Received April 16, 2018, accepted May 20, 2018, date of publication June 6, 2018, date of current version June 20, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2844255

A Highly Accurate Deep Learning Based Approach for Developing Wireless Sensor Network Middleware

REMAH A. ALSHININA, (Member, IEEE), AND KHALED M. ELLEITHY, (Senior Member, IEEE)
Department of Computer Science and Engineering, University of Bridgeport, Bridgeport, CT 06604, USA
Corresponding author: Remah A. Alshinina (ralshini@my.bridgeport.edu)

ABSTRACT Despite the popularity of wireless sensor networks (WSNs) in a wide range of applications, security problems associated with them have not been completely resolved. Middleware is generally introduced as an intermediate layer between WSNs and the end user to resolve some limitations, but most of the existing middleware is unable to protect data from malicious and unknown attacks during transmission. This paper introduces a secure wireless sensor network middleware (SWSNM) based on an unsupervised learning technique called generative adversarial network algorithm. SWSNM consists of two networks: a generator (G) network and a discriminator (D) network. The G creates fake data that are similar to the real sample and combines it with real data from the sensors to confuse the attacker. The D contains multi-layers that have the ability to differentiate between real and fake data. The output intended for this algorithm shows an actual interpretation of the data that is securely communicated through the WSN. The framework is implemented in Python with experiments performed using Keras. Results illustrate that SWSNM algorithm improves the accuracy of the data and enhances its security by protecting data from adversaries. In addition, the SWSNM algorithm consumes significantly less energy, has higher throughput, and lower end-to-end delay when compared with a similar conventional approach.

INDEX TERMS Middleware, unsupervised learning, WSNs, generator, discriminator, visualization, confusion matrix, security, GANs, energy consumption, delay.

I. INTRODUCTION

The rapid increase in the use of wireless sensor networks (WSNs) in a variety of scientific and industrial applications has pushed the research envelope in the field of computer and systems engineering [1]. The countless sensor nodes installed within WSNs enable wireless communication between nodes and devices. Distributed systems can be designed for data collection and processing using resource-constrained sensor nodes. Some of the tasks that WSNs perform are signal processing, data aggregation, quality of service, and wireless communication. Secure communication between WSNs has been a challenge in recent years [2]. WSNs produce massive data through their low-capacity sensors, which results in the loss of important information during transmission. In addition, sensor nodes have several limitations such as security, data aggregation, middleware requirements, power consumption, and the heterogeneity of the sensors' networks.

Previous research has shown that using middleware as an intermediate layer between WSNs and the end user provides a solution to the previously mentioned limitations. The middleware provides a bridging platform between the applications and the hardware components of WSNs. The middleware controls the sensor data nodes while providing them temporary storage [3]. The ability to synchronize newer nodes with the existing nodes allows the middleware to be more efficient while providing support to various resources. This allows minimum or no disturbance in the network's performance if changes occur to the network [4]. Since the data sent over the wireless networks is sensitive, it is prone to unwanted intrusions. Security parameters, such as resource distribution and resource management, enable secure communication within WSNs. End-to-end security auditing can also be enabled to achieve secure communication between nodes [5].

The loss of data during transmission to and from the middleware is still prone to attacks. Alshinina and Elleithy [4] showed a comprehensive, systematic study of the most recent research on WSNs' middleware and compared existing efficient system designs, addressed most significant challenges,

made several distinguished contributions including security, data aggregation, message exchange, and quality of service. The authors concluded that the middleware has to be both scalable to dynamic resources and secure at the same time. It was also hypothesized that synchronizing newer nodes with the existing nodes would allow the middleware to perform more efficiently while providing support to various resources.

II. RELATED WORK

In the last decade, wireless sensor networks (WSNs) have been applied in monitoring systems that are capable of controlling and monitoring various indoor premises, agricultural lands, and forest monitoring applications [6]. Foremost issues associated with WSNs are related to network security due to an increase in the usage of these devices. Traditional security algorithms in WSNs have achieved security goals such as base station protection [7], cryptography [8], attack detections [9], and security location and routing [10]–[12]. Many researchers have developed significant solutions to address WSNs' security challenges. The Intrusion Detection System (IDS) is a security management system that monitors all events within a network. IDS is capable of detecting attacks without compromising network security. The anomaly detection types of Intrusion Detection (ID) can detect any abnormal behavior in the online data. Misuse detection is another type of ID, which works on offline data and is able to detect known attacks [13], [14]. These sensors introduce massive data for processing and transmission to the base station. Standard security algorithms are not suitable for WSNs due to limitations in power consumption, low memory (storage capacity), communication capabilities and resource constraints in sensors [15], [16].

The communication and exchange of information between sensors is a critical challenge due to energy consumption in network. This information must be protected against various threats [17], [18]. The networks should be secured by support security properties such as confidentiality, authenticity, availability, and integrity. Standard [17] applied cryptographic algorithms such as signature and encryption/decryption. However, these mechanisms used secret keys that are unsuitable to the large scale of WSNs due to the large memory requirement to store these keys [17]. Most of these sensors lack physical protection, which leads to compromised nodes. Compromising one or more nodes in a network allows the adversary to launch different attacks to disrupt inter-network communication [19]. There are various attacks such as adversary, compromised node(s), eavesdropper, etc. [20]. These attacks are capable of dropping packets or modify them, resulting in an impact in the performance of WSNs. Source location privacy (SLPs) are mechanisms that protect sensor data from attacks by generating fake nodes. The fake node and packets (dummy message) create fake identity and packets without mentioning the source and destination identity. The drawback of this technique is that it requires more energy and overhead [19], [20].

A. WSNs MIDDLEWARE

Recently, middleware has been integrated with WSNs to address some of the aforementioned challenges. Alshinina and Elleithy [4] reviewed and discussed various middleware approaches such as SOMM, USEME, ESOA, and MiSense.

Most middleware approaches lack the security mechanism to secure the network and sensitive data from malicious attacks. While middleware systems are primarily developed for WSNs, different agents use it for various applications to detect any intrusion using the agent model. Lingaraj *et al.* [21] introduced mobile agent middleware called Eagilla that is integrated with the WSN for sensing data. This framework provides scalability and flexibility to the network. The agent is responsible for communication and acts as a mobile to move around in the network and update required tasks. Sensor nodes in the network act as a cluster head (CH) and run their agents based on the functionalities of the CH. The CH is applied to increase network scalability and the application is controlled by CH instead of the base station. There are three types of sensor nodes; free, client, and server. Free nodes act as independent nodes and can leave or join the cluster/network at any time. Server nodes are the CH that pass the communications to and from the base station. Finally, client nodes have communication authority with the CH. This framework increases the network scalability and supports heterogeneity of sensor hardware.

B. MACHINE LEARNING FOR WSNs

Middleware systems have also been introduced to apply machine learning (ML) algorithms. The work presented in [22] applied an unsupervised learning technique called self-organizing map (SOM) into WSN. Avram *et al.* [23] introduced SOM to address the problem of detecting network attacks in ad hoc networks. The limitation of SOM is that it is not suitable to detect attacks in complex and enormous dataset that are typically used in WSNs. Machine learning middleware (MaML) tackled the problem of ontology heterogeneity. However, the potential problem in MaML is the overhead. The dynamic behavior of a WSN has been continuously optimized due to system design requirements. In order to eliminate the need for an unnecessary redesign of the network, machine learning (ML) techniques are adopted in WSNs [24]. The designers of sensor networks describe Machine Learning as an algorithm and a collection tool that is used to create prediction models. ML improves resource allocation, utilization, and delegation in an effort to prolong the life of the network. ML uses mathematical models that are based on statistical methods for artificial intelligent data sampling. It learns and adapts to a constantly changing environment [25]. Interfacing techniques in ML play an important role in WSN applications. ML interfacing is carried out in three steps: the processing of data, data aggregation, and interfacing [26]. These steps are used for monitoring and modeling the dynamic environments associated with WSNs.

Machine learning is used to create prediction models, and these algorithms are categorized into supervised, unsupervised, and reinforcement learning [27]. Supervised learning takes place when the data sample (or the training set) is labeled. Machine learning algorithms such as support vector machine (SVM), decision tree (DT), and K-nearest neighbor (K-NN) have successfully addressed several challenges of WSNs such as data aggregation, localization, clustering, energy aware, detection and real-time routing.

The purpose of using an unsupervised learning is to classify data into different groups as clusters and enable them to investigate the similarity between the input samples. Reinforced learning takes place when results from learning assist in some sort of environment change. Reinforcement learning algorithms control the behavior of the agent (as sensor nodes) within their environments. Based on the rules, the agents in the environment can select the action to transmit it from one state to another [27]. Neural Networks (NNs) are ML models that can solve several challenges and tasks in WSNs such as quality of service (QoS) and security. There is an immense need to boost the security to improve the QoS through NNs, which are comprised of distributed computation nodes as well as WSNs [27].

Machine learning algorithms are used to address non-functional requirements associated with WSNs. However, accuracy problems might be associated with each of the machine learning algorithms. One of them, a non-functional requirement in WSNs, is security. Machine learning algorithms provide solutions to resource constraints that pose a major security challenge in WSNs [28]. The observations in the network can sometimes be misleading due to a number of factors, such as unexpected attacks or intrusions from intruders, so it becomes important to detect a particular anomaly through machine learning algorithms and maintain a secure network [24]. When machine learning is applied to WSNs, it helps decrease the vulnerability of WSNs to misleading information and unwanted attacks.

The implementation of ML also drastically increases the reliability of the network by eliminating misleading information and unexpected intrusions. Additionally, ML techniques also increase the lifespan of the WSN by significantly reducing energy required by the sensor nodes. Moreover, ML also reduces (and strives to eliminate) human intervention, resulting in the meticulous use of the WSNs.

The design of WSN is challenging because of its dynamic environment that makes the models' parameter optimization difficult. Genetic Algorithm (GA) is well known for flexibility and can be applied in a wide variety of scenarios in WSNs. GA is applied to address many WSN challenges such as communication, topology control, and for finding the shortest path in a large-scale dynamic network [29]. Vasseur *et al.* [30] introduced a new approach for predictive learning machine to detect traffic outside of service level agreements (SLA). The request is to make predications for one or more SLA in the network. The requirement of network traffic parameter and SLA are linked with network traffic parameter depending on the number of SLA. The learning machine (LM) employs to map the estimates of network performance metrics with target SLA.

Literature [31]–[34] presents a number of machine learning algorithms that address the security problems in WSNs. Janakiram *et al.* [31] showed the detection of outliers using Bayesian belief networks (BBNs). The authors correlated both temporal and spatial data points to identify similar readings in neighboring nodes. These readings are approximated and matched with one another to find possible outliers in the data obtained from sensor nodes. Conditional relationships are built to not only identify outlying data points, but also to fill in the missing data [31]. Similar to the investigation of k-nearest neighbor presented in [35], Branch *et al.* [32] developed an outlier detection method within the network using k-nearest neighbor. A major disadvantage, however, of using the k-nearest method is that it requires a large memory space to store data.

Black hole attacks are common in the transmission of data in WSNs. In such attacks, misleading routing reply messages are sent by the nodes whenever route requests are received. These misleading messages result in the termination of the route discovery; real routing reply messages are ignored [24]. Kapalantzis *et al.* [33] presented a mechanism of detecting similar forwarding attacks using support vector machine (SVM). This mechanism detects black hole attacks by using routing information, bandwidth, and the hop count of the nodes [33]. Rajasegarar *et al.* [34] were able to combine SVM to the outlier detection scheme to establish a one-class, quarter-sphere SVM anomaly recognition technique [34]. The SVM methods are far superior due to their efficient learning and enhanced performance in non-linear and complex network problems.

On the basis of the related work and its implications, it is noted that most existing middleware approaches do not provide a comprehensive system to handle massive data and communicate between sensors and the base station securely. Since data is most prone to attack during its transmission, a robust method that not only provides secure communication, but also enhances the efficiency of the network is needed. This paper introduces a novel technique, based on machine learning that originates fake data to secure communication between sensor nodes and base station by deceiving the attacker. This technique eliminates the need to generate fake packets or nodes for security and reduces power consumption, end-to-end delay, and increases throughput.

The remaining parts of this paper are organized as follows: Section III describe the motivation and research problem. Section IV explains the Generative Adversarial Networks (GANs) algorithm. The proposed unsupervised learning for WSNs' middleware methodology is illustrated in Section V. Section VI provides experimental results and discussion. Section VII provides details of NS2 simulation. Finally, Section VIII shows significant conclusions from the research.

III. MOTIVATIONS AND RESEARCH PROBLEM

To address the security challenges of WSNs, we developed an intelligent WSN middleware based on an unsupervised learning approach that provides a comprehensive security algorithm that can handle large-scale WSNs.

The proposed middleware is able to secure information and resources from malicious attacks and also detect node misbehavior. The special characteristics of WSN such as power consumption, throughput, and network lifetime are taken into account in this contribution. The proposed intelligent wireless sensor network middleware, which is based on generative adversarial networks, has improved the traditional middleware and other security mechanism but can handle the heterogeneous characteristics of sensor nodes and is capable of filtering and passing only real data. To the best of our knowledge, it is the first time that the GANs algorithm has been used for solving the security problem in WSNs' middleware. Additionally, in the proposed contribution, WSNs' middleware applies a GAN that is capable of filtering and detecting anomalies in the data.

The proposed approach is motivated by the limitations of the existing middleware and will improves performance based on the following reasons:

1) The proposed techniques provide a unique WSN middleware which can control and monitor sensor data by using intelligent, unsupervised machine learning to secure the data. The power consumption and overhead can be increased by updating and filtering unnecessary information from the sensors. This problem is addressed through the proposed unsupervised learning.
2) From the given samples, the generator network creates fake data very similar to the real data. This fake data is combined with the real data from sensors so that the attackers cannot differentiate between them. In this case, there is no need to generate fake packets or data to confuse the attackers, which significantly decreases power consumption.
3) Different analytical models are developed: Confusion Matrix, Visualization, and different CNNs layers confirm the validation of the proposed algorithm.
4) We provide a comprehensive comparison with other approaches such as Eagilla for verification of the proposed approach. The following metrics are used in comparison: average energy consumption, throughput, and end-to-end delay.

IV. GENERATIVE ADVERSARIAL NETWORKS (GANs)

The GANs, inspired by Goodfellow *et al.* [36] in 2014, are a class of artificial intelligence and are used in unsupervised ML. The GANs contain two networks: the generator (G) network and the discriminator (D) network as a minimax two player game [36]–[38].

The generator network creates fake data similar to the real samples, and the fake data passes through the discriminator network (D) with data from the real distribution as inputs.

Figure 1 represents the general model of a GANs algorithm. The G network is designed to learn the distribution of the training data, while the D network is designed to calculate the probability of the data originating from the training data (real) rather than the generator data (fake). These networks improved WSNs' performance and optimization during iterative optimization and mutual confrontation. The discriminator improves by extending the target dataset. The generator and discriminator networks must be differentiable during implementation. The proposed GANs provide an efficient way to learn deep representations without extensively explained training data. These networks achieve this by deriving backpropagation results from the competitive process including a pair of networks as shown in Figure 2.

$$\min G \max D = x \in D \, logD\,(x) + x \in g \, log\,(1 - D\,(G\,(z))) \tag{1}$$

The general formula for GANs is shown in equation 1. The D takes real data (x) and fake data from the generator, represented as G(z), and the output is the probability of that data being real (p(x)). Thus, the D network is capable of increasing the likelihood of identifying real data and lowering the probability of accepting fake data (from the generator). The G network takes vector random number (z) as the input. The first term corresponds to optimizing the probability of the real data (x) (close to 1) and the second term corresponds to optimizing the probability of the fake data (G(z)) (close to zero) [36]–[38].

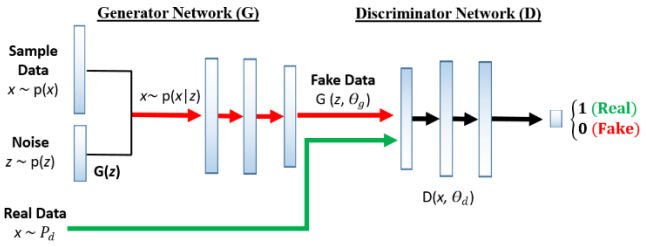

FIGURE 1. GANs framework illustrates the sample flow from generator network (G) to discriminator network (D).

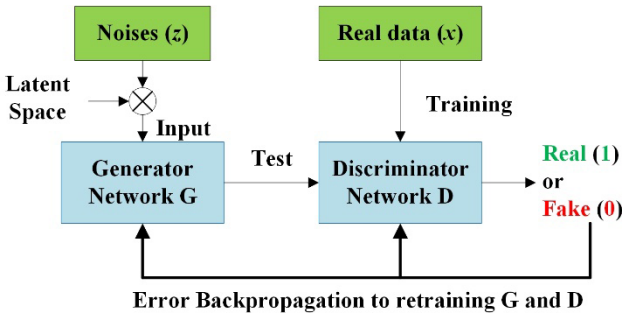

FIGURE 2. Two models which are learned during the training process for a GAN are the discriminator (D) and the generator (G).

The proposed GANs are based on a minimax game where one agent attempts to maximize the probability while the other attempts to minimize it. G's ability to generate new data

that is similar to the real samples is thus improved. The idea is to confuse the attacker and prevent them from differentiating between the new data from the generator and the real data from the sensors and dataset. The D differentiates between real and attack data by maximizing the probability of the real data to 1 and minimizing the probability of fake data (from the G or an attacker) to 0.

V. THE PROPOSED WSN MIDDLEWARE

We introduce an intelligent security system for WSNs' middleware based on GANs to improve traditional middleware in terms of security mechanism, handling of heterogeneous characteristics of sensor nodes, and to filter and pass only the real data. To the best of our knowledge, it is the first time that the GANs algorithm has been used for solving the security problem in WSNs' middleware. Additionally, in the proposed contribution, WSNs' middleware applies a GAN that is capable of filtering and detecting anomalies in the data. The proposed procedure is described in Algorithm 1:

A. DATASET

Many conventional classifiers fail to differentiate between normal and attack traffic. The benchmark NSL-KDD dataset [39] is used to detect any intrusion in the sensors' data in the system. The NSL-KDD contains an imbalance of classes in normal and attack data traffic. The ratio of attack to normal traffic is comparatively low. The phenomenon of normal traffic outweighing the attack traffic is referred to as the Class Imbalance Problem (CIP). This occurs when the minority class, also known as the attack class, exhibits a much lower representation in comparison to that of the majority, or normal, attack classes. The CIP benefits the attack traffic, and the intrusion detection system is unable to withstand it. Therefore, there is a strong need to identify specialized techniques to counteract such an attack by placing an importance on minority classes.

The proposed approach solves the imbalance problem through the proposed generator model. The main difference between this model and existing algorithms is that the generator creates a balanced data that is more representative of the real data by providing the generator only one feature vector of this dataset. This feature is then used as feedback in the discriminator, enabling it to distinguish between fake data (corresponding to 0) and real data (corresponding to 1).

B. GENERATOR NETWORK

The proposed generator network (G) is used to create various attacks data (fake) from one sample (which is acquired from the NSL-KDD dataset). Crucially, generator has no any direct access to the real data (dataset) G learns only through its interactions with D. The discriminator has access to both the real data and the synthetic data drawn from the dataset. From the error backpropagation results, as shown in Figure 2, the G uses it to retrain the generator again, leading it towards being able to produce fake data of better quality.

Algorithm 1 The Proposed Framework Pseudocode

1: **Inputs**: training set : $X = \{(x_i, y_i)\}_{i=1}^{N}$, N_g: sample size is randomly selected from X for Generator (G) to learn data distribution
Inputs: M_F: number of fake data will be generated from the G once the training is completed, n : mini-batch size, T_{ts} is testing set
2: **Outputs**: M_F samples generated from the G, *Accuracy*
3: Select N_g samples (x) randomly from original data
4: **For** $i = 1$ to training iterations **do**
5: **For** k steps **do**
6: Sample of n noise samples $z = \{z_1, z_2, \ldots z_n\}$ from noise $p_g(z)$
7: select n samples from original data $x = \{x_1, x_2, \ldots x_n\}$
8: Concatenate x and z. Then, define y $= [1] * n + [0] * n$
9: Update the Discriminator by descending its stochastic gradient
$$\nabla_{\theta_d} \frac{1}{2n} \sum_{i=1}^{2n} \left[\log D\left(x_i\right) + \log\left(1 - D(G(z_i))\right)\right]$$
10: Update the Generator by descending its stochastic gradient
$$\nabla_{\theta_g} \frac{1}{n} \sum_{i=1}^{n} \log((1 - D\left(G(z_i)\right))$$
End for
11: **End for**
12: Generate new data (T_d) from the Generator after the training is completed.
13: $T_r =$ Append x to T_d
14: $T_r =$ Shuffle (T_r)
15: Update the Discriminator by descending its stochastic gradient
$$\nabla_{\theta_d} \frac{1}{n} \sum_{i=1}^{n} \left[\log D\left(x_i\right)\right]$$
16: Compute the accuracy of the Discriminator based on testing set T_{ts}
17: **Return** accuracy

The generator network is mapped from a representation space called latent space to space of data. The general formula $g : g(z) \rightarrow R^{|x|}$ where $z \in R^{|z|}$ is a sample from latent space, where the data is $x \in R^{|x|}$ then turns these into multilayer feed forward neural networks with a weight of θg. The proposed G network calculates this with equation 2. The output of the generator is$g = \{x_i\}_{i=1}^{M_F}$

Where M_F stands for the newly generated fake data from G with random sample data $z = \{z_i\}_{i=1}^{M_F}$ as inputs.

$$G = \sum_{o=1}^{h} \sum_{i=1}^{N} \beta_o f((\omega_i) + v_o) \tag{2}$$

Where h is the number of hidden neural nodes, o and i represent the output and input of the hidden layers respectively,

f stands for the activation function in the neural networks, ω_i represents the input weights of the *i-th* hidden neural nodes. β_o is the output weights, and v_o represents the threshold values of the *i-th* hidden neural nodes.

The output of this network range (0, 1) contains the numbers of neurons, where the activation function applied in the last layer of this network is Sigmoid. The first layer of the G, the noise input, is fully connected, and this layer is reshaped into a size of (8×5) and then fed into the convolutional layers. G's architecture is comprised of a fully connected layer and two convolutional layers. This network architecture is illustrated in Figure 3.

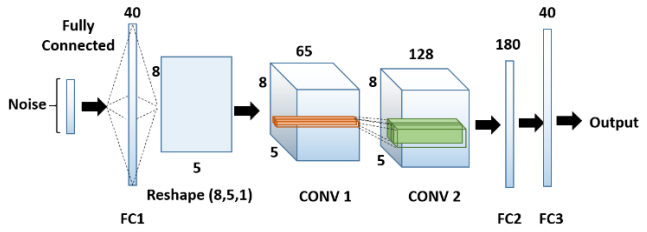

FIGURE 3. The Generator Network's Architecture.

C. DISCRIMINATOR NETWORK

The discriminator D, takes both real (authentic) and fake data, and aims to differentiate between them. Both the G and D networks are trained simultaneously and in competition with each other. Therefore, the discriminator has access to both the real data and synthetic data drawn from the dataset. From The D uses error backpropagation results for 150 iterations as shown in Figure 2 to retraining and updated, and leading it towards being able to distinguish between real and fake data.

The inputs of the discriminator are $D = \{x_i\}_{i=1}^{N}$, where N represents the sample number of the dataset. The discriminator is initialized in Keras (TensorFlow) as shows in following equation 3:

$$D(x_i) = \sum_{o=1}^{h} \sum_{i=1}^{N} \beta_o f(\omega_i^T x_i + v_o) \tag{3}$$

where h is the number of hidden neural nodes, o and i represent the output and input of the hidden layers respectively, f stands for the activation function in the neural networks, ω_i represents the input weights of the i-th hidden neural nodes. β_o is the output weights, and v_o represents the threshold values of the i-th hidden neural nodes.

In the training set, the discriminator takes

$$g = \{x_i\}_{i=1}^{M_F} \text{ and } D = \{x_i\}_{i=1}^{N} \text{ as inputs,}$$

with the outputs zero for fake/attacks data and one for real data respectively. The discriminator is capable of determining the probability of new generated fake data falling within the interval time; if it does, then the D network accepts it as real data. The G network performs very well in convergence.

The generated fake data (new)**g** and the real dataset **D** will combined and then send full data to destination, base station. The base station then takes the combined data, defined as $D = \{x_i\}_{i=1}^{N+M_F}$ then feeds into another discriminator to distinguish between the real and fake data, filtering them before transmitting them to user.

The discriminator network D contains multiple-layers that feedforward the neural network with a weight of θd. The input is a feature vector x. The D network has the ability to differentiate between real and attack data. The training data for D network is comprised of real data and malicious (attack) data generated by the generator. The output shows a true interpretation of whether the data is normal or abnormal. Figure 4 shows the visualization of the discriminator network's architecture. The first layer of D is the input, the real and fake data from the G network. The last convolutional layer of D is flattened and then fed into a sigmoid function, giving an output in the range of 0 to 1. Batch normalization is used as the input for both the D and G networks, shifting inputs to zero-mean and unit variance. This method helps deal with training issues from poor initialization and supports the gradient flow in deeper models.

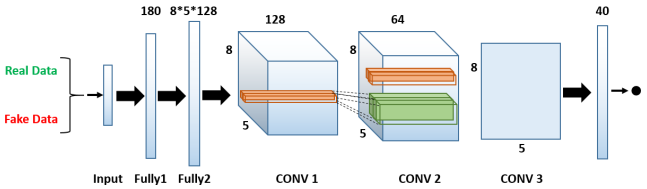

FIGURE 4. The Discriminator Network's Architecture.

VI. EXPERIMENTAL RESULTS AND DISCUSSION

In this architecture, we use the publicly available NSL-KDD dataset [39], [40]. NSL-KDD is solved redundant records and duplicate data issues in training set in KDDcup99 dataset [41], [42]. Moreover, this issues affects the performance of evaluate system. The proposed approach is used NSL-KDD dataset for training and testing that is comprised of normal and attacks data.

The proposed technique consists of following steps. First, split dataset into training set and testing set, shown in Table 1, is made up of 125,973 data samples in the training set and 22,544 samples in the testing set. The testing set is also comprised of additional attacks that are not in the training dataset. This dataset has 41 different features to define each threat as shown in Table 1. Second, perform pre-processing, the NSL-KDD dataset should be converted to binary, since the neural networks can only process this type of data. Once converted, the dataset can feed into the neural network model as an input layer. Preprocessing this dataset is done by hand, similar to other techniques such as the flag, service, and protocol types, and is converted as a number from 1 to 100. For example, the flag pre-processing technique uses OTH=76 and REJ=77 [13], [43].

TABLE 1. Overview of NSL-KDD dataset.

Dataset	Normal	Attacks	Total
NSL-KDD Train	67343	58630	125973
NSL-KDD Test	9711	12833	22544

A. EXPERIMENTAL SETTING

The proposed framework is implemented in Python and all experiments are performed in the Keras library [44]. Keras is a high-level neural network API and self-contained framework for deep learning. It supports scikit-learn features such as grid search, cross validation, etc. The framework was evaluated on the NSL-KDD dataset by using 40 features. The analytical model was developed using MATLAB.

1) GENERATOR NETWORK SETTING

The G is designed with a fully connected layer with 40 neurons. We then reshape the output of the fully connected layer into a size of (8×5) before feeding it into two convolutional layers. The last two layers are fully connected, as shown in Figure 3. We employ batch normalization [45] in some layers to normalize the inputs into zero-mean and unit variance to make the learning faster. We train the G model using the stochastic gradient descent (SGD) optimization algorithm with a mini-batch size of 128. The learning rate is set to 0.01 and the momentum at 0.9 for 150 epochs. The hyperbolic tangent (tanh) activation function applies for all layers. We use the L2 norm regularizer to prevent overfitting with a weight decay of 0.001.

2) DISCRIMINATOR NETWORK SETTING

We train the D network using Adam Optimizer with a learning rate of 0.001 with momentum. The mini-batch is 128, $\beta_2 = 0.5$ and $\beta_2 = 0.99$, which helps stabilize the training. Adam optimizer has shown faster convergence than SGD. We employ dropout with a rate of 0.5 for fully connected layers to combat overfitting. The Sigmoid output is a scalar value of the probability of whether data is real or an attack. For the real data, the scalar output is more than 0.5, and for attacks, the output is less than 0.5 as shown in Figure 4.

The weights of all of the layers in the G and D networks are initialized according to the Xavier initialization [46] technique and biases are set to zero. The input features of each vector is normalized between -1 and 1.

B. CONVOLUTIONAL NEURAL NETWORKS (CNNs)

There is significant design research on deep Convolutional Neural Networks (CNNs) layers to achieve improved accuracy [47]–[50]. He and Sun [47] performed an experimental study on depth, or the total number of layers in a network. The author kept time constraints constant while only increasing the depth. This practice resulted in an overall performance reduction, having more layers makes the network more difficult to optimize and more prone to overfitting. Moreover, the accuracy becomes either stagnant with increased depth or much reduced. Literature has shown that while the training errors tend to decrease, errors increase with low accuracy after a while [47].

Since deep networks are mostly used for complex data with multiple classes, we used a simple binary classification dataset in our proposed architecture: the number of CNN layers is set to three to obtain a high-performing network. Experimentally, increasing CNN layers leads to inaccuracy while also requiring a higher computational time and cost.

A high-performance, optimized architecture is obtained with three CNN layers to maintain accuracy of results while also minimizing overhead and overfitting, as shown in Table 2. Increased CNN layers can affect the accuracy and provide a high loss function based on data generated from the generator network (G). The loss function for the generator is computed by using the feedback from D. We use stochastic gradient descent (SGD) with a learning rate of 0.01 on over 150 training iterations to minimize loss.

TABLE 2. The accuracy comparison for different layers of CNN architectures.

Number of CNNs Layers	Accuracy
6 or more	82 %
5	84%
4	86%
3	~87%

Table 2 shows that the accuracy increases with a minimum number of layers, with the optimum accuracy achieved with three. The results in Figure 1 illustrate that the quality of data generated improves by increasing the accuracy and minimizing the loss function of the G. The G network is updated based on the output feedback from the D network until it generates more accurate data that the D accepts as real.

C. CONFUSION MATRIX

The confusion matrix is applied to evaluate the performance and effectiveness of the proposed G network and the original dataset, NSL-KDD. For this purpose, the Accuracy Rate (AR), False Positive Rate (FPR), True Positive Rate (TPR/Recall), and F-measure (F1) are applied and computed by equations, numbered 4, 5, 6, and 7 respectively. In the equations, TP, TN, FP, and FN denote number of true positive, true negative, false positive, and false negative cases respectively.

$$\text{AR} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{4}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{5}$$

$$\text{TPR/Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{6}$$

$$\text{F}_1 = \frac{2(\text{P}^*\text{R})}{\text{P} + \text{R}} \tag{7}$$

$$\text{P} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{8}$$

$$\text{Error Rate} = 1\text{-AR} \tag{9}$$

1) FULL FEATURE OF NSL-KDD DATASET

In this section, 40 features of NSL-KDD dataset are used to evaluate the performance of the proposed approach. Figure 5 shows the confusion matrix between testing target output and predicted output for the generated data from the G network. The G network achieved a better binary distribution while also improving the accuracy and decreasing the classification error. Additionally, TN and FP are two main criteria for evaluating the performance of the G network data compared with the NSL-KDD dataset results.

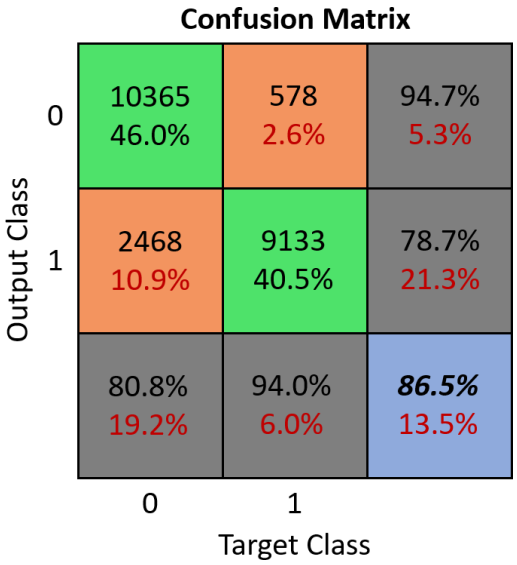

FIGURE 5. Generated data from the proposed generator network.

The results show that FP is reduced from 14.3% to 10.9% and the FPR is lower compared to the original dataset and existing algorithms. The original dataset produced a lower accuracy (81.1%) with a high FPR (27.1%) and FP (14.3 %), as shown in Figure 6. Precision (P) is a measure of accuracy correctness achieved in the positive prediction of the class, calculated from equation 8. The Recall (R), or TPR, is a measure of whether or not actual observations will be predicted correctly and is obtained from equation 6. The low precision and high recall show that most positive examples are correctly recognized due to a decrease in FN. The F-Measure (F1) is the harmonic mean that measures the quality of classifications. F-Measure computes the average of P and R, as given in equation 7.

The aim is to provide a high level of adversarial system on our generator model, one that is much better than the attack samples through an increase in accuracy and a decrease in the error rate. FPR occurs when the results are incorrectly predicted positive when they are indeed negative, an outcome which is reduced in the proposed model, obtained in equation 5. The experimental results show that the proposed generator network gives significantly better accuracy and a robust representation of data with the ability to reduce the error rate from 17.4% to 10.9%.

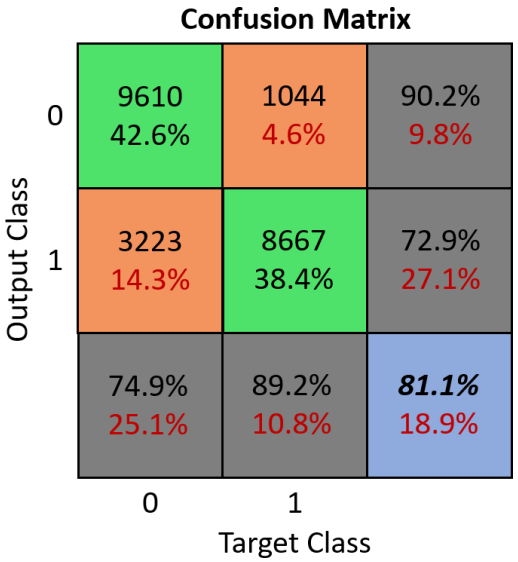

FIGURE 6. Original Dataset (NSL-KDD).

The Error Rate (ER) can be calculated from the accuracy. The accuracy is the number of correct classifications divided by the total number of classifications. The ER is obtained from equation 9. The results obtained from the proposed G network were evaluated based on the error rate, FPR and F1, and compared with the NSL-KDD dataset and Artificial Neural Network (ANN) approach [51]. Table 3 shows the limitations of the dataset and the ANN technique due to a high error rate in FPR and low accuracy.

TABLE 3. Comparison of proposed method with different methods.

Method	FP	FPR	F-Measure
Original Data [39]	14.3%	27%	81.8%
Artificial Neural Network (ANN) [51]	17.4%	31%	81.6%
SWSNM	**10.9%**	**21%**	**87.2%**

We compared the performance of our approach alongside existing methods that use the NSL-KDD dataset with 40 features. In Table 4, the ML algorithms are simulated to perform

TABLE 4. Comparison of accuracy rate of SWSNM with other ML methods.

Method	Accuracy
SVM [40]	69.52%
Decision Tree [40]	81.5%
DMNB with RP [52]	81.47%
SOM [53]	75.49%
ANN based IDS [51]	81.2%
Original Dataset	81.1%
SWSNM	**86.5%**

this comparison. As shown in Table 4, the proposed model achieves significantly better accuracy with a lower error rate. The performance of ML techniques optimized accuracy over the NSL-KDD dataset. For example, the accuracy of support vector machine (SVM) [40] and decision tree [40] are much lower compared to other ML techniques [40]. Panda *et al.* [52] introduced Discriminative Multinomial parameter learning using Naïve Bayes (DMNB) with a supervised filter called Random Projection at the second level. The authors achieved 81.47% accuracy in their system. Ibrahim *et al.* [53] implemented self-organizing map (SOM) with a very low accuracy rate. The ANN [51] reported that their accuracy was similar to other ML techniques.

2) FEATURES SELECTION

Feature reduction is applied by using principal component analysis (PCA). The goal of PCA is to select the most significant feature and reduce the dimensionality of the data into 20 features while keeping the variation in the NSL-KDD dataset as much as possible. Figure 7 shows that the G network generates 86.4% accurate data with the FPR of 18.5% in comparison to the original dataset with 76.3% accuracy and FPR of 33.8%, as shown in Figure 8. The results in Table 5 show that FP is reduced from 14.3% to 8.7% of the data generated from G network with 40 features, due to the efficiency of the GAN algorithm.

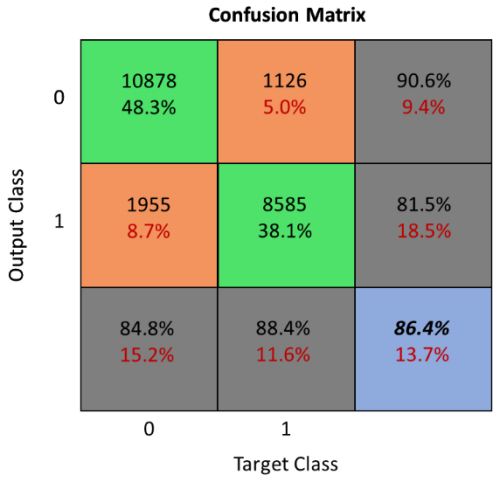

FIGURE 7. Generated data from G Network with 20 features.

TABLE 5. The comparison results between our g networks with original dataset (nsl-kdd) with only 20 features.

Method	FPR	FP	F_1
Original Dataset	33.8%	20.2%	75.5%
SWSNM	**18.5%**	**8.7%**	**87.6%**

In Table 6, different ML algorithms are simulated to carry out comparative analysis with 20 features. It can be observed that SWSNM produced a much higher accuracy when the

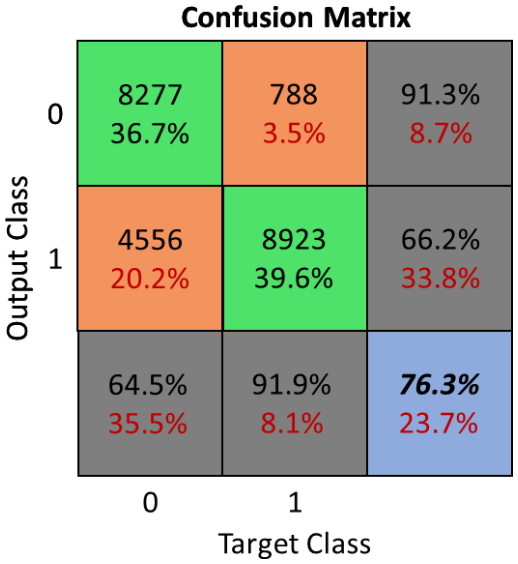

FIGURE 8. Original Dataset (NSL-KDD) with 20 features.

TABLE 6. Comparison of accuracy rate of SWSNM with other ML methods with 20 features.

Method	Accuracy
SVM	78.7%
Decision Tree	81.1%
AdaBoost	77.6%
Original Dataset	76.3%
SWSNM	**86.4%**

selection features are applied. Moreover, in Table 5, the F_1 for SWSNM is higher than NSL-KDD dataset, more specifically FPR is reduced from 33.8% (for the original dataset) to 18.5% (for SWSNM).

D. DATA VISUALIZATION

The t-distribution stochastic neighbor embedding (t-SNE) is a machine learning algorithm used to visualize the structure of very large data [54]. The visualization produced by this algorithm is significantly better on almost all datasets. We used t-SNE to visualize the output data of our model's results and compared it with the original dataset for both full feature (40 features) and reduced feature (20 features). The aim is to take a set of points in high-dimensional space and find the correct representation of those points in a lower-dimensional space (2D). The t-SNE builds a probability distribution over pairs of high-dimensional data in such a way that similar data have high probability of being selected, while dissimilar have small probability of being selected. It minimizes the divergence between the two distributions. Suppose a given dataset of objects $x = (x_1, x_2..., x_N)$ in which each point has a very high dimension and function $d = (x_i, x_j)$ computes a distance between pair of objects then convert it into two-dimensional data $x_j = (x_1, x_2, ...x_N)$. The similarity of data

point x_j to data point x_i is the conditional probability $P(j|i)$, and D is the number of data points obtained, as represented in equation 10. The t-SNE aims to learn a d-dimensional map of $y_i = (y_1, y_2, ..y_N)$ that reflects the similarities of P_{ij}.

$$P_{ij} = \frac{P_{j|i} + P_{i|j}}{2D} \tag{10}$$

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k^1 m} (1 + ||y_k - y_m||^2)^{-1}} \tag{11}$$

The similarity measure q_{ij} of two points y_i and y_j is defined in equation 11. The t-distribution can withstand outliers and is faster in evaluating data. The original dataset and the data generated from our model contained a high number of dimensions along which the data is distributed.

The original NSL-KDD dataset, shown in Figures 9a and 10a, reveal poor visualization in comparison to the data generated through the proposed model, as evident from Figures 9b and 10b. The experiment shows the G network has produced accurate data and achieved diversity with more coverage of the data distribution. The original dataset has poor diversity and less coverage of the data distribution.

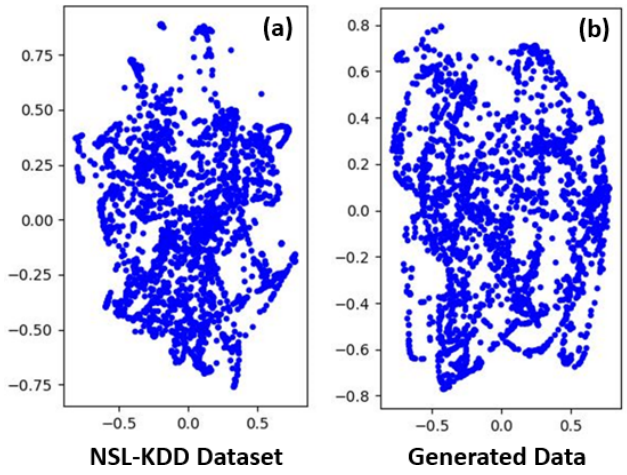

FIGURE 9. t-SNE Visualization with Full features. (a) Original Dataset (NSL-KDD) and (b) Generated Data in proposed SWSNM.

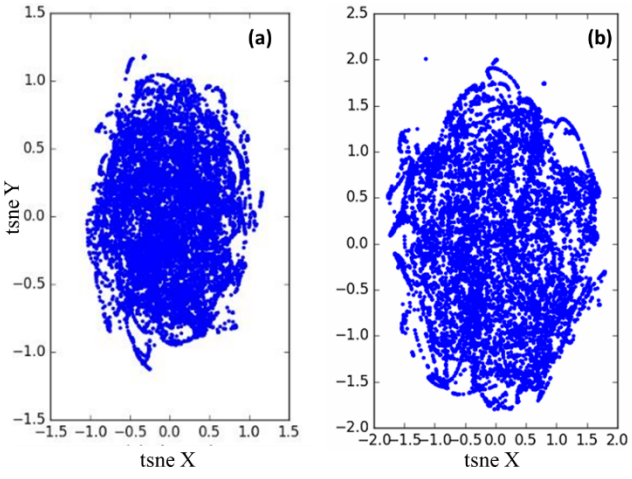

FIGURE 10. t-SNE Visualization with 20 Features. (a) Original Dataset (NSL-KDD) and (b) Generated Data in proposed SWSNM.

E. REFEEDING THE GENERATED DATA

In this section, the generated data with accuracy of 86.5% obtained from the generator network is re-fed to generate new data. The G network is able to generate a better quality data and takes much less time than the first time training.

Figure 11 shows the confusion matrix results after 150 iterations. It is crucial to consider the FP rate since it represents the cost of learning. The aim is to have a high TP rate (high benefits) and a low FP rate (low costs). Figure 11 shows that the G network is capable of generating accurate data with 85.1% accuracy and much lower FPR of 20.4%.

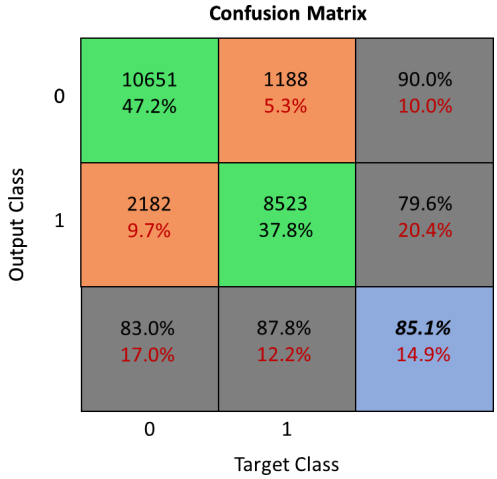

FIGURE 11. Re-feeding the Generated Data into Generator Network.

F. EVALUATION

We evaluate the capacity of the proposed wireless sensor network middleware (WSNM) based on machine learning for adaptive evolution through a component in middleware called adoption. Adoption allows the addition of new sensor nodes during runtime in a secure manner. Mechanisms such as security, flexibility, and fault-tolerance must be considered during middleware implementation [4]. Numerous standard algorithms are applied to detect node failure [55], [56]. Jiang [57] proposed a distributed fault detection approach capable of checking node failure through an exchange of data between neighboring nodes within the network. However, this scheme is not suitable for diagnosis or the detection of accuracy with small number of neighboring nodes [57].

Sensor nodes are prone to failure due to energy constraints and environmental factors that frequently affect the network topology. In our contribution, we consider the message freshness mechanism, which ensures that the existing data is new and guarantees that no adversary uses old data (messages). Moreover, new sensors are easily deployed by considering the forward and backward secrecy mechanism [7]. Forward secrecy restricts nodes from failing or leaving the network with future data. Backward secrecy does not allow any node to join the network to read any previous transmitted data [7].

Most existing security algorithms are impractical for WSNs due to the resource constraints in nodes. We applied a unique, unsupervised learning technique on the middleware to secure the entire network. The proposed middleware supports and adapts to node failure and node mobility without affecting the performance of the overall network. We designed a scalable middleware where the network has the capability to grow in size while continuing to meet network's security requirements. Middleware based on machine learning techniques can not only minimize the probability of node failure, but also eliminate the need for a network redesign. The intelligent discriminator (D) is capable of detecting attacks and diagnosing failed nodes and abnormal data. In case of incorrect readings from nodes, the information is sent to D. The D will consider this reading as faulty and remove the node from the network because it can negatively affect the performance and accuracy of the network.

The proposed architecture has the capability of re-feeding the output data into the generator depending on the accuracy of the results. This is done through a comparison check of the final result with the desired data. Empirically, we investigated the proposed architecture by testing the discriminator network (D) on data that came from the generator and contained errors. As a result, we found that the D is capable of rejecting any erroneous data. The MINST dataset [58] is applied to represent the simulation scenarios of the proposed architecture. For example, if the output data from the generator is fake and less than the set accuracy threshold (AT) of 80%, the discriminator network automatically sends it back to the generator for further iterations, as shown in Figure 12. Similarly, if data at each iteration is deterministic but the final data results in error and is not real, the network rejects the final data containing errors and feeds the most recent accurate data to the generator until the obtained result is error-free.

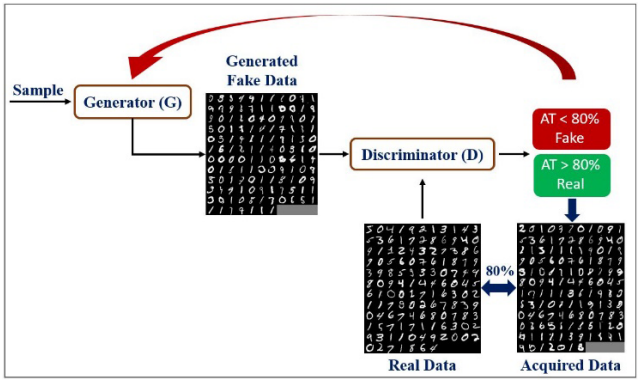

FIGURE 12. Generate Accurate Data Scenario and Detecting Errors for each Iteration.

VII. SIMULATION RESULTS

The proposed network composed of sensor nodes and base station (BS) is distributed randomly with the same power, resources, and computational capabilities. These nodes collect information about an event and embed their data with fake data from the G network before transmitting it to BS. The BS has a higher capacity in terms of power and resources than other sensor nodes within the network.

The main objective of the simulation is to monitor the network and secure data communication from internal and external attacks. Extensive experimental evaluation on this approach ensures that the discriminator network is robustly capable of protecting the network from any attackers or malicious nodes. It improves the security of the network without compromising on the network delay.

A. PERFOMANCE MATRICS

This section evaluates the performance of our proposed GAN-based wireless sensor middleware called SWSNM and compare is with the Eagilla approach [21] on different scenarios such as energy consumption, throughput, and end-to-end delay.

The NS2 simulator is used with similar parameters for both approaches. The size of the network is 1500×1500 m^2 network topology. The network involves 150 sensor nodes with a transmission range of 40 meters. The initial energy of the nodes is set to 6 joules. The maximum energy consumption of the sensor nodes for receiving (Rx) and transmitting (Tx) the data is set to 14 mW and 13.0 mW, respectively. Sensing and idle nodes have 10.2 mW and 0.42 mW, respectively. The maximum simulation time is 45 minutes.

The network consists of 12 mobile nodes and 138 static nodes. We assume that the nodes that drop all packets passing through them are malicious nodes. When it receives an indication of dropped packets, the algorithm assigns a malicious flag to those nodes. The location of each of the malicious node within the network is calculated and those nodes are replaced with static (normal) nodes.

B. ENERGY CONSUMPTION

A comparison of SWSNM approach with and without malicious nodes with Eagilla approach are shown in Figures 13, 14, and 15. Figure 13 shows the average amount of energy consumed by the nodes within the network. It is clearly seen that when the malicious nodes are replaced with new static nodes, the energy consumption of the network is reduced.

In the proposed approach, the energy consumed during data transmission as well as during sleep and idle modes are taken into account. The energy consumption is obtained from equation 12. We assume that the energy consumed by node j has bits of packets to transmit/receive while the node is active. Further, both sleep and idle modes are counted and n represents the total number of nodes in the network.

$$\sum_{j=1}^{n} \frac{\text{Total energy consumed at node}_j}{n} \tag{12}$$

Figure 13 shows the average energy consumption for SWSNM (with and without malicious nodes) and the Eagilla [21] approach. It is seen that removal of the malicious nodes reduces the energy consumption of the network. The energy consumption curve for Eagilla [21] is rather

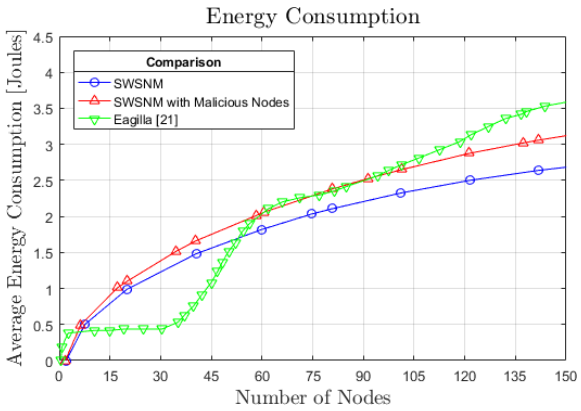

FIGURE 13. Energy consumption vs. number of sensor nodes.

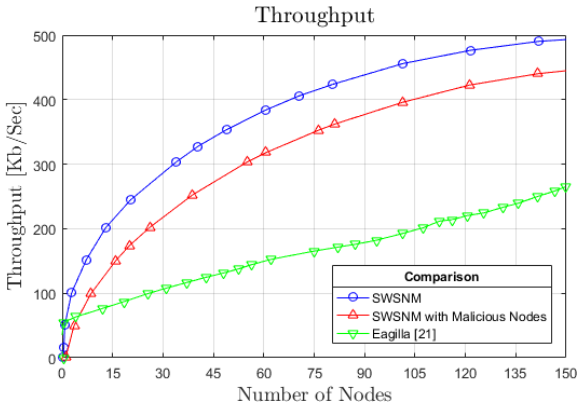

FIGURE 14. Throughput for SWSNM and Eagilla [21] approaches.

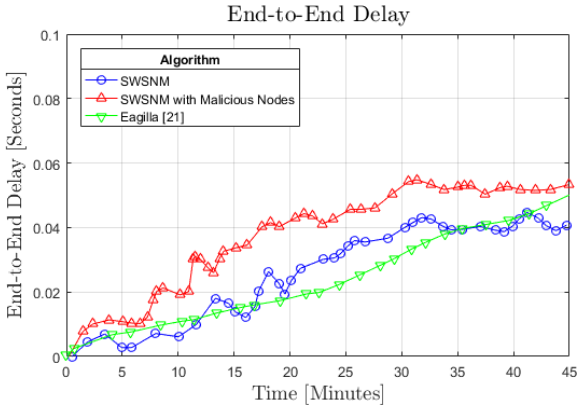

FIGURE 15. End-to-End Delay with simulation time.

interesting. While the energy consumption is much less for small number of nodes (a little over 30 nodes), significant increase in the energy consumption is seen at network size ranging between 30 and 60 nodes. This shows that in terms of the energy consumption, the Eagilla approach [21] is only feasible for small number of nodes.

C. THROUGHPUT

Throughput is defined as the amount of data that is transmitted from source nodes to the destination or base station within a certain time, obtained in equation 13. Figure 13 shows

TABLE 7. Comparison of proposed SWSNM With and without malicious nodes.

	Location/ Time	SWSNM with 10 MN	SWSNM	% Diff.
Energy Consumption	140th Node	3.056 J	2.6 J	**14.9%**
Throughput	140th Node	440.25	490.39	**10.2%**
End-to-End Delay	35 Minutes	0.053	0.04	**24.5%**

TABLE 8. Comparison of proposed SWSNM and Eagilla [21] techniques.

	Location/ Time	Eagilla [21]	SWSNM	% Diff.
Energy Consumption	140th Node	3.6 J	2.6 J	**27.7%**
Throughput	140th Node	250	490.39	**49.0%**
End-to-End Delay	35 Minutes	0.042	0.039	**4.87%**

a comparison of network throughput for each of the three cases. As seen from Figure 14, the throughput of the network without malicious nodes is significantly higher than the network with malicious nodes. The SWSNM outperforms the Eagilla [21] by a significant margin.

$$\text{Throughput} = \frac{\text{Number of bytes received at BS}}{\text{Total bytes transmitted at source nodes}} \tag{13}$$

D. DELAY

The end-to-end delay, obtained in equation 14, is another important parameter to evaluate the performance of the proposed approach. In equation 14, it is noteworthy that the EED is obtained by summing the delays of all the nodes and averaged over total number of nodes. The delay of each node is calculated through equation 15 and normalized by the total number of packets by the given node j. Figure 15 shows that the end-to-end delay increases until a certain time (∼32 minutes) and stays fairly constant after that. It is noteworthy that while the trends are similar, the SWSNM shows significantly lower end-to-end delay when compared to that with the existence of malicious nodes (SWSNM w/10 MN). The end-to-end delay of the Eagilla [21] approach is comparable to the proposed SWSNM.

$$\text{EED} = \frac{\sum_{j=1}^{n} D_j}{n} \tag{14}$$

$$D_j = \frac{\sum_{p=1}^{pkt} D^p_{rec} - D^p_{snd}}{\text{Number of packets by node}_j} \tag{15}$$

The D^i_{rec} represents arrival time at the destination for packet p, D^p_{snd} is transmission time at the source node.

SWSNM comparison of three factors; energy consumption, throughput, and the end-to-end delay, is shown in Tables 7 and 8 for SWSNM with malicious nodes (MN) and Eagilla [21], respectively. For the ease of comparison,

one location or time is selected for each of the variables. Table 7 shows that 14.9% more energy is consumed when the network included malicious nodes compared to that with no malicious nodes. Similarly, more than 10% throughput was increased when all malicious nodes were replaced with a 24.5% lower end-to-end delay. It can be inferred that if the probability of malicious nodes is higher in the network (for example 20%), percentage differences in the calculated variables are expected to be much larger. SWSNM comparison with the Eagilla [21] approach, shown in Table 8, reveals significant percentage differences for energy consumption (27.7%) and throughput (49%) while the end-to-end delay is comparable (4.87%) for both approaches.

VIII. CONCLUSION

Wireless sensor networks (WSNs) are an essential medium for the transmission of data for numerous applications. In order to address power consumption, communication, and security challenges, middleware bridges the gap between applications and WSNs. Most existing middleware does not completely address the issues that significantly impact WSNs' performance. Thus, our paper proposes unsupervised learning for the development of WSNs middleware to provide end-to-end secure system. The proposed algorithm (SWSNM) consists of a generator (G) and a discriminator (D). The generator is capable of creating fake data to confuse the attacker and resolving imbalanced data by generating more data to balance the proportion of classes, the normal and attack data. We render the D to be a powerful network that can easily distinguish between two datasets, even if the fake data is very close to real samples. Extensive testing on the NSL-KDD dataset with different supervised learning techniques and comparisons with our generator network data shows that our generator model provides a better accuracy of 86.5% with a low FPR of 21% and 84% with a lower FPR of 13% by using full (40 features) and reduced (20 features) respectively. Additionally, we employed the t-SNE algorithm for both full features and reduced features to compare the output of our generator to the original dataset. Results show that the proposed generator performs very well with data visualization while the original, conventional dataset NSL-KDD performed worse in both algorithms.

NS2 simulation results demonstrate that the proposed SWSNM provides stronger security mechanism by detecting and replacing malicious nodes which leads to lesser energy consumption, higher throughput, and lesser end-to-end delay. Ongoing work will include estimations of error rates and the lifetime of the network along with comparison of the proposed technique with other contending approaches.

REFERENCES

[1] K. A. Bispo, N. S. Rosa, and P. R. F. Cunha, "SITRUS: Semantic infrastructure for wireless sensor networks," *Sensors*, vol. 15, no. 11, pp. 27436–27469, 2015.

[2] G. Xu, W. Shen, and X. Wang, "Applications of wireless sensor networks in marine environment monitoring: A survey," *Sensors*, vol. 14, no. 9, pp. 16932–16954, 2014.

[3] S. Hadim and N. Mohamed, "Middleware for wireless sensor networks: A survey," in *Proc. 1st Int. Conf. Commun. Syst. Softw. Middleware*, New Delhi, India, Jan. 2006, pp. 1–7.

[4] R. Alshinina and K. Elleithy, "Performance and challenges of service-oriented architecture for wireless sensor networks," *Sensors*, vol. 17, no. 3, p. 536, 2017.

[5] J. Al-Jaroodi and A. Al-Dhaheri, "Security issues of service-oriented middleware," *Int. J. Comput. Sci. Netw. Secur.*, vol. 11, no. 1, pp. 153–160, 2011.

[6] A. Shchzad, H. Q. Ngo, S. Y. Lee, and Y.-K. Lee, "A comprehensive middleware architecture for context-aware ubiquitous computing systems," in *Proc. 4th Annu. ACIS Int. Conf. Comput. Inf. Sci. (ICIS)*, Jeju, South Korea, Jul. 2005, pp. 251–256.

[7] Y. Wang, G. Attebury, and B. Ramamurthy, "A survey of security issues in wireless sensor networks," *IEEE Commun. Surveys Tuts.*, vol. 8, no. 2, pp. 2–23, 2nd Quart., 2006.

[8] Y. W. Law, J. Doumen, and P. Hartel, "Benchmarking block ciphers for wireless sensor networks," in *Proc. IEEE Int. Conf. Mobile Ad-Hoc Sensor Syst.*, Fort Lauderdale, FL, USA, Oct. 2004, pp. 447–456.

[9] J. Newsome, E. Shi, D. Song, and A. Perrig, "The sybil attack in sensor networks: Analysis & defenses," presented at the 3rd Int. Symp. Inf. Process. Sensor Netw., Berkeley, CA, USA, 2004.

[10] A. A. Pirzada and C. McDonald, "Secure routing with the AODV protocol," in *Proc. Asia–Pacific Conf. Commun.*, Perth, WA, Australia, Oct. 2005, pp. 57–61.

[11] S. Bhargava and D. P. Agrawal, "Security enhancements in AODV protocol for wireless ad hoc networks," in *Proc. IEEE VTS 54th Veh. Technol. Conf. (VTC Fall)*, Atlantic City, NJ, USA, vol. 4, Oct. 2001, pp. 2143–2147.

[12] S. Capkun and J.-P. Hubaux, "Secure positioning of wireless devices with application to sensor networks," in *Proc. 24th Annu. Joint Conf. IEEE Comput. Commun. Soc. (INFOCOM)*, Miami, FL, USA, vol. 3, Mar. 2005, pp. 1917–1928.

[13] P. Aggarwal and S. K. Sharma, "Analysis of KDD dataset attributes—Class wise for intrusion detection," *Procedia Comput. Sci.*, vol. 57, pp. 842–851, Mar. 2015.

[14] J. A. Jeyanna, E. Indumathi, and D. S. Punithavathani, "A network intrusion detection system using clustering and outlier detection," *Int. J. Innov. Res. Comput. Commun. Eng. (IJIRCCE)*, vol. 3, no. 2, pp. 975–982, 2015.

[15] E. Shi and A. Perrig, "Designing secure sensor networks," *IEEE Wireless Commun.*, vol. 11, no. 6, pp. 38–43, Dec. 2004.

[16] X. Chen, K. Makki, K. Yen, and N. Pissinou, "Sensor network security: A survey," *IEEE Commun. Surveys Tuts.*, vol. 11, no. 2, pp. 52–73, 2nd Quart., 2009.

[17] S. Roy, N. Maitra, J. Nath, S. Agarwal, and A. Nath, "Ultra encryption standard modified (UES) version-I: Symmetric key cryptosystem with multiple encryption and randomized vernam key using generalized modified vernam cipher method, permutation method, and columnar transposition method," in *Proc. IEEE Sponsored Nat. Conf. Recent Adv. Commun., Control Comput. Technol. (RACCCT)*, 2012, pp. 29–30.

[18] A. Swaminathan, B. S. Krishnan and M. Ramaswamy, "A novel security enhancement strategy for improving the concert of CDMA based mobile ad-hoc network," in *Proc. Int. J. Mod. Electron. Commun. Eng. (IJMECE)*, Jan. 2017, pp. 1–8.

[19] A. Kaur and S. S. Kang, "Attacks in wireless sensor network—A review," *Int. J. Comput. Sci. Eng.*, vol. 4, no. 5, pp. 157–162, May 2016.

[20] R. D. Shinganjude and D. P. Theng, "Inspecting the ways of source anonymity in wireless sensor network," in *Proc. 4th Int. Conf. Commun. Syst. Netw. Technol. (CSNT)*, 2014, pp. 705–707.

[21] K. Lingaraj, R. V. Biradar, and V. C. Patil, "Eagilla: An enhanced mobile agent middleware for wireless sensor networks," *Alexandria Eng. J.*, to be published, doi: 10.1016/j.aej.2017.03.003.

[22] L. Capra, "MaLM: Machine learning middleware to tackle ontology heterogeneity," in *Proc. 5th Annu. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PerCom Workshops)*, White Plains, NY, USA, Mar. 2007, pp. 449–454.

[23] T. Avram, S. Oh, and S. Hariri, "Analyzing attacks in wireless ad hoc network with self-organizing maps," in *Proc. 5th Annu. Conf. Commun. Netw. Services Res. (CNSR)*, Frederlcton, NB, Canada, May 2007, pp. 166–175.

[24] M. A. Alsheikh, S. Lin, D. Niyato, and H. P. Tan, "Machine learning in wireless sensor networks: Algorithms, strategies, and applications," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 1996–2018, 4th Quart., 2014.

[25] M. Ribeiro, K. Grolinger, and M. A. M. Capretz, "MLaaS: Machine learning as a service," in *Proc. IEEE 14th Int. Conf. Mach. Learn. Appl. (ICMLA)*, Miami, FL, USA, Dec. 2015, pp. 896–902.

[26] R. Husain and D. R. Vohra, "A survey on machine learning in wireless sensor networks," *Int. Edu. And Res. J.*, vol. 3, no. 1, pp. 17–18, 2017.
[27] N. Ahad, J. Qadir, and N. Ahsan, "Neural networks in wireless networks: Techniques, applications and guidelines," *J. Netw. Comput. Appl.*, vol. 68, pp. 1–27, Jun. 2016.
[28] Y. Zhang, N. Meratnia, and P. Havinga, "Outlier detection techniques for wireless sensor networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 12, no. 2, pp. 159–170, 2nd Quart., 2010.
[29] U. Mehboob, J. Qadir, S. Ali, and A. Vasilakos, "Genetic algorithms in wireless networking: Techniques, applications, and issues," *Soft Comput.*, vol. 20, no. 6, pp. 2467–2501, 2016.
[30] J.-P. Vasseur, G. Mermoud, and S. Dasgupta, "Predictive learning machine-based approach to detect traffic outside of service level agreements," Google Patents 9 338 065, May 10, 2016.
[31] D. Janakiram, V. A. Reddy, and A. V. U. P. Kumar, "Outlier detection in wireless sensor networks using Bayesian belief networks," in *Proc. 1st Int. Conf. Commun. Syst. Softw. Middleware*, New Delhi, India, Jan. 2006, pp. 1–6.
[32] J. W. Branch, B. Szymanski, C. Giannella, W. Ran, and H. Kargupta, "In-network outlier detection in wireless sensor networks," in *Proc. 26th IEEE Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Lisboa, Portugal, 2006, p. 51.
[33] S. Kaplantzis, A. Shilton, N. Mani, and Y. A. Sekercioglu, "Detecting selective forwarding attacks in wireless sensor networks using support vector machines," in *Proc. 3rd Int. Conf. Intell. Sensors, Sensor Netw. Inf.*, Melbourne, QLD, Australia, Dec. 2007, pp. 335–340.
[34] S. Rajasegarar, C. Leckie, M. Palaniswami, and J. C. Bezdek, "Quarter sphere based distributed anomaly detection in wireless sensor networks," in *Proc. IEEE Int. Conf. Commun.*, Glasgow, U.K., Jun. 2007, pp. 3864–3869.
[35] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is 'nearest neighbor' meaningful?" in *Proc. 7th Int. Conf. Database Theory (ICDT)*, Jerusalem, Israel, C. Beeri and P. Buneman, Eds. Berlin, Germany: Springer, 1999, pp. 217–235.
[36] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.
[37] T. Salimans *et al.*, "Improved techniques for training GANs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2234–2242.
[38] J. T. Springenberg. (2015). "Unsupervised and semi-supervised learning with categorical generative adversarial networks." [Online]. Available: https://arxiv.org/abs/1511.06390
[39] University of New Brunswick. *NSL-KDD*. [Online]. Available: http://nsl.cs.unb.ca/nsl-kdd/
[40] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl.*, Ottawa, ON, Canada, Jul. 2009, pp. 1–6.
[41] S. Hettich and S. Bay, "The UCI KDD archive," Dept. Inf. Comput. Sci., Univ. California, Irvine, Irvine, CA, USA, Tech. Rep., 1999, vol. 152. [Online]. Available: http://kdd.ics.uci.edu
[42] University of California, Irvine. (1999). *KDD Cup 1999*. [Online]. Available: http://Kdd.Ics.Uci.Edu/Databases/Kddcup99.html
[43] L. Ray, "Determining the number of hidden neurons in a multi layer feed forward neural network," *J. Inf. Secur. Res.*, vol. 4, no. 2, pp. 63–70, 2013.
[44] F. Chollet. (2015). *keras*. [Online]. Available: https://github.com/fchollet/keras
[45] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.
[46] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.
[47] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 5353–5360.
[48] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2377–2385.
[49] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778.
[50] S. Albelwi and A. Mahmood, "A framework for designing the architectures of deep convolutional neural networks," *Entropy*, vol. 19, no. 6, p. 242, 2017.
[51] B. Ingre and A. Yadav, "Performance analysis of NSL-KDD dataset using ANN," in *Proc. Int. Conf. Signal Process. Commun. Eng. Syst.*, Guntur, India, Jun. 2015, pp. 92–96.
[52] M. Panda, A. Abraham, and M. R. Patra, "Discriminative multinomial Naïve Bayes for network intrusion detection," in *Proc. 6th Int. Conf. Inf. Assurance Secur.*, Atlanta, GA, USA, Aug. 2010, pp. 5–10.
[53] L. M. Ibrahim, D. T. Basheer, and M. S. Mahmod, "A comparison study for intrusion database (KDD99, NSL-KDD) based on self organization map (SOM) artificial neural network," *J. Eng. Sci. Technol.*, vol. 8, no. 1, pp. 107–119, 2013.
[54] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.
[55] S. S. Singh and Y. B. Jinila, "Sensor node failure detection using check point recovery algorithm," in *Proc. Int. Conf. Recent Trends Inf. Technol. (ICRTIT)*, Chennai, India, Apr. 2016, pp. 1–4.
[56] G. Sumalatha, N. Zareena, and C. G. Raju. (2014). "A review on failure node recovery algorithms in wireless sensor actor networks." [Online]. Available: https://arxiv.org/abs/1407.0009
[57] P. Jiang, "A new method for node fault detection in wireless sensor networks," *Sensors*, vol. 9, no. 2, pp. 1282–1294, 2009.
[58] Y. LeCun. (1998). *The MNIST Database of Handwritten Digits*. [Online]. Available: http://yann.lecun.com/exdb/mnist/

REMAH A. ALSHININA received the bachelor's degree from King Abdulaziz University, Saudi Arabia, in 2006, and the master's degree in information technology from Southern New Hampshire University in 2012. She is currently pursuing the Ph.D. degree in computer science and engineering with the University of Bridgeport. She has presented and published papers in national/international conferences and journals. Her research interests are WSNs, machine learning, and network security.

KHALED M. ELLEITHY received the B.Sc. degree in computer science and automatic control and the M.S. degree in computer networks from Alexandria University in 1983 and 1986, respectively, and the M.S. and Ph.D. degrees in computer science from the Center for Advanced Computer Studies, University of Louisiana at Lafayette, in 1988 and 1990, respectively. He is currently the Associate Vice President for graduate studies and research with the University of Bridgeport. He is also a Professor of computer science and engineering. He supervised hundreds of senior projects, M.S. theses, and Ph.D. dissertations. He developed and introduced many new undergraduate/graduate courses. He also developed new teaching/research laboratories in his area of expertise. He has authored over 350 research papers in national/international journals and conferences in his areas of expertise. He is an editor or co-editor for 12 books published by Springer. His research interests include wireless sensor networks, mobile communications, network security, quantum computing, and formal approaches for design and verification. He has been a member of the ACM since 1990, a member of the ACM Special Interest Group on Computer Architecture since 1990, a member of the Honor Society of the Phi Kappa Phi University of South Western Louisiana Chapter since 1989, a member of the IEEE Circuits and Systems Society since 1988, a member of the IEEE Computer Society since 1988, and a Lifetime Member of the Egyptian Engineering Syndicate since 1983. He is a member of the technical program committees of many international conferences as recognition of his research qualifications. He is a member of several technical and honorary societies. He is a Senior Member of the IEEE Computer Society. He was a recipient of the Distinguished Professor of the Year at the University of Bridgeport for academic year 2006–2007. His students received over twenty prestigious national/international awards from the IEEE, the ACM, and the ASEE. He was the Chair Person of the International Conference on Industrial Electronics, Technology, and Automation. He was the Co-Chair and the Co-Founder of the Annual International Joint Conferences on Computer, Information, and Systems Sciences, and Engineering virtual conferences 2005–2014. He served as a guest editor for several international journals.