES is an innovative algorithm that combines cryptography and hash functions to secure data transfers.

Real-time attack detection and blocking method based on machine learning were proposed by Sebopelo et al. (2009). A Logistic Regression model (LR) and support vector machine model (SVM) were used to test the prediction model based on the Iris data set. A logistic regression model (LR) and

Support Vector Machine model (SVM) was used to test the prediction model based on the Iris data set. The detection of malicious assaults in MANETs is thus better served by LR.

Srinivasan et al. (2019) [20] emphasized the network's vulnerability to a black hole attack. To recognize such attacks, the Honeypot Agent-based detection Scheme (HPAS) with Long-Short Term Memory (LSTM) has been investigated. By sending Route Request packets (RREQ), honeypots attract and capture black hole attackers. This method is called HPAS-LSTM. It has been demonstrated that the recommended detection method exists using extensive model results obtained using the NS-2 simulator.

Majumder & Bhattacharya (2019) [21] presented an extensive review of different techniques to detect and avoid various kinds of attacks in Mobile ad hoc networks. This paper also gives a brief idea about different routing algorithms used in MANETs. Proactive and Reactive routing algorithms are explained well. Performance parameters like average delay, throughput, the average number of hops per route etc. are also discussed in the context of security attacks.

In 2020, Laqtib et al. proposed improving the performance of intrusion detection systems [22]. The NSL-KDD dataset consists of intrusion information and regular network connections that were compared to determine which deep learning model should be used in MANET using Inception architecture convolutional neural networks (Inception-CNN), Bidirectional long short-term memory (BLSTM) and deep belief networks (DBN).

3 METHODOLOGY

Figure 2 depicts the overall architecture analysis of the proposed framework in which the following are the stages. For any deep learning model, feeding enough dataset to understand the model. a) Data sources are gathered from well-known repositories like AWID, DARPA, and UNSW-NB15. DARPA contains DDoS, privilege escalation (remote-to-local and user-to-root) and probing, while UNSW-NB15 contains backdoors, DoS, exploits, fuzzes, generic, and port scans. Once these networks are collected, they will be handed over to the appropriate agencies b) Preprocessing to eliminate missing data, redundant data, noisy data and data cleansing. Once these have been preprocessed, they were passed to c) Feature extraction procedure where quintessential network feature is been extracted using Autoencoder, convolutional neural network variant. Then feature selections are done using Particle Swarm Optimization (PSO). Once these have been done, they are finally passed for classifier where e) LSTM is been used for classifying various attacks. Finally, these are secured using the FAES cryptography framework.

3.1 Data Source

So, here in this section, we took 3 popular datasets such as AWID, DARPA, and UNSW-NB15 and each of them is described below.

AWID: AWID is a set of publicly accessible data4 that focuses on 802.11 networks. Its developers recorded WLAN traffic in a packet-based format using a modest network environment (11 clients). There were 37 million packets seized in one hour. Each packet has 156 attributes that were extracted. Executing 16 unique attacks on the 802.11 network resulted in malicious network traffic. A training subset of AWID and a test subset were labelled. Table 1 lists the characteristics of the AWID dataset [23,24].

DARPA [25,26]: In an emulated network environment, MIT Lincoln Lab produced the most famous intrusion detection data sets based on DARPA 1998/99 data sets. A variety of attacks such as DoS, buffer overflows, port scans, and rootkits are included in DARPA 1998 and 1999 data sets, respectively. The website14 contains download links and further information. It is often criticized that the data sets are overly redundant or contain artificial attacks. Table 2 lists all attacks made against the dataset.

UNSW-NB15 [27,28]: In a simulated environment, data was collected from the UNSW-NB15 dataset using the IXIA Perfect Storm tool. It consists of both safe and harmful network traffic. Worms, backdoors, denial-of-service attacks, exploits and fuzzers are a few of the nine different attack types. There are data sets with additional attributes accessible in a flow-based format as well. Splits for the training and exam were already included in the UNSW-NB15. There are 45 different IP addresses in the publicly available data collection. Table 3 displays the general traits and a description of the dataset.

3.2 Preprocessing

In order to eliminate the noisy, missing, and inconsistent data included in the dataset, pre-processing of the data becomes essential. Pre-processing is necessary since the quality of the data degrades as a result of the dataset's records being gathered from various and diverse sources. The quality of the data is impacted by many means. Accuracy, thoroughness, consistency, timeliness, plausibility, and interpretability are among these criteria.

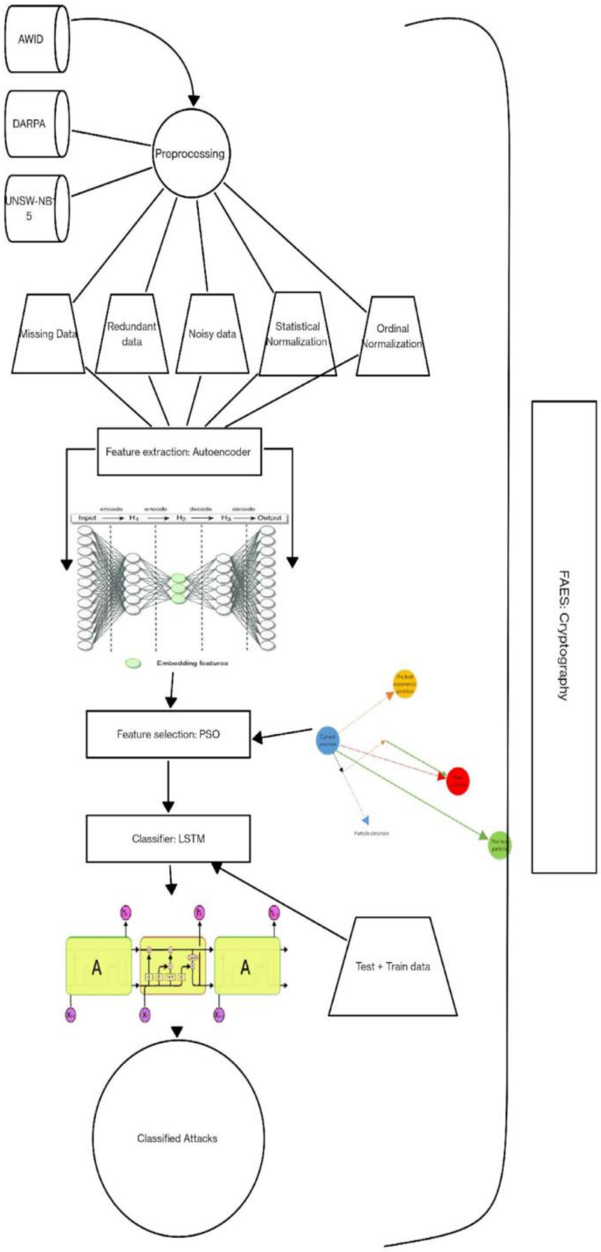
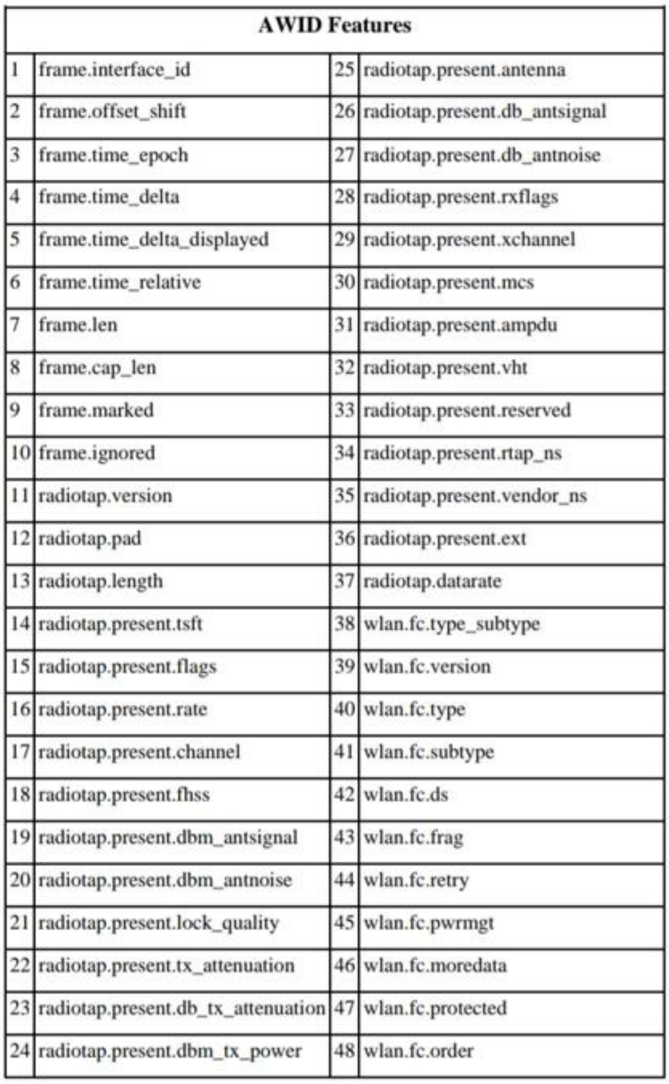


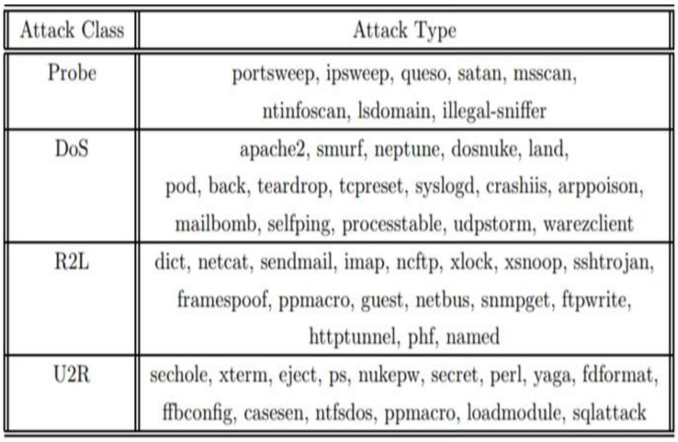
Figure 2. The overall architecture of the proposed framework

Table 1. Overall features in the AWID dataset



AWID Features			
1	frame.interface_id	25	radiotap.present.antenna
2	frame.offset_shift	26	radiotap.present.db_antsignal
3	frame.time_epoch	27	radiotap.present.db_antnoise
4	frame.time_delta	28	radiotap.present.rxflags
5	frame.time_delta_displayed	29	radiotap.present.xchannel
6	frame.time_relative	30	radiotap.present.mcs
7	frame.len	31	radiotap.present.ampdu
8	frame.cap_len	32	radiotap.present.vht
9	frame.marked	33	radiotap.present.reserved
10	frame.ignored	34	radiotap.present.rtap_ns
11	radiotap.version	35	radiotap.present.vendor_ns
12	radiotap.pad	36	radiotap.present.ext
13	radiotap.length	37	radiotap.datarate
14	radiotap.present.tsft	38	wlan.fc.type_subtype
15	radiotap.present.flags	39	wlan.fc.version
16	radiotap.present.rate	40	wlan.fc.type
17	radiotap.present.channel	41	wlan.fc.subtype
18	radiotap.present.fhss	42	wlan.fc.ds
19	radiotap.present.dbm_antsignal	43	wlan.fc.frag
20	radiotap.present.dbm_antnoise	44	wlan.fc.retry
21	radiotap.present.lock_quality	45	wlan.fc.pwrmgt
22	radiotap.present.tx_attenuation	46	wlan.fc.moredata
23	radiotap.present.db_tx_attenuation	47	wlan.fc.protected
24	radiotap.present.dbm_tx_power	48	wlan.fc.order

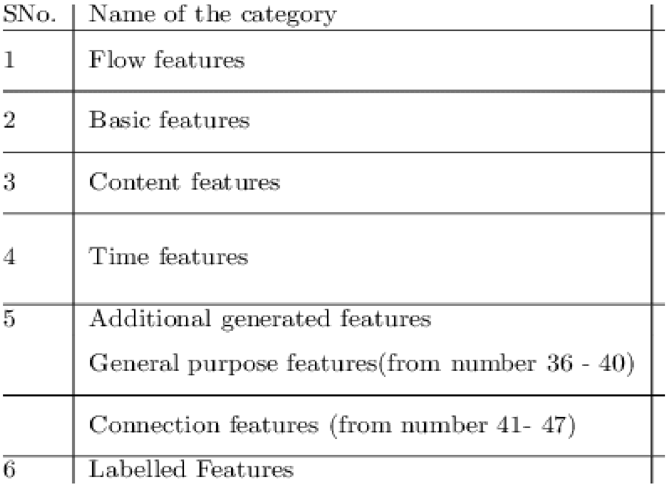
Table 2. Overall attack types of DARPA



Attack Class	Attack Type
Probe	portsweep, ipsweep, queso, satan, msscan, ntinfoscan, lsdomain, illegal-sniffer
DoS	apache2, smurf, neptune, dosnuke, land, pod, back, teardrop, tcpreset, syslogd, crashiis, arppoison, mailbomb, selfping, processtable, udpstorm, warezclient
R2L	dict, netcat, sendmail, imap, ncftp, xlock, xsnoop, sshtrojan, framespoof, ppmacro, guest, netbus, snmpget, ftpwrite, httptunnel, phf, named
U2R	sechole, xterm, eject, ps, nukepw, secret, perl, yaga, fdformat, ffbconfig, casesen, ntfsdos, ppmacro, loadmodule, sqlattack

Data might be wrong for a variety of reasons. It might be the result of mistakes made by either humans or computers when entering the data. Additionally, mistakes could be made while transmitting data. Duplicate records can exist and need to be removed. Different types of incomplete data exist. It might be because the data wasn't available when the entry was made. Alternately, it may be because the apparatus used to collect the data malfunctioned. Additionally, the information may be missed or misinterpreted [29].

Table 3. depicts the overall features of UNSW-NB15.



SNo.	Name of the category
1	Flow features
2	Basic features
3	Content features
4	Time features
5	Additional generated features General purpose features(from number 36 - 40)
	Connection features (from number 41- 47)
6	Labelled Features

3.2.1 Missing data

Due to the various data mining scenarios that occurred while collecting the data, data can be missing. The absence of the event, the absence of the data itself, or the inapplicability of the field in the context of the experiment are all examples of missing data. To deal with missing data, the following policies have been implemented: a) Ignore the records with missing values. When the class label is absent, this is possible. The standard records in this instance lacked values for the class label, which was filled in by matching the most pertinent attribute values to the records lacking the class label. b) Manually filling in values based on the obtained domain knowledge. This can take more time and possibly become impractical for rather large datasets. Using the RemoveWithValuesFilter option in the "WEKA (Waikato Environment for Knowledge Analysis)" tool, missing data that were unavailable and could not be inferred by matching with the most pertinent ones were eliminated.

3.2.2 Redundant Data

For a given record instance, these databases contain numerous redundant records. Due to the difficulty the learning algorithm may have in determining whether an attack has been detected or not, this may lead to bias during the actual detection of intrusion. The records that are few in nature are rather destructive for attack detections like U2R and R2L where the hackers try to send a few packets for accessing the system either directly or remotely. The unsupervised RemoveDuplicatesfilter developed by WEKA has been used to eliminate duplicate records [30].

3.2.3 Noisy Data

The statistical methods using measures of central tendency can be used to handle noisy data, which are random errors or variances. The noisy or nonsensical data, which frequently contribute to extreme values and outliers were analyzed using the WEKA tool. The extreme and outlier values were located using the InterQuartileRange filter.

3.2.4 Statistical Normalization

Normal distributions with means of 0 and variances of 1 are normalized using statistical normalization. There is a definition of statistical normalization [31].

$$x_i = \frac{\vartheta_i - \mu}{\sigma} \qquad (1)$$

where ⍰ is mean of n values for a given attribute ⍰=$\frac{1}{n}\sum_{i=1}^{n}$ v_i, σ *is its stand deviation*

$$⍰ = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (v_i - \mu)} \qquad (2)$$

The data set should follow a normal distribution, however, and according to the central limit theorem, there should be a high number of samples. The attribute's value is not scaled into the [0,1] range using statistical normalization. Instead, it spans [-3,3] for 99.9% of the attribute samples.

3.2.5 Ordinal Normalization

Ranking the continuous value of an attribute before normalizing the rank into [0,1] is known as ordinal normalization. Ordinal normalization, defined by letting r be the rank of a particular value in an attribute [32].

$$x_i = \frac{r-1}{max(r)-1} \qquad (3)$$

Ordinal normalization, obviously, also ranges the values of an attribute into [0,1]. If any attribute values in this study are the same, the rank is not raised. For example, if several numbers are rated as,15,15,15, then 16 other than 18 is the following rank.

3.3 Feature extraction

Unsupervised machine learning algorithms use the autoencoder of artificial neural networks. Autoencoders are taught to recreate output that closely resembles the original input. The three layers of an autoencoder are the input layer, the output layer, and the hidden layer, which is smaller than the input layer. Figure 3 illustrates a straightforward autoencoder. Autoencoders reveal data structures more effectively than Principle Component Analysis (PCA) by reducing data via non-linear transformations. With the input value serving as the goal value, the procedure employs backpropagation, which is based on the encoder-decoder paradigm. Data input is encoded first, then expanded to restore the original data (decoder). The layer transfers its output to the layer below after instructing it to construct a highly nonlinear dependence model on the input. The data input size is reduced using this technique. The classification process makes use of an extracted feature from the middle encoded layer of the autoencoder.

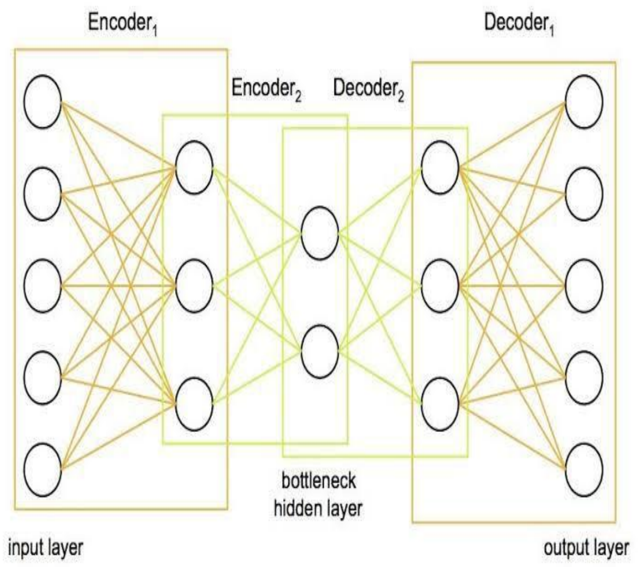


Figure 3. Overall Autoencoder framework for attack detection

The encoder equation for the neural network's single hidden layer is in equation (4), and the decoder equation is in equation (5).

$$Y \ = \ f_\theta(X) \ = \ s(WX + b_x) \qquad (4)$$

$$X \ = \ f_\theta(X) \ = \ s(WX + b_x) \qquad (5)$$

f(X) is the encoding function, while g(Y) is the decoded function. W (weighting) and b (bias) are parameters for the data x. Reconstructing X' from the hidden representation of Y are performed using the decoder function g.

For the X dataset, the objective function is given as follows, and the parameters = (W, bx, by) are determined by autoencoder training:

$$⍰ \ = \ min\, L(X, X') = min\, L(X, g\big(\big(f(X)\big) \quad (6)$$

Loss Reconstruction (L1) is derived from squares error for linear reconstruction:

$$L_1(\theta) \ = \sum_{i=1}^{n} \quad ||x_i - x_i'||\ ^2 \qquad (7)$$

Regarding nonlinear functions, cross-entropy is used to determine reconstruction loss (L2).

$$L_2(\theta) \ = -\sum_{i=1}^{n} \quad [x_i(log(y_i) + (1 - x_i)log(1 - y_i)] \quad (8)$$

3.5 Feature Selection

The purpose of feature selection is to determine the minimum subsets of features needed by the classifier to classify an activity or connection as normal or intrusive. There are always fewer features in the original feature set m than in the feature subset p. Figure 4 illustrates how to select a feature subset.

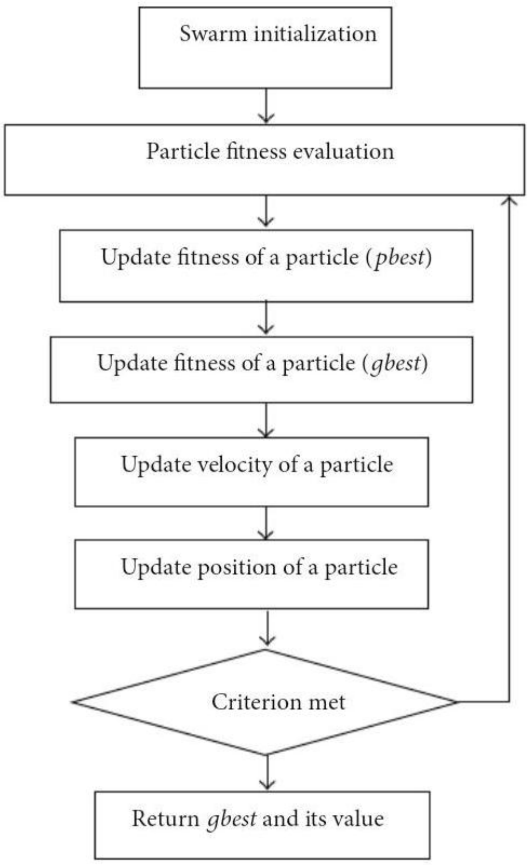

Swarm initialization
Particle fitness evaluation
Update fitness of a particle (pbest)
Update fitness of a particle (gbest)
Update velocity of a particle
Update position of a particle
Criterion met
Return gbest and its value

Figure 4. The overall flow of PSO for attack detection

Eberhat and Kennedy created the population-based method known as the PSO[22]. PSO is an effective and highly regarded worldwide search methodology. It is a suitable technique to tackle feature selection issues because of its straightforward feature encoding, capacity for global searches, reasonable processing requirements, absence of parameters, and simplicity of implementation [35,36]. The PSO is used to choose features for the preceding reasons. The search area in which a subset of key traits or components was examined and picked using PSO is known as the primary space. In PSO, a population, or swarm, is formed by the particles, which stand in for potential solutions in the search space. By spreading 1s and 0s at random, the particle swarm is created. The principal component with a value of 1 for each particle is chosen, while the principal component with a value of 0 is disregarded. A different subset of the primary components is thus shown by each particle. The particle swarm initially disperses at random and then moves in the search space or primary space by updating its position and velocity to find the optimal subset of features. The expressions in (7) and (8) are for particle i's current position and velocity (8).

$$x_i = \{x_{i1}, x_{i2} \ldots\ldots\ldots\ldots\ldots x_{iD}\} \qquad (9)$$

where D denotes the major search dimension.

$$\square_i = \{v_{i1}, v_{i2} \ldots\ldots\ldots\ldots\ldots v_{iD}\} \qquad (10)$$

The following formula is used to determine the particle's position and speed.

$$\square_{id}^{t+1} = \omega * v_{id}^{t} + c_1 * (p_{id} - x_{id}^{t} + c_2 * r_{2i} * (P_{gd} - x_{id}^{t}),$$

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \qquad (11)$$

where d represents the search space's dth dimension and t represents the process's tth iteration. c1 and c2 are acceleration constants, while W is the inertia weight. The r1i and r2i random variables have uniform distributions in the range [0, 1]. Pid and Pgd in the dth dimension stand in for the elements of pbest and Gbest. The position and velocity data of each particle are updated continuously in an attempt to identify the greatest possible combination of features, up until a stopping criterion is satisfied, which might be a maximum number of iterations or an appropriate fitness value.

3.6 Classifier

Hochreiter and Schmidhuber [37,38] introduced LSTM as an extension of RNN in 1997, in contrast to RNN, which is unable to retain data for long periods. Long-term reliance is a problem that LSTM was designed to solve. While the LSTM design is more sophisticated and has four hidden layers, the RNN architecture's hidden layers are simpler (e.g., a single tanh layer) (Figure 5).

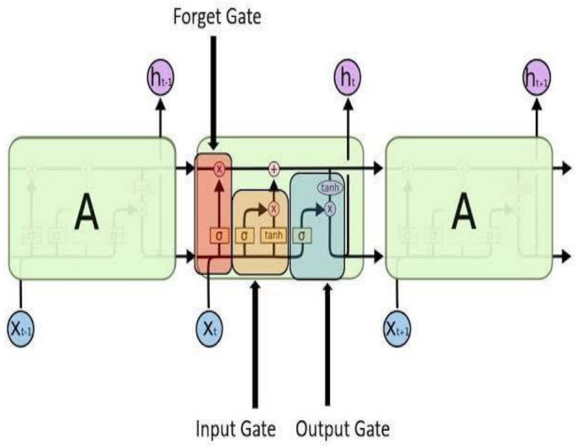


Figure 5. LSTM neural network framework

The key component of an LSTM is the cell state. The sigmoid function is employed by the gates to add or remove information, protecting the state of the cell (one means allows the modification, while a value of zero means denies the modification). There are three distinct gates.

To get the gate layer: The LSTM analyzes the input data and the data received from the previously buried layer to determine which information it will remove from the cell state. It then employs a sigmoid function to make this determination (One means keeps it, 0 means delete it). It is determined as:

$$f_t = \sigma\left(W_i.[h_{t-1}, x_t] + b_f\right) \qquad (12)$$

Which information the LSTM will store in the cell state at the input/update gate layer is decided. Following the input gate layer's determination of which information will be updated using a sigmoid function, the Tan h layer then proposes a new vector to be added to the cell state. The data we decided to disregard will then be deleted, and the new vector values will be substituted, updating the cell state in the LSTM. Here's the calculation:

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \; and$$
$$c_t = tanh(W_c.[h_{t-1}, x_t] + b_c \quad (13)$$

A sigmoid function is used by the output layer to decide which part of the LSTM cell will be output to determine what will be our output. The output is then transferred to the next neuron using a Tanh layer, which outputs only the data we chose to send (value between 1 and 1). Here's the calculation:

$$O_t = \sigma(W_0[h_{t-1}, x_t] + b_0 \text{ and}$$
$$h_t = O_t * tanh(C_t) \qquad (14)$$

3.7 Security using cryptography

The data stored in an array must undergo many changes that are specified by the AES encryption method. The data are first organized into an array as part of the cipher's first phase, following which the cipher modifications are repeated through many encryption rounds. As the first transformation of the AES encryption process, data is replaced using a replacement table. The second transformation involves moving data rows. In the third, columns are mixed. Each column is subjected to the final transformation, which utilizes a unique portion of the encryption key. More rounds are necessary to accomplish longer keys. These are used to create FAES, a more sophisticated variant of AES.

The data masking process is the fundamental tenet of the False-AES algorithm. Each iteration of AES masks the input plaintext with a different piece of random data. The genuine cipher data at the output, which is shown in Figure 6 [5], must be produced when encryption is complete with the random mask removed.

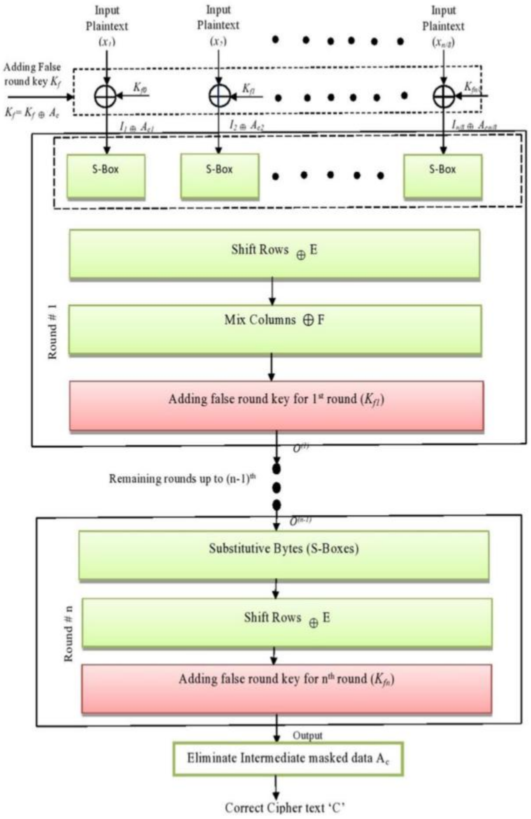

Input Plaintext (x_1)
Input Plaintext (x_2)
Input Plaintext ($x_{n/8}$)
Adding False round key K_f
$K_f = K_f \oplus A_c$
K_{f0}
K_{f1}
K_{fn}
$I_1 \oplus A_{c1}$
$I_2 \oplus A_{c2}$
$I_{n/8} \oplus A_{cn/8}$
S-Box
S-Box
S-Box
Shift Rows ⊕ E
Mix Columns ⊕ F
Adding false round key for 1st round (K_{f1})
Round # 1
$O^{(1)}$
Remaining rounds up to (n-1)th
$O^{(n-1)}$
Substitutive Bytes (S-Boxes)
Shift Rows ⊕ E
Adding false round key for nth round (K_{fn})
Round # n
Output
Eliminate Intermediate masked data A_c
Correct Cipher text 'C'

Figure 6. False AES overall encryption - decryption flow

The architecture of the False-AES algorithm includes the correct round key Kci=(Kc0, Kc1, Kc2,..., Kcn/8)added with the constant intermediate data Ac=(Ac1, Ac2, Ac3,..., Acn/8) for the generation of the false keys Kf=Kci ⊕ Ac=(Kci1 ⊕ Aci1, Kci2 ⊕ Aci2,..., Kcin/8 ⊕ Acin/8). The process of False-AES [5]. The 1stencryption round is followed with the input plaintext x=(x1,x2,x3,...,xn/8), the jth S-box input data where (j=1,2,3,...., n/8) includes xj ⊕ Kco,j ⊕ Ac, j=Ij ⊕ Acj where Ij says standard encryption data and Ac,j stand as the equivalent mask (I=(I1, I2,I3,..., In/8)=x ⊕ Kc0). To eradicate the Ac mask from the encrypted data, reconstructed S-boxes of AES are used. Thus, the generated j thS-box output of 1st encryption round turns into S-box (Ij) ⊕ S-box (Ac, j). The false round keys (Kf0, Kf1, Kf2,..., Kfn) are also applied in the remaining rounds of encryption to mask the outflow of correct round keys Kc1, Kc2, Kc3,..., Kcn.

In the last step of the procedure, the mask AMcis removed, and the right cipher text data is retrieved. The False-AES algorithm's output data for the zth bit is shown as follows:

$$C_z \otimes A^m_{c,z} \qquad (15)$$

The correct encryption text output is retrieved as soon as the intermediate mask component AMC, z has been removed.

$$C_z(C_z \otimes A^m_{c,z} \otimes A^m_{c,z} = C_z \qquad (16)$$

4 PERFORMANCE ANALYSIS

The proposed model is implemented over hardware specifications like Ryzen 5/6 series CPU, 1 TB HDD, NV GTX, windows 10 OS and software specifications like PyTorch, an open source python library for building deep learning models and Google Collaboratory, an open source Google environment for building deep learning model. The experimental evaluation is taken over various other models like CNN, VGG16, VGG19 and GAN over various measures like accuracy, sensitivity, specificity, recall, precision, F1-score, detection rate, TPR and FPR. Table 4 depicts the overall analysis under accuracy, sensitivity, and specificity. Figure 7(a,b,c) depicts the comparative analysis of various models over 3 datasets.

Table 4. Overall analysis under accuracy, sensitivity, specificity

Models	Dataset	Accuracy	Sensitivity	Specificity
CNN		94	97	96
VGG16		81	88	90
VGG19	AWID	83	90	93
GAN		86	92	96
LSTM		97	98	98

CNN		92	97.3	96.3
VGG16	DARPA	82	88.5	90.7
VGG19		85	90.4	93.6
GAN		82	93	96.9
LSTM		96	98.3	98.2
CNN		94	97.8	96.7
VGG16	UNSW-NB15	80	89.3	91.2
VGG19		86	91.7	94.2
GAN		84	93.5	97
LSTM		97	98.8	98.5

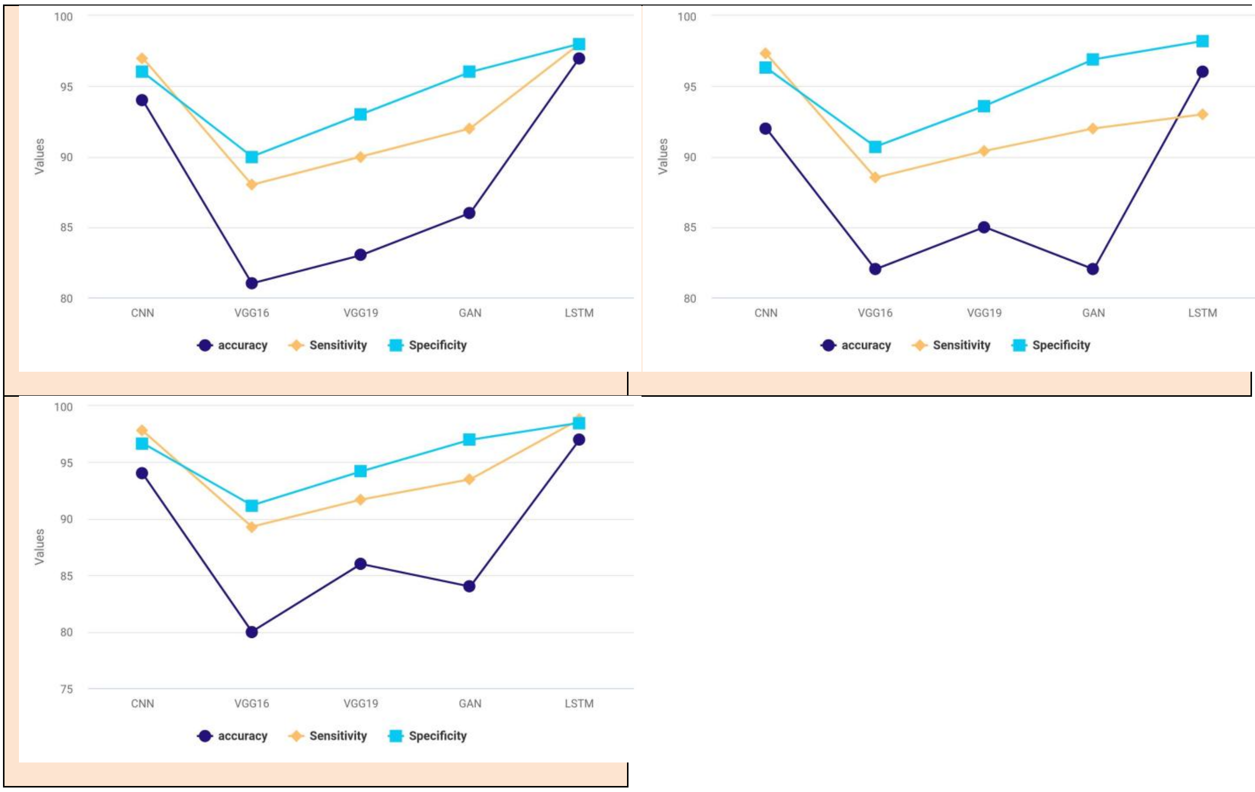
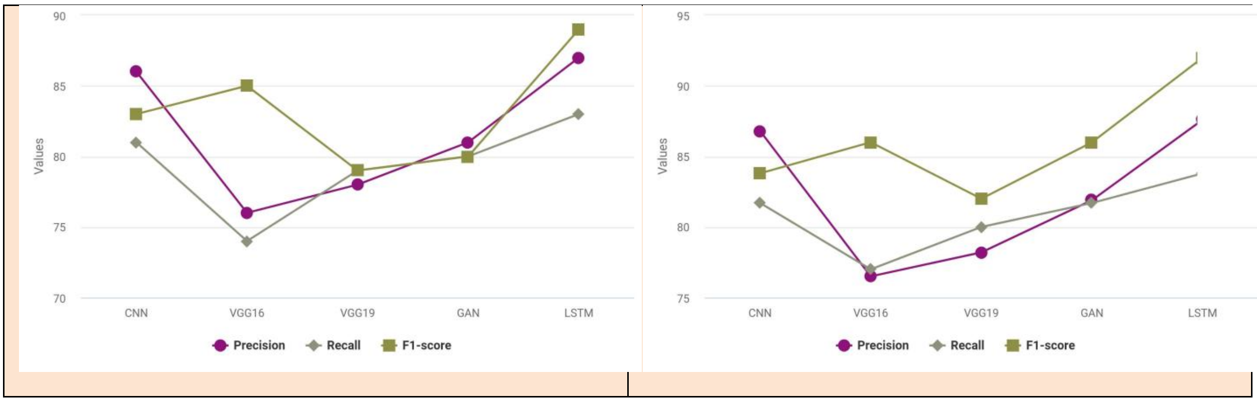


Figure 7. Model vs Accuracy, Sensitivity, Specificity over a) AWID, b) DARPA, c) UNSW-NB15

Precision, recall, and F1-score results are shown in Table 5 as a whole. The overall analysis for the various datasets is shown in Figure 8(a,b,c) under the headings of precision, recall, and F1-score.

Table 5. Overall analysis under precision, recall and F1-score

Models	Dataset	Precision	Recall	F1-score
CNN		86	81	83
VGG16		76	74	85
VGG19	AWID	78	79	79
GAN		81	80	80
LSTM		87	83	89
CNN		86.8	81.7	83.8
VGG16	DARPA	76.5	77	86
VGG19		78.2	80	82
GAN		81.9	81.7	86
LSTM		87.6	83.8	92
CNN		88	83	84
VGG16	UNSW-NB15	78	78	88
VGG19		80	81	84
GAN		84	82	89
LSTM		90	85	95



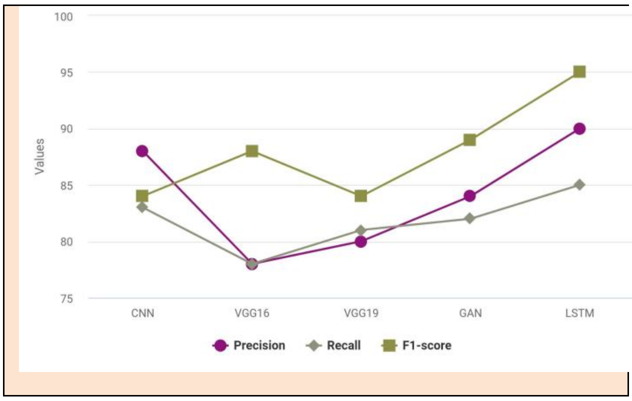


Figure 8. Model vs Precision, Recall, F1-score over a) AWID b) DARPA c) UNSW-NB15

Table 6 depicts the overall analysis under detection rate, TPR and FPR. Figure 9(a,b,c) depicts the graphical representation of various models over 3 datasets. Figure 10 depicts the overall security analysis of FAES.

Table 6. The overall analysis of detection rate, TPR FPR

Models	Dataset	Detection rate	TPR	FPR
CNN		90	85	15
VGG16		84	79	21
VGG19	AWID	87	81	19
GAN		81	72	28
LSTM		93	88	12
CNN		90.2	85.4	14.6
VGG16	DARPA	84.7	79.2	20.8
VGG19		87.9	81.9	18.1
GAN		81.3	73.5	26.5
LSTM		93.4	88.8	11.2
CNN		91.6	86	14
VGG16	UNSW-NB15	85	80	20
VGG19		88	82	18
GAN		82	74	26
LSTM		94	89	11

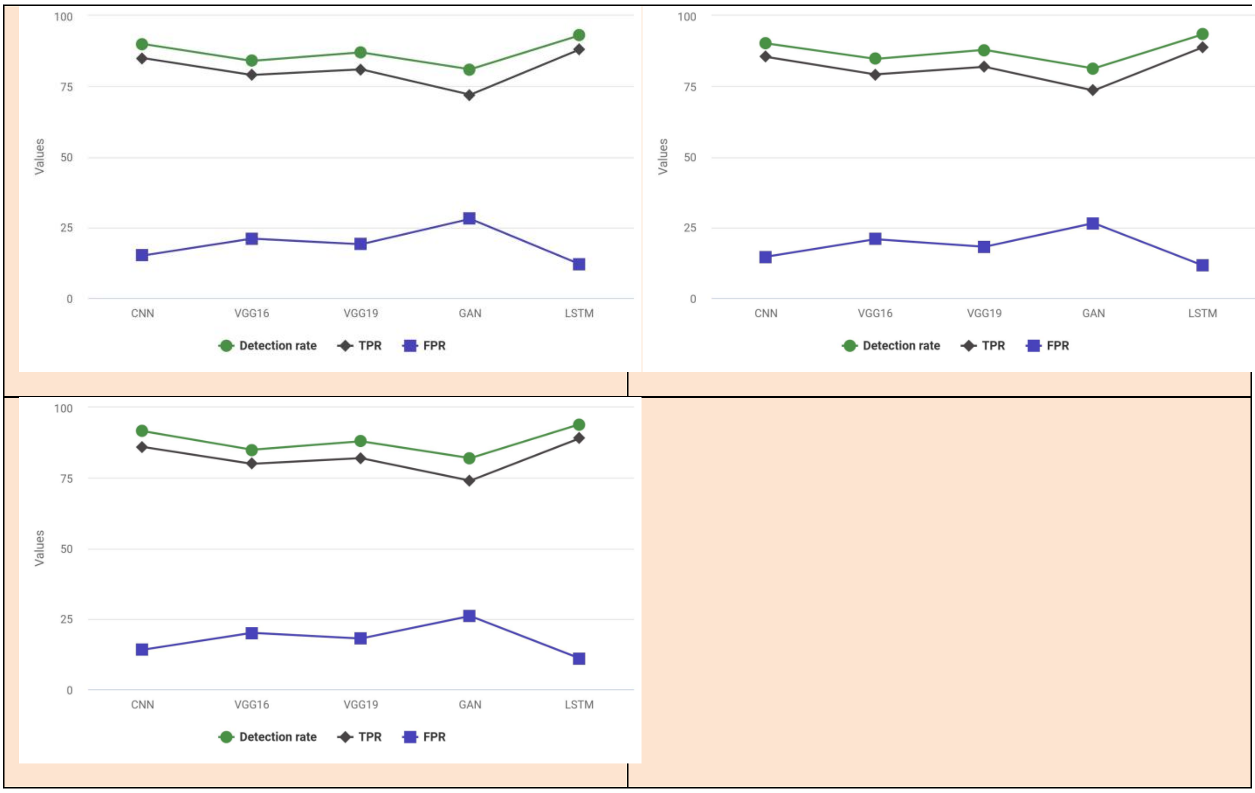


Figure 9. Model vs Detection rate, TPR, FPR over a) AWID, b) DARPA, c) UNSW-NB15

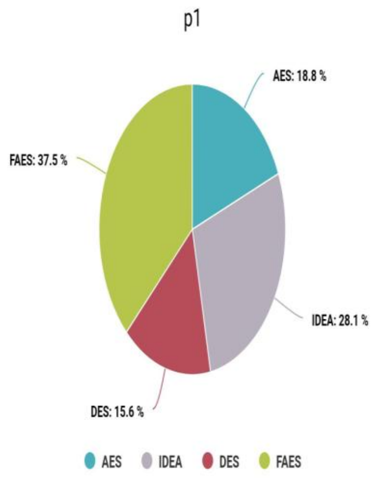


Figure 10. Security analysis of various Encryption models

5 CONCLUSION

This paper brings an effective deep learning model for attaching detection in routing using the LSTM network. The use of LSTM was good enough to bring a secure layer over a network and along with that use of effective feature extraction and selection, the stage adds another improvement to the classifier network as well. Also adding a final cryptographic layer with FAES brings double layer security to the overall framework as well. The proposed model is evaluated under various measures and it outperforms all other existing methods as well.

REFERENCES

J. Su and H. Liu, “Protecting flow design for DoS attack and defence at the MAC layer in mobile ad hoc network,” in International Conference on Applied Informatics and Communication, pp. 233–240, Springer, Berlin, Heidelberg, 2011

M. Chitkara and M. W. Ahmad, “Review on MANET: characteristics, challenges, imperatives, and routing protocols,” International journal of computer science and mobile computing., vol. 3, no. 2, pp. 432–437, 2014.

M. Sookhak, H. Tang, Y. He, and F. R. Yu, “Security and privacy of smart cities: a survey, research issues and challenges,” IEEE Communications Surveys & Tutorials., vol. 21, no. 2, pp. 1718–1743, 2019.

B. Mandal, S. Sarkar, S. Bhattacharya, U. Dasgupta, P. Ghosg, and D. Sanki, “A review on cooperative bait based intrusion detection in MANET,” SSRN 3515151.2020.

W. Qiao, M. Khishe, and S. Ravakhah, “Underwater targets classification using local wavelet acoustic pattern and multi-layer perceptron neural network optimized by a modified whale optimization algorithm,” Ocean Engineering, vol. 219, article 108415, 2021.

A. Ala, F. E. Alsaadi, M. Ahmadi, and S. Mirjalili, “Optimization of an appointment scheduling problem for healthcare systems based on the quality of fairness service using whale optimization algorithm and NSGA-II,” Scientific Reports, vol. 11, no. 1, pp. 1–19, 2021.

B. Alizadeh, D. Li, Z. Zhang, and A. H. Behzadan, "Feasibility study of urban flood mapping using traffic signs for route optimization," 2021, https://arxiv.org/abs/2109.11712.

M. F. Nezhadnaeini, M. Hajivand, M. Abasi, and S. Mohajerami, "Optimal Allocation of Distributed Generation Units Based on Different Objectives by a Novel Version Group Search Optimizer Algorithm in Unbalance Load System," Revue roumaine des sciences techniques Série Électrotechnique et Énergétique, vol. 61, no. 4, pp. 338–342, 2016.

S. Hassantabar, M. Ahmadi, and A. Sharifi, "Diagnosis and detection of infected tissue of COVID-19 patients based on lung X-ray image using convolutional neural network approaches," Chaos, Solitons & Fractals., vol. 140, no. 140, article 110170, 2020.

M. Ahmadi, A. Sharifi, S. Hassantabar, and S. Enayati, "QAIS-DSNN: tumour area segmentation of MRI image with optimized quantum matched-filter technique and deep spiking neural network," BioMed Research International., vol. 2021, pp. 1–16, 2021.

M. Ahmadi, A. Sharifi, M. Jafarian Fard, and N. Soleimani, "Detection of brain lesion location in MRI images using convolutional neural network and robust PCA," International Journal of Neuroscience., vol. 4, 2021.

M. Rezaei, F. Farahanipad, A. Dillhoff, R. Elmasri, and V. Athitsos, "Weakly-supervised hand part segmentation from depth images," in In The 14th PErvasive Technologies Related to Assistive Environments Conference, pp. 218–225, 2021.

F. Farahanipad, M. Rezaei, A. Dillhoff, F. Kamangar, and V. Athitsos, "A pipeline for hand 2-D keypoint localization using an unpaired image to image translation," in In The 14th PErvasive Technologies Related to Assistive Environments Conference, pp. 226–233, 2021

B. Alizadeh Kharazi and A. H. Behzadan, "Flood depth mapping in street photos with image processing and deep neural networks," Computers, Environment and Urban Systems, vol. 88, article 101628, 2021.

M. Ahmadi, F. Dashti Ahangar, N. Astaraki, M. Abbasi, and B. Babaei, "FWNNet: presentation of a new classifier of brain tumour diagnosis based on fuzzy logic and the wavelet-based neural network using machine-learning methods," Computational Intelligence and Neuroscience, vol. 2021, Article ID 8542637, 13 pages, 2021.

J. Wang, Y. Gao, X. Yin, F. Li, and H. J. Kim, "An enhanced PEGASIS algorithm with mobile sink support for wireless sensor networks," Wireless Communications and Mobile Computing., vol. 2018

M. Abasi, M. Joorabian, A. Saffarian, and S. G. Seifossadat, "Accurate simulation and modelling of the control system and the power electronics of a 72-pulse VSC-based generalized unified power flow controller (GUPFC)," Electrical Engineering, vol. 102, no. 3, pp. 1795–1819, 2020.

Ponguwala, M., & Rao, D. S. (2019). Secure group-based routing and flawless trust formulation in MANET using an unsupervised machine learning approach for IoT applications. EAI Endorsed Transactions on Energy Web, 6(24), e4-e4.

Sebopelo, R., Isong, B., & Gasela, N. (2019). Identification of compromised nodes in MANETs using machine learning technique. International Journal of Computer Network & Information Security, 11(1).

Srinivasan, V. (2021). Detection of Black Hole Attack Using Honeypot Agent-Based Scheme with Deep Learning Technique on MANET. Ingénierie des Systèmes d'Information, 26(6).

Majumder, S., & Bhattacharyya, D. (2019). Adopting machine learning techniques to mitigate various attacks in MANET—a survey report. Int. J. Sci. Res. Rev, 8(6), 288-295.

Laqtib, S., El Yassini, K., & Hasnaoui, M. L. (2020). A technical review and comparative analysis of machine learning techniques for intrusion detection systems in MANET. International Journal of Electrical and Computer Engineering, 10(3), 2701.

Lee, S. J., Yoo, P. D., Asyhari, A. T., Jhi, Y., Chermak, L., Yeun, C. Y., & Taha, K. (2020). IMPACT: Impersonation attack detection via edge computing using deep autoencoder and feature abstraction. IEEE Access, 8, 65520-65529.

Lopez-Martin, M., Carro, B., & Sanchez-Esguevillas, A. (2020). Application of deep reinforcement learning to intrusion detection for supervised problems. Expert Systems with Applications, 141, 112963.

Omrani, T., Dallali, A., Rhaimi, B. C., & Fattahi, J. (2017, December). Fusion of ANN and SVM classifiers for network attack detection. In 2017 18th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA) (pp. 374-377). IEEE.

Gniewkowski, M. (2020, June). An overview of DoS and DDoS attack detection techniques. In International Conference on Dependability and Complex Systems (pp. 233-241). Springer, Cham.

Kumar, V., Sinha, D., Das, A. K., Pandey, S. C., & Goswami, R. T. (2020). An integrated rule-based intrusion detection system: analysis of the UNSW-NB15 data set and the real-time online dataset. Cluster Computing, 23(2), 1397-1418.

Meftah, S., Rachidi, T., & Assem, N. (2019). Network-based intrusion detection using the UNSW-NB15 dataset. International Journal of Computing and Digital Systems, 8(5), 478-487.

Obeidat, I., Hamadneh, N., Alkasassbeh, M., Almseidin, M., & AlZubi, M. (2019). Intensive pre-processing of KDD cup 99 for network intrusion classification using machine learning techniques.

Wu, Y., Wei, D., & Feng, J. (2020). Network attacks detection methods based on deep learning techniques: a survey. Security and Communication Networks, 2020.

Azizjon, M., Jumabek, A., & Kim, W. (2020, February). 1D CNN-based network intrusion detection with normalization on imbalanced data. In 2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC) (pp. 218-224). IEEE.

Umar, M. A., & Zhanfang, C. (2020). Effects of Feature Selection and Normalization on Network Intrusion Detection.

Kunang, Y. N., Nurmaini, S., Stiawan, D., & Zarkasi, A. (2018, October). Automatic feature extraction using autoencoder in an intrusion detection system. In 2018 International Conference on Electrical Engineering and Computer Science (ICECOS) (pp. 219-224). IEEE.

Yan, B., & Han, G. (2018). Effective feature extraction via stacked sparse autoencoder to improve intrusion detection system. IEEE Access, 6, 41238-41248.

Moukhafi, M., El Yassini, K., & Bri, S. (2018). A novel hybrid GA and SVM with PSO feature selection for the intrusion detection system. Int. J. Adv. Sci. Res. Eng, 4(5), 129-134.

Abdel-Basset, M., Mohamed, R., Chakrabortty, R. K., Ryan, M. J., & Mirjalili, S. (2021). An efficient binary slime mould algorithm integrated with a novel attacking-feeding strategy for feature selection. Computers & Industrial Engineering, 153, 107078.

Alkahtani, H., & Aldhyani, T. H. (2021). Botnet attack detection by using the CNN-LSTM model for the Internet of Things applications. Security and Communication Networks, 2021.

Hossain, M. D., Ochiai, H., Fall, D., & Kadobayashi, Y. (2020, August). LSTM-based network attack detection: performance comparison by hyper-parameter values tuning. In 2020 7th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2020 6th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom) (pp. 62-69). IEEE.

Mamatha, C. R., & Ramakrishna, M. (2022). Integrating Interval Type-2 Fuzzy Logic and Modified Dingo optimization for Node Ranking and Energy Efficient MANET.

Singh, P., Khari, M., & Vimal, S. (2021). EESSMT: an energy-efficient hybrid scheme for securing mobile ad hoc networks using IoT. Wireless Personal Communications, 1-25.