IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 9, 2019, accepted March 24, 2019, date of publication March 28, 2019, date of current version April 12, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2907965

Toward a Lightweight Intrusion Detection System for the Internet of Things

SANA ULLAH JAN, (Member, IEEE), SAEED AHMED, (Member, IEEE), VLADIMIR SHAKHOV, (Member, IEEE), AND INSOO KOO, (Member, IEEE)
Department of Electrical/Electronic and Computer Engineering, University of Ulsan, Ulsan 44610, South Korea

Corresponding author: Insoo Koo (iskoo@ulsan.ac.kr)

This work was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant NRF-2017R1D1A3B03030386.

ABSTRACT Integration of the Internet into the entities of the different domains of human society (such as smart homes, health care, smart grids, manufacturing processes, product supply chains, and environmental monitoring) is emerging as a new paradigm called the Internet of Things (IoT). However, the ubiquitous and wide-range IoT networks make them prone to cyberattacks. One of the main types of attack is a denial of service (DoS), where the attacker floods the network with a large volume of data to prevent nodes from using the services. An intrusion detection mechanism is considered a chief source of protection for information and communications technology. However, conventional intrusion detection methods need to be modified and improved for application to the IoT owing to certain limitations, such as resource-constrained devices, the limited memory and battery capacity of nodes, and specific protocol stacks. In this paper, we develop a lightweight attack detection strategy utilizing a supervised machine learning-based support vector machine (SVM) to detect an adversary attempting to inject unnecessary data into the IoT network. The simulation results show that the proposed SVM-based classifier, aided by a combination of two or three incomplex features, can perform satisfactorily in terms of classification accuracy and detection time.

INDEX TERMS Intrusion detection system, anomaly detection, Internet of Things, support vector machine.

I. INTRODUCTION

The concept of Internet of Things (IoT) is based on the integration of uniquely identifiable heterogeneous physical objects around us (humans, animals, sensors, instant cameras, vehicles etc.) and the cyber world with the ability to transfer data over a network without requiring human-to-human or human-to-computer interfaces. As illustrated in Figure 1, the applications of the IoT may range from a simple appliance for a smart home to a complex apparatus in a smart grid. The IoT provides a tremendous opportunity for societies around the world. Even with different objectives, contrasting IoT applications have an intersection set of characteristics. Broadly speaking, a primary node in IoT has capability to perform three distinct actions; *data collection*, *data transmission*, and *data processing and utilization* [1]–[3].

In the data collection stage, small, memory-constrained and low energy-consumption sensors with a short-range communications capability are employed to collect information about the physical environment. Ethernet, WiFi, ZigBee, and wire-based technologies are combined with Transmission Control Protocol/Internet Protocol to connect the objects and users across prolonged distances during data transmission. During the data processing and utilization stage, applications process the data to obtain useful information, and may initiate control commands to act on the physical environment after making decisions based on the collected information. The coordination of diverse technologies, the heterogeneity, and the distributed nature of communications technologies proposed for the IoT by different standards development organizations [4] magnify the threat to end-to-end security in IoT applications.

Numerous methods for improving data confidentiality, authentication, and access have been reported in the literature; however, even with these mechanisms, IoT networks are prone to multiple attacks aimed at disrupting the network. The growth, complexity, ubiquity, and diversity of the IoT expands the potential attack surface.

The associate editor coordinating the review of this manuscript and approving it for publication was Kashif Saleem.

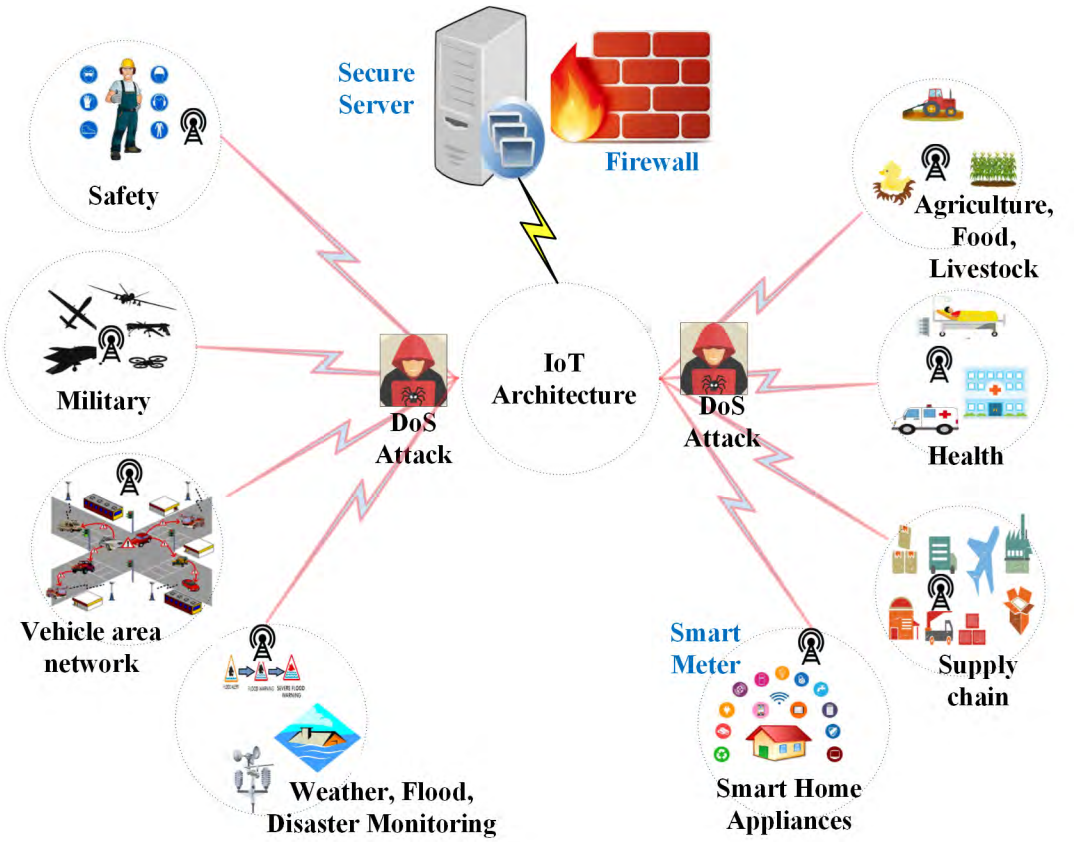

FIGURE 1. An example of IoT applications.

Therefore, intrusion prevention tools and signature-based intrusion detection methods cannot be effective against modified attacks, and fundamentally new types of attacks, in the IoT. A defense mechanism aiming to detect novel and potential intrusions is required. Intrusion detection systems (IDSs) based on anomaly detection (a.k.a. statistically based) fulfill this purpose [5]. Anomaly detection does not require prior identification of attack signatures.

Considering that the development of IDSs for the IoT represents a significant challenge for information security, researchers describe IoT networks in terms of specific characteristics as follows [1], [6].

- Unlike traditional networks, where the system administrator deploys IDS agents in network entities with high computing and storage capacities, the memory capacity, processing power, and battery energy-capacity constraints of IoT network nodes that host IDS agents is challenging.
- In conventional networks, end systems are directly connected to specific nodes (e.g., wireless access points, switches, and routers) that are responsible for forwarding packets to the destination. In contrast, there are multiple hops in IoT networks. Regular nodes may simultaneously forward packets and work as end systems. Moreover, in some IoT applications, the network topology regularly changes (e.g., VANETs, mobile sinks, dynamic selection of cluster heads). The specificity of the topology poses new challenges for IDSs.
- Protocols used in IoT networks are different from conventional networks, such as IEEE 802.15.4, IPv6 over Low-power Wireless Personal Area Network (6LoWPAN), IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) and Constrained Application Protocol (CoAP). Heterogeneity in protocols introduces new weaknesses which result in new challenges for IDSs in the IoT.

The characteristics of IoT networks given above limit the design of an IDS to be lightweight still efficient enough to secure the network from potential attacks. The term lightweight does not refer to simplicity of the system. It means that the IDS should be able to perform its operation with the available amount of resources in the sensor nodes of the network. Roesch *et al.* [7] defined a lightweight IDS as small, powerful, and flexible enough to be used as permanent element of the network security infrastructure. According to Hai *et al.* [8], a lightweight system aims at energy saving and reduced computational resources. Maleh and Ezzati [9] consider a system as lightweight if it has reduced energy consumption. Concluding, a lightweight system is the one which can perform in limited energy and computation resources regardless of simplicity. Keeping these definitions in mind, we design a lightweight IDS system by avoiding the complex

features extraction and feature selection steps. We provide rather uncomplicated and limited number of features to be extracted from raw data. The results show that, this type of system can perform efficiently in discriminating an intrusion in the IoT network.

The proposed scheme is intuitive to perform well in this type of application. However, there is no literature addressing the intrusion detection problem using such a foolproof algorithm. This leaves gap for proposing and analyzing such uncomplicated and intuitive algorithms for these applications rather than utilizing complex statistical techniques.

There are many datasets available providing samples DoS attacks under different scenarios including KDD'99, DARPA, CAIDA DDoS, etc. [10]. The issue to use these datasets for evaluating our proposed algorithm lies in the attributes in which these datasets are available. The only characteristic of network traffic that is used in our algorithm is the packet arrival rate per node. However, this characteristic of the data is not given in any of the above given dataset. For instance, the dataset KDD'99 represent the data samples in attributes such as protocol type, duration of connection, Land etc. but packet arrival rate [11]. This limits our algorithm to be tested utilizing these online available datasets.

A. MOTIVATION

The novelty of this manuscript lies in the design of an IDS for IoT networks with the characteristics of lightweight i.e., minimizing cost of system in terms of energy consumption and computational resources. An Ideal IDS system is lightweight enough to be implemented in a sensor node equipped with limited battery capacity and computational resources, still performing efficiently. In this work, this is achieved by eliminating the complex features extraction from data and the feature selection steps. Instead of taking different characteristics the proposed IDS rely on only the packet arrival rate attribute of raw data. Moreover, the complexity of an SVM-based classifier directly depends on the dimensions of input vector. The higher the dimensions of input vector the higher the complexity of SVM. Keeping this in mind, we reduce the dimensions by extracting only 2 to 3 features from input vector. In short, we try to develop a lightweight IDS by the following way. We considering only one attribute, i.e., the packet transmission rate, and extract only 2 to 3 features from that attribute. The three features utilized include mean, median and maximum values obtained to perform the classification. Intuitively, these steps reduce the energy and computational cost as compared to a system considering up to 40 complex attributes, such as protocol type, service, land, wrong fragments etc. as given in NSL-KDD dataset [11]. This approach makes the proposed IDS suitable for implementation in sensor nodes of IoT while keeping the efficiency of system satisfactory as illustrated in the experimental results.

Furthermore, several researchers have proved that an SVM-based classifier outperform neural networks, k-nearest neighbor, random forest etc. [12]–[14]. This is the motivation to design our proposed algorithm based on an SVM-based classifier. A performance comparison of SVM-based classifier and other machine learning-based algorithms can also be found in literature supporting this argument.

B. CONTRIBUTIONS

Thus, to unlock the IoT potential, we need to improve IoT security and the performance of IDS. In this paper, we are motivated to consider intrusions (and corresponding anomaly-based IDS) accompanied by changes in traffic intensity. This effect is typical for a wide range of attacks in the IoT environment. The main contributions of this paper are as follows.

- We analyze DoS attacks in the IoT that were reported in the literature [15]–[20] and conclude that the consequences of the intrusions include changes in the intensity of the transmitted packets. In some cases, the change in traffic intensity is an attack tool; in other cases, it is a concomitant effect. Analysis reveals the relationship between traffic change profiles and types of intrusion.
- An intrusion detection on a sensor-by-sensor basis is a challenging problem. At the same time, there is an industrial demand on intrusion detection in devices [21]. In some recent papers it has been declared a low quality of SVM based intrusion detection on a sensor-by-sensor basis [22]. However, in this paper we demonstrate that a foolproof SVM based approach combined with proper statistics and feature engineering provides good performance in various scenarios.
- Instead of utilizing complex attributes (given in online datasets such as NSL-KDD) of the system, we utilize only one attribute, the packet arrival rate to the sensor node. To the best of authors' knowledge, this work is pioneer considering specifically this attribute for developing an IDS for IoT.
- Based on the above analysis, we develop a support vector machine (SVM)-based classifier for a lightweight IDS. The performance of classifier is analyzed for linear, polynomial, and radial-basis kernel functions.
- Simulation experiments are conducted to verify the choice of SVM parameters and to demonstrate the method's efficiency. The performance of IDS is analyzed in terms of true positive rate, true negative rate, false positive rate, false negative rate, accuracy and detection time.
- Furthermore, the performance of proposed SVM-based IDS is compared with other machine learning algorithms-based IDS including neural network, KNN and decision tree. The accuracies comparison of 100 iterations of experiments prove the efficiency SVM-based classifier using linear and polynomial kernel functions.
- Finally, the performance of proposed algorithm is compared with some of the proposed algorithms in literature. The accuracy measure is used to assess how efficiently an algorithm can detect the intrusions. The CPU time measure is used to compare the lightweightness measure of different algorithms. The results show that

the proposed algorithm is not only lightweight among the given set of algorithms but it also outperforms these algorithms.

C. PAPER ORGANIZATION

The rest of this paper is organized as follows. Section II introduces the related work. An analysis of IoT attacks is presented in Section III, and we discuss the models of trustworthy activity and abnormal behavior. Section IV proposes the machine learning-based framework of the intrusion detection system, followed by a performance evaluation in Section V. The conclusion and future research directions are presented in Section VI. The abbreviations used in this paper, are summarized in Table 1.

II. RELATED WORKS

Farahnakian and Heikkonen [20] proposed a deep auto-encoder (DAE)-based classification model to detect and classify intrusions in the network. The proposed classifier was obtained by combining a set of auto-encoders (AE) such that the output of an AE at $(n-1)th$ layer is the input to the AE at the *nth* layer. The results presented showed that the proposed DAE performed better in detecting intrusions in the systems, compared to the deep belief network (DBN) and a combination of AE+DBN. Shone *et al.* [23] stepped forward and proposed a nonsymmetric deep auto-encoder (NDAE) to learn features in an unsupervised manner. A set of NDAEs is stacked to perform the learning and classification tasks.

A malicious pattern detection mechanism was proposed by Oh *et al.* [24] to secure networks in the IoT. They reduced memory usage in a pattern-matching process by proposing an auxiliary shifting method and an early decision scheme. They reported efficient results in early detection of a malicious pattern; however, they failed to detect and classify other attacks, such as denial of service (DoS), false data injection, etc. Furthermore, an attacker may try a unique pattern each time, making it difficult for the node to detect an attack. Moreover, Ali *et al.* [25] proposed a fast learning network (FLN) with particle swarm optimization (PSO) applied for convergence of classifier parameters. Although the results were satisfactory, the complexity of the system is too high to be applied to sensor nodes due to their low computation and energy-storage capabilities. Moukhafi *et al.* [26] combined a hybrid genetic algorithm (GA) and an SVM with PSO for feature subset selection in their proposed intrusion detection system. This system was successful in differentiating DoS attacks from other types of attack with an accuracy of almost 100%; however, it could not discriminate normal class signals from other types of attacks with reasonable accuracy. Vajayanand *et al.* [27] tried to improve classification accuracy by proposing a hybrid feature-selection technique based on a GA and mutual information (MI) for an SVM-based classifier. They also proved (by illustrating their experimental results) that an SVM-based classifier is successful in achieving better performance than an artificial neural network (ANN). The highest accuracy achieved in their experiment was 96% when the classifier was trained with 400 samples. The results showed that utilizing both the GA and MI could need as few as three informative features to obtain these results. However, considering the battery and computation-cost limitations of IoT devices, this scheme does not seem like a promising solution.

Recently, Kabir *et al.* [28] proposed an optimum allocation-based least square SVM (OA-LS-SVM) for intrusion detection systems. This technique first combines the training and testing datasets. Then, an optimal allocation (OA) scheme determines the volume of training and testing sets. Later, it selects representative samples directly from training and testing datasets for the classifier. Although this paper presented some interestingly satisfactory results, it can miss some important information or features in the dataset owing to their limiting the training dataset to samples having a specific relation with a representative sample. Furthermore, obtaining all the samples from training and testing datasets is a challenge difficult to overcome.

Another IDS was proposed in [6] based on an automata or finite state machine. The automata transitions are used to characterize the network, and are later used to detect if an intrusion occurred. The experimental results addressed only three types of attack: the jamming attack, the false attack, and the reply attack. DoS or false data injection attacks were not addressed. A two-step technique to effectively detect intrusions was proposed in [29]. In the first step, several binary classifiers are utilized to classify the sample. In the next step, the sample is classified by a k-nearest neighbors (k-NN) algorithm if the output of step 1 is ambiguous. The highest accuracy obtained in these experiments was around 94% at the cost of high computation resources usage and high energy consumption to implement the several types of classifier. Furthermore, Tao *et al.* [30] proposed an IDS based on feature selection, weight, and parameter optimization of an SVM based on a GA (shortened to FWP-SVM-GA). The GA first selects the features subset, and then simultaneously optimizes the parameters of the SVM. Finally, the trained classifier is used to detect and classify anomalies in the network. In [31] the authors proposed an intrusion detection system based on a conditional variational autoencoder (CVAE). The labels of the samples are added as extra input to the decoder block of a VAE. An IDS based on long short-term memory recurrent neural networks (LSTM-RNNs) was proposed [32]. The authors showed that their proposed technique can successfully overcome various machine learning techniques, including SVM, k-NN, and Bayesian. Khalvati *et al.* [33] used both SVM and naïve Bayes classifiers to efficiently detect and classify intrusions in a network. Han *et al.* [34] focused on the energy efficiency of the system, yet achieved satisfactory performance with the proposed IDS based on game theory and an autoregressive model. In [35], the authors used naïve Bayes, SVM, and a random forest decision tree algorithm to detect DoS attacks in wireless sensor networks (WSNs). Ozay *et al.* [36]

TABLE 1. Nomenclature.

Abbreviation	Term	Abbreviation	Term
ACC	accuracy	k-NN	k-nearest neighbors
AE	auto encoder	LLN	low-power and lossy networks
ANN	artificial neural network	LoPAN	low power personal area networks
CVAE	conditional variational auto encoder	LSTM	long short-term memory
CoAP	constrained application protocol	LS	least square
CPU	central processing unit	MI	mutual information
DAE	deep auto encoder	NDAE	non-symmetric deep auto encoder
DoS	denial of service	NN	neural network
DT	decision tree algorithm	OA	optimum allocation
DBN	deep belief network	PSO	particle swarm optimization
DDoS	distributed denial of service	RBF	radial basis function
FLN	fast learning network	RNN	recurrent neural networks
FDR	false detection rate	RPL	Routing protocol for LLN
FS	feature selection	SVM	support vector machines
FP	false positive	TDR	true detection rate
FN	false negative	TPR	true positive rate
FPR	false positive rate	TP	true positive
GA	genetic algorithm	τ	detection time
IoT	Internet of Things	WSNs	wireless sensor networks
IDS	intrusion detection system		

presented an analysis of supervised and unsupervised machine learning algorithms to detect attacks targeting smart grids. James *et al.* [37] targeted the detection of false data injection attacks using a wavelet transform and a deep neural network. Several other authors focused on false data injection attacks and proposed detection systems based on deep learning techniques [38]–[40]. Teng *et al.* [41] proposed a collaborative intrusion detection system using an SVM and decision tree algorithms.

Helmer *et al.* [42] presented a distributed agent to implement a lightweight IDS. The reserved mobile agents would move in the network to collect and report to a mediator to obtain the status of the network. Although, this type of architecture is successful to overcome the critical problems of centralized network such as computing cycles on user's computer and providing higher ease of adding nodes to the network. However, this type of network can have some critical issues of higher delay due to the transmission of data to other nodes and then waiting for the response from mediator. Furthermore, these transmissions between nodes makes this network more prone to cyber-attacks. Another attempt to develop a lightweight IDS was done by Li *et al.* [43]. This algorithm includes feature selection phase as the first step of training a linear SVM-based classifier. Although, this algorithm performs satisfactory results when used to classify an online sample, this scheme relies on a random mutation hill climbing (RMHC) optimization technique to select the set of best features. Furthermore, the time taken by feature selection schemes to obtain final set of features increases with increase in the number of features in dataset. Moreover, it is possible that the feature selection strategy may select features which are not capable of discriminating new types of intrusions.

Although all these proposed methods were able to perform efficiently in detecting intrusions in the network, these techniques rely on resource-intensive computing and may be too heavy for the low-capacity nodes of IoT networks. It is necessary to design an IDS requiring low computational costs, miminal energy, and little memory in the network nodes [1].

III. ANALYSIS OF IOT THREATS

A. TYPICAL ATTACKS IN IOT AND CONCOMITANT EFFECT

Considering the specific characteristics of IoT networks, an adversary can launch attacks to disrupt the system in many ways. In this paper, we consider the typical attacks reported in the literature [15]–[17], [44], [45]. Remark, there are several projects on, and standardization initiatives for, WSNs, which may eventually converge with the Internet of Things (IoT), for example European Union projects of Internet of Things Architecture (IoT-A) have been addressing the challenges of IoT solutions development from the WSNs perspective. A brief description of the typical attacks follows. A brief description of these attacks follows.

- *Packets Flooding:* In a wide range of attacks, an intruder can generate a storm of spoofed packets or repeatedly duplicated legal packets. This results in the channels being overloaded, network node buffers overflowing, and in some cases, the goal of the intruder can be the depletion of a network node battery (a vampire attack). However, in all cases, the attack obviously increases traffic intensity.
- *Vulnerability Attacks:* During a vulnerability attack, some malformed packets are sent to the target to mislead a protocol or an application running under it. It leads to degradation of device functionality, and therefore, data transmission intensity is degraded as well.
- *Blackhole Attack:* A malicious node can attract all the packets by requesting a fresh, misleading route to the destination. Then, it accepts them without forwarding them to the destination.
- *Jamming:* An intruder transmits a signal and jams network working frequencies in a way that decreases the

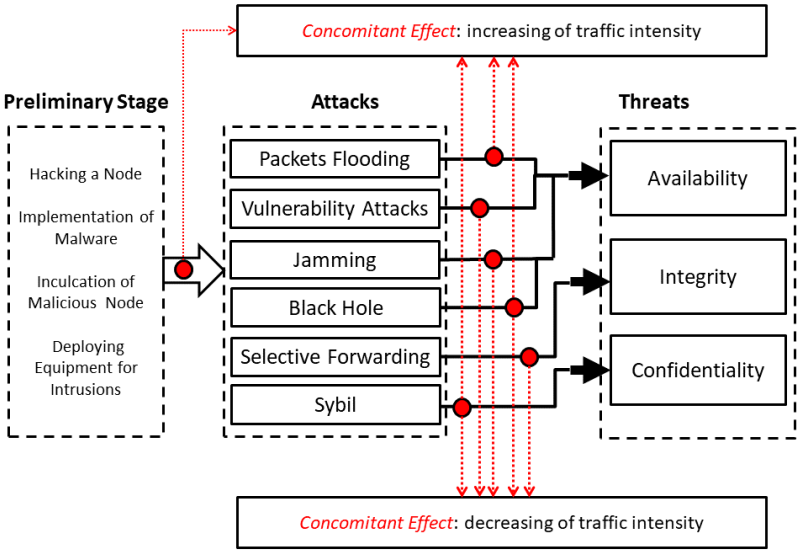

FIGURE 2. Typical IoT Attacks and Concomitant Effects.

signal-to-noise ratio to a level where the nodes of the wireless network can no longer function. As a result of the attack, a group of nodes becomes isolated and does not generate traffic. Therefore, the attack decreases traffic.

- *Selective Forwarding:* An intruder drops part of the packets, which leads to information integrity degradation. If an intruder does not replace the legal packets with spoofed packets, that quickly unmasks the intrusion, and then, traffic intensity decreases.
- *Sybil Attack:* A node in an IoT network is compromised by adversaries in such a way that it depicts itself with false identities to other nodes. Depending on an intruder's goal, the attack can lead to two scenarios of traffic change. Sham nodes can generate additional traffic, or inhibit the traffic of legal nodes.
- *Sinkhole Attack:* In this scenario, a compromised node tries to attract network traffic by advertising false routing information. Subsequently, it can be used to initiate other attacks, like selective forwarding, acknowledge spoofing, altering packets or dropping them etc.
- *Clone Attack:* In this situation, adversaries acquire the secret information of nodes and create duplicates of this information in the whole network to mislead data packets. These kinds of attack are very dangerous to wireless sensor networks. Cloned nodes can launch a variety of attacks: blackhole, inject false data etc.
- *Wormhole Attack:* The adversary can attract and avoid a huge amount of network data by creating a tunnel between two distant nodes in an IoT network. This attack is generally used in conjunction with eavesdropping or selective forwarding.
- *Hello, Flood Attack:* In the network, each new node sends "Hello" messages to discover its neighbor nodes. Also, it broadcasts its route to the base station. Other nodes may choose to route data through this new node if the path is shorter. If a malicious node equipped with a power transmitter sends a "Hello" message with attractive conditions, then a lot of nodes choose it for data transmission. However, the packets of these nodes will never be retransmitted. Therefore, the attack decreases general intensity.

Thus, the typical attacks in the IoT are accompanied by changes in traffic intensity. As the result of some attacks, the intensity grows; in others, it declines. There are some cases where the same attack leads to traffic increasing in one location yet decreasing in another. In the preliminary stages of the attack, an intruder usually explores the network looking for vulnerabilities, which can be accompanied by an increase in traffic. The relationships of typical IoT attacks to traffic change are shown in Figure 2. The concomitant effect can be inherent in all components of the CIA triad.

DoS attacks, especially distributed DoS (DDoS) are serious problems in the IoT, which have been inherited from traditional IP networks. An efficient protection against these types of attack does not exist yet; for example, the biggest attack ever, recorded in 2016, left hundreds of thousands of connected devices infected [18]. In the IoT, the situation becomes worse due to the limited resources of IoT devices. In the preliminary stages of an attack, an intruder can generate some traffic to explore a network and identify system bottlenecks. Moreover, fundamentally new attacks on the IoT lead to traffic change, as well [19].

Thus, observations of traffic intensity can be used for an IDS. To design an IDS for the IoT, the representative characteristics of low computing power, limited memory capacity, and constrained energy capacity in the nodes should be taken into account. In this paper, we investigate a foolproof and

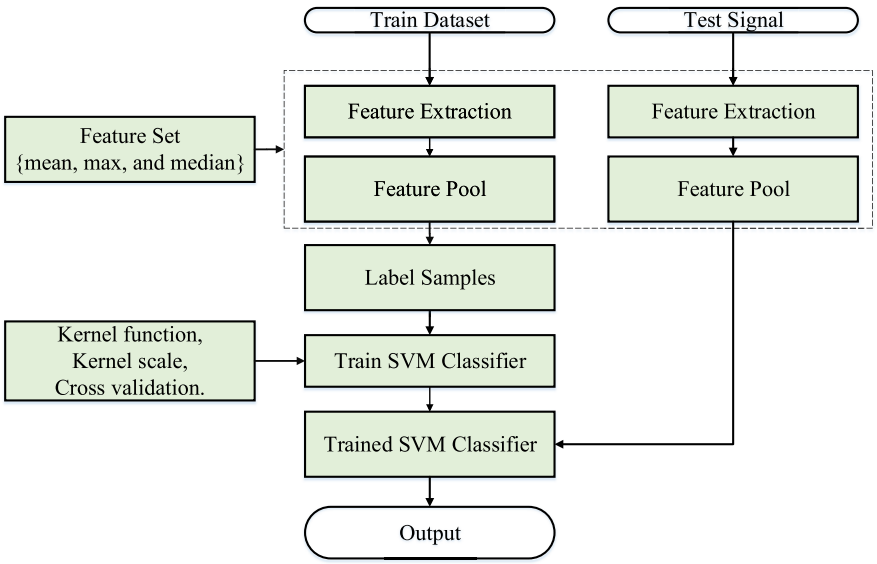

FIGURE 3. Framework of the proposed support vector machine-based intrusion detection system.

lightweight IDS based on an SVM. We show in a series of experiments that by extracting only two or three features from an input sample, the SVM can achieve satisfactory results detecting attacks against the network in a timely manner.

B. IOT TRAFFIC MODELING

Generally, to analyze network behavior, a mathematical model of the traffic is used. Real traffic records are only available in particular cases. The models are based on simplifying assumptions; however, some often provide a basis for adequate approximations of network behavior, as well as worthwhile insights. The scientific and engineering community has accepted the following fact [46]–[53]: the Poisson process (i.e. exponentially distributed times between packet arrivals) is appropriate for traffic modeling in the IoT as well as in WSNs, which are considered an essential part of the IoT [54].

Here, we consider conditions for Poisson process derivation. Let $N(t)$ be the number of packets that have arrived in the time interval $(0, t)$, and let λ be a positive constant. Let us formulate the following four conditions:

1) $N(0) = 0$.
2) Packet arrivals in non-overlapping time intervals are mutually independent.
3) The probability reflecting the number of packet arrivals in the interval $(t, t + h)$ depends only on length h and not on time origin t.
4) For a sufficiently small h, we get equations for the probabilities as follows:

$$\begin{aligned} P\left[N\left(t+h\right) - N\left(t\right) = 1\right] &= \lambda h + o\left(h\right) \\ P\left[N\left(t+h\right) - N\left(t\right) = 0\right] &= 1 - \lambda h + o\left(h\right) \\ P\left[N\left(t+h\right) - N\left(t\right) > 1\right] &= o\left(h\right) \end{aligned} \tag{1}$$

where $o\left(h\right)$ is the quantity as $\lim_{h\to 0} \frac{o(h)}{h} = 0$.

In other words, if interval h is small enough, then the probability of the event "more than one packet arrival during time h" is negligibly small.

If the four conditions above are met, then the traffic is described by a Poisson process [55], i.e. the time between packet arrivals is exponentially distributed, and the probability mass function of $N(t)$ is as follows:

$$P\left[N(t) = n\right] = \frac{(\lambda t)^n}{n!} e^{-\lambda t}, \quad n \geq 0. \tag{2}$$

The Poisson process is used in many practical situations. So in this paper, we use it to generate training and testing samples for SVM performance analysis. However, we would remark that we do not use special properties of a Poisson probability mass function. Our features are limited by order statistics, mean, and median. So, the proposed approach can be applied even in more general situations.

IV. THE PROPOSED INTRUSION DETECTION SYSTEM

The framework of the proposed IDS is given in Figure 3. The two main phases of the system include the training phase and the evaluation phase. Remark, in this paper we consider intrusions accompanied by changing traffic intensity. However, the proposed approach does not utilize any specific properties of intrusions. Thus, it can be adopted for other cases. In the training phase, a training dataset containing labeled samples is obtained. Features are extracted from these samples in the first stage of this phase to obtain a feature pool. The resulting feature pool along with a vector of labels is then used to train the classifier. After a trained classifier is obtained, it is then presented to classify the unobserved samples from the test dataset. To evaluate the performance of the classifier, similar features used in the training phase are extracted from the test samples in the test dataset. These unlabeled test samples

are then given input to the classifier to obtain the predicted output.

1) SUPPORT VECTOR MACHINE

The SVM was developed from the concepts of statistical learning theory in the late 1970s. The SVM primarily deals with two-class classification problems. A linear line, or hyperplane, is constructed as a decision boundary between the datasets of two classes for classification. The data points nearest to the hyperplane, which impart the construction of the hyperplane, are called support vectors. Hence, the algorithm is a support vector machine [12], [13], [56]. The optimized hyperplane can be mathematically expressed as

$$w^T x + b = 0, \tag{3}$$

where w is the vector of weights, x is an input vector, and b represents the bias. The equations of the support vectors for each class are given as

$$\begin{aligned} w^T x + b &= +1, \quad for d_i = +1, \\ w^T x + b &= -1, \quad for d_i = -1, \end{aligned} \tag{4}$$

where d_i corresponds to the respective class, i.e., $d_i = +1$ for class A, and $d_i = -1$ for class B. The optimization problem for training sample $\{(x_i, d_i)\}_{i=1}^{k}$ to find the optimal hyperplane, is given as:

$$\min \Phi(w) = \frac{1}{2} w^T w, \tag{5}$$

such that $d_i(w^T x_i + b) \geq 1, for i = 1, 2, \ldots, k$.
The final decision function can be obtained as follows:

$$f(x) = sign\left(\sum_{i=1}^{N} \alpha_{o,i}(x^T x_i) + b\right), \tag{6}$$

where x denotes the input vector to be classified and N is the number of support vectors obtained in the training phase. The non-negative parameters $\alpha_{o,i}$ are used to define support vectors among input vectors. The linearly non-separable patterns are transformed into a higher dimensional feature space, using a mapping function $\varphi(x)$, allowing for classification of the data using the linear hyperplane. The decision function in Eq. (4) can be modified to

$$f(x) = sign\left(\sum_{i=1}^{N} \alpha_{o,i}(\varphi(x)\varphi(x_i)) + b\right). \tag{7}$$

The inner-product kernel function, defined as $K(x, y) = \varphi(x)\varphi(x_i)$, is used to reduce the complexity of numerical optimization in high-dimensional space. The decision function can be updated as follows:

$$f(x) = sign\left(\sum_{i=1}^{N} \alpha_{o,i} K(x, x_i) + b\right). \tag{8}$$

There are several kernel functions used in an SVM for non-linear pattern classification, such as linear, polynomial, sigmoid, and the radial-basis function (RBF). In this work, three kernel functions (linear, polynomial and RBF) are used. The mathematical expressions of these kernel functions are given in Table 2.

TABLE 2. Inner-product kernels.

Type of kernel function	Kernel function $K(x, x_i), i = 1, 2, ..., N$
Linear	$x^T x_i + c$
Polynomial	$(x^T x_i + 1)^p$
Radial-basis function	$\exp\left(-\frac{\|x - x_i\|^2}{2\sigma^2}\right)$

Algorithm 1 Training Phase of the Classifier

INPUT

λ_{norm}: Parameter of the normal class
λ_{int}: Parameter of the intrusion class
m: Length of the sample/observation
M: Number of observations/samples
$F_{1\times K}$: Feature set of K variables
Kernel_function
Kernel_scale
cross_validation.

OUTPUT

svm_model: Trained classification model.

1: $X^{norm}_{M\times n} \leftarrow$ Generate M signals of m-dimensions using the Poisson distribution parameter λ_{norm}
2: $X^{Fnorm}_{M\times K} \leftarrow$ Extract features from $X^{norm}_{M\times n}$
3: $Y^{norm}_{M\times 1} \leftarrow$ Label the normal class
4: $X^{int}_{M\times n} \leftarrow$ Generate M signals of m-dimensions using the Poisson distribution parameter λ_{int}
5: $X^{Fint}_{M\times n} \leftarrow$ Extract features from $X^{int}_{M\times n}$
6: $X^{int}_{M\times 1} \leftarrow$ Label the intrusion class
7: $Y_{2M\times 1} \leftarrow X^{norm}_{M\times 1}$: Concatenate the two vectors vertically
8: *svm_model* $\leftarrow$ Train SVM model using *Kernel_function*, *cross_validation* with samples $X_{2M\times K}$ and $Y_{2M\times K}$

For multi-class classification problems, the SVM can be used in a one-versus-rest manner [12], [30], [41]. In this approach, m distinct classifiers are formed for m-class classification. In each m^th classifier, the data related to the m^th class are trained as true values, while the rest of the $m - 1$ classes are false values. The label of the test dataset is determined by the classifier giving the maximum output value.

Algorithm 1 illustrates the training phase of the classifier. The parameters of normal class λ_{norm} and intrusion class λ_{int}, the length of sample m, the number of samples M, the set K of features $F_{1\times K}$, the kernel function, the kernel scale, and the cross-validation technique applied, are all given as input to the algorithm. The output of the algorithm is a trained classification model based on an SVM: 'svm-model'.

At first, M signals of m-dimensions using a Poisson distribution with parameter λ_{norm} are generated. Then, the features are extracted from each sample, and all observations

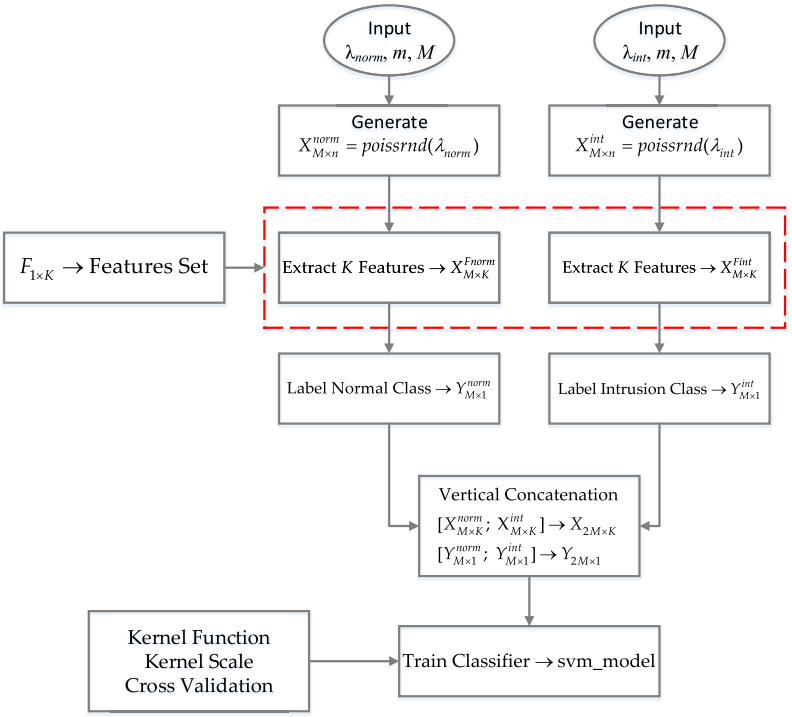

FIGURE 4. Flowchart of Training Phase.

are labeled as the normal class. Similarly, M signals of m-dimensions using a Poisson distribution with parameter λ_{int} are generated. Then, the features are extracted from each sample, and all observations are labeled as an intrusion class. Then, the vectors of observations and labels are concatenated vertically. Finally, the classifier is trained using this set of observations. A flowchart is given in Figure 4 showing the steps as explained above of training phase.

Algorithm 2 shows the steps involved in the testing phase of the proposed scheme. The parameters of the normal class, λ_{norm}, and the intrusion class, λ_{int}, the length of test signal N, the length of window w, the set K of features $F_{1\times K}$, and the trained classifier model (*svm_model*) are given input to the algorithm. The output is obtained in the form of a vector showing the predicted outputs of the test signals.

In first step, generate random number r between 1 and N. Then, an r-dimensional normal signal is generated using Poisson distribution parameter λ_{norm}. Similarly, an $(N - r)$-dimensional intrusion signal using Poisson distribution parameter λ_{int} is generated. These vectors are then concatenated horizontally to obtain a single signal of N dimensions. Then, starting from the first element of the resulting signal, a window of size w is extracted from the signal and given to the classifier for classification. The output label is stored in vector Y. In the next step, the window is obtained by starting from the second element of the signal up to the $(w + 1)^{th}$ element and tested. Similarly, all the $(N - w + 1)$ elements of the signal are presented for testing and the output labels are stored in vector Y. These steps of testing phase are illustrated in flowchart given in Figure 5.

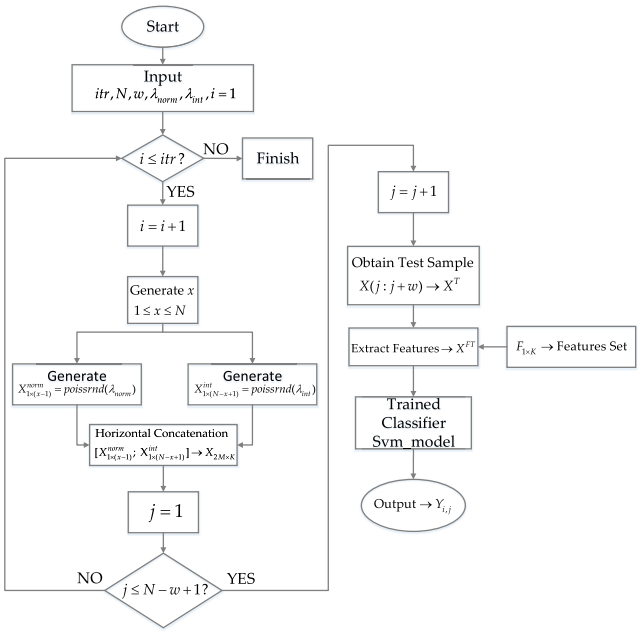

FIGURE 5. Flowchart of Testing Phase.

V. EXPERIMENTAL RESULTS

All the experiments and data acquisition steps are performed in MATLAB version 2018b simulation tool. To obtain the Poisson distributed signal, a built-in function, *poissrnd* is

Algorithm 2 Testing Phase of the Classifier

INPUT

λ_{norm}: Parameter of the normal class
λ_{int}: Parameter of the intrusion class
N: Length of the signal/sample
w: Length of the window/sample
$F_{1\times K}$: Feature set of K variables
itr: Number of iterations.

OUTPUT

$Y_{itr\times(N-n+1)}$: Predicted labels.

1: **for** i in itr **do**
2: $x \leftarrow$ random number $\leq N$
3: $X^{norm}_{1\times(x-1)} \leftarrow$ Generate $(x-1)$- dimensional normal signal using Poisson distribution parameter λ_{norm}
4: $X^{int}_{1\times(N-x+1)} \leftarrow$ Generate $(N-x+1)$-dimensional normal signal using Poisson distribution parameter λ_{int}
5: $X_{1\times N} \leftarrow \left[X^{norm}_{1\times(x-1)}, X^{int}_{1\times(N-x+1)}\right]$ concatenate these vectors horizontally
6: **for** j in $N-w+1$ **do**
7: $X^T \leftarrow X\,(j:j+n)$
8: $X^{FT} \leftarrow$ Extract Features
9: $Y_{i,j} \leftarrow$ *svm_model* $\left(X^{FT}\right)$, Test X^{FT} using trained *svm_model*
10: **end for**
11: **end for**

used. Similarly, for implementation of different models such as NN, k-NN and SVM, the built-in functions *feedforwardnet*, *fitcknn*, *fitcsvm* are used, respectively. For some experiments, the online dataset is utilized. The online dataset is downloaded and saved in the computer first in the format of comma separated values (csv) file. Later on, they are loaded to MATLAB to perform these experiments. Our intrusion detection method and methods offered in other considered papers are independent of routing protocol.

A. PERFORMANCE EVALUATION PARAMETERS

The performance evaluation parameters terms employed in this paper are defined as follows

- *True Positive (TP):* Actual positive predicted as positive
- *True Negative (TN):* Actual negative predicted as negative
- *False Positive (FP):* Actual negative predicted as positive
- *False Negative (FN):* Actual positive predicted as negative
- *Accuracy (ACC):* The ratio of true values to total observations, calculated as follows:

$$ACC = \frac{TP + TN}{N}, \tag{9}$$

where N is the total number of observations.

- *True Positive Rate:* The ratio of true positives to the number of observations predicted as positive. Also known as sensitivity, recall, or hit rate:

$$TPR = \frac{TP}{TP + FN.} \tag{10}$$

- *False Positive Rate:* The ratio of false positives to the sum of false positives and true negatives. Also known as fall-out.

$$FPR = \frac{FP}{FP + TN} \tag{11}$$

- *False Detection Rate:* Defined as (1 - accuracy) and calculated as follows:

$$FDR = \frac{FP + FN}{N.} \tag{12}$$

B. DATA ACQUISITION

The dataset used to prove the efficiency of the proposed IDS system is obtained through simulation. It is composed of 100 normal samples and 100 intruded samples. A sample or observation is the reading of the sensor in a unit time. For instance, a raw sample represent a vector of readings of packet arrival rates obtained during a time instant. The 100 samples are the 100 vectors obtained in 100 different time instants where each vector is comprised of numbers representing packet arrival rates. The term raw here represent the reading obtained from the sensor without applying preprocessing i.e., feature extraction. Each vector has a length, N, referred to as the number of elements per observation directly related to the size of observation time instant. To obtain a normal sample, a vector is generated through simulation, assuming as the packet arrival ratio reading of a time instant under no attack/intrusion scenario, using Poisson distribution with the parameter λ_{norm}. Similarly, the vector obtained using Poisson distribution with the parameter λ_{intr} is considered as a sample from intruded class i.e., the network is under attack. In this way, 100 raw samples from each class are obtained. The simulation parameters are varied for each experiment as illustrated in Table 3. Here, N represent the number of elements per sample and w is the length of window explained and used in next experiments 2 to 4. The features mean, median, and max values are extracted from each of these vectors. The three features or a combination of any two of these features extracted from a single vector are referred to as a preprocessed sample or observation. These preprocessed observations are then utilized for training and validating the proposed IDS, as illustrated in Figure 3.

C. EXPERIMENTS

To avoid any misunderstanding, we use two different terms; attribute and features. The primary measure of packet arrival rate which is obtained from the input data to node is termed as attribute. The minimum, maximum and median extracted from the only attribute (packet arrival rate) are termed as features. Hence, the attribute and features should be considered two different parameters throughout this paper.

TABLE 3. Experimental parameters.

Parameter	Experiment				
	1	2	3	4	5
λ_{norm}	2.2	2	2	2	2
λ_{intr}	2.4	4	6	1.5	1
Train Samples	40	40	40	40	40
Sample Length	500	500	500	500	500
Test Samples	60	$N-w+1$	$N-w+1$	$N-w+1$	$N-w+1$
Sample Length	500	w	w	w	w

TABLE 4. Combinations of features.

Feature Set	Feature list
Set1	Mean and Maximum
Set2	Mean and Median
Set3	Maximum and Median
Set4	Mean, Maximum and Median

TABLE 5. Performance evaluation of linear kernel-based SVM.

Feature Set	Acc(%)	TPR	FPR	TDR	FDR	τ(ms)
Set1	91.6667	0.9000	0.0667	0.9167	0.0833	0.0833
Set2	94.1667	0.9333	0.0500	0.9417	0.0583	0.0583
Set3	56.6667	0.8667	0.7333	0.5667	0.4333	0.4333
Set4	91.6667	0.9000	0.0667	0.9167	0.0833	0.0833

1) EXPERIMENT 1

To obtain the initial simulation results, an SVM-based classifier is trained with three kernels: linear, polynomial, and radial-basis function. The four feature combinations include mean and maximum (max) values, mean and median values, max and median values, and mean, max, and median values. The parameter of the Poisson distribution to generate a normal (or non-intruded) signal is 2.2, whereas to generate an intruded signal, the parameter value is 2.4. The length of a single observation, i.e., the number of elements per observation, is 500. The dataset contained 100 observations from each class (normal and intruded), and 40 observations from each class were used to train the classifier, whereas the remaining 60 observations were used for testing. The feature combinations are listed in Table 4.

Table 5 shows ACC, TPR, FPR, TDR, FDR, and total detection time (τ) from classifying all 60 test observations with the linear kernel-based SVM classifier (linear-SVM). The results show that the classifier can achieve at least 91% accuracy if the input combination of features has a mean value. The worst performance was reported when only max and median were used as input features. A combination of mean and median obtains the highest accuracy among the given combinations. This combination of features also had the highest TPR and TDR, as well as the lowest FPR and FDR. However, τ was slightly higher than the two combinations of max and median, and mean, max, and median.

The results of the performance metrics of the polynomial kernel-based SVM (poly-SVM) are given in Table 6, where kernel scale = 3. A similar trend in performance can be seen across the different combinations of features, from mean and max, through to mean, max, and median. Mean and median

TABLE 6. Performance evaluation of polynomial kernel-based SVM.

Feature Set	Acc(%)	TPR	FPR	TDR	FDR	τ(ms)
Set1	89.1667	0.9167	0.1333	0.8917	0.8917	0.0192
Set2	92.5000	0.9167	0.0667	0.9250	0.0750	0.0059
Set3	55.8333	0.3000	0.1833	0.5583	0.4417	0.0057
Set4	89.1667	0.9167	0.1833	0.8917	0.1083	0.0052

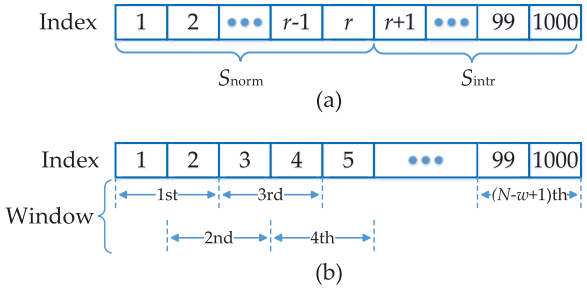

FIGURE 6. Graphical representation of bigram techniques used to obtain (a) signal S obtained for Experiments 2 to 4 and (b) testing signal used in Experiments 2 to 4 with $N = 1000$ and $w = 2$.

outperformed all other combinations of features, achieving a 92% accuracy. However, the accuracies in all feature combinations were degraded, compared to linear SVM. TPR, FPR, TDR, and FDR have almost equal values, with an average difference of 0.099, compared to that of the linear SVM, except for one combination of features: max and median. The TPR in this case was degraded from 0.866 to 0.3; however, the FPR was reduced from 0.7333 to 0.1833. TDR, FDR, and τ are similar to the linear-SVM for this combination of features, as well.

The results obtained from the RBF kernel-based SVM (rbf-SVM) for kernel scale = 0.8 are illustrated in Table 7. As can be seen, the performance of each combination of features is degraded further by using rbf-SVM. However, the mean and median proved to be the best combination from among all of them, achieving the highest accuracy of 91%. Moreover, the TPR increased from 0.3 with poly-SVM to 0.43, as did the FPR (from 0.18 to 0.33) for the max and median combination of features.

In conclusion, linear-SVM outperformed the counter poly-SVM and rbf-SVM in terms of accurately classifying the input signal using all four combinations of features. However, rbf-SVM achieved the highest values in TPR for all combinations of features other than max and median. FPR, TDR, and FDR values varied for different combinations of features for different kernel-based SVMs. In addition, the detection time

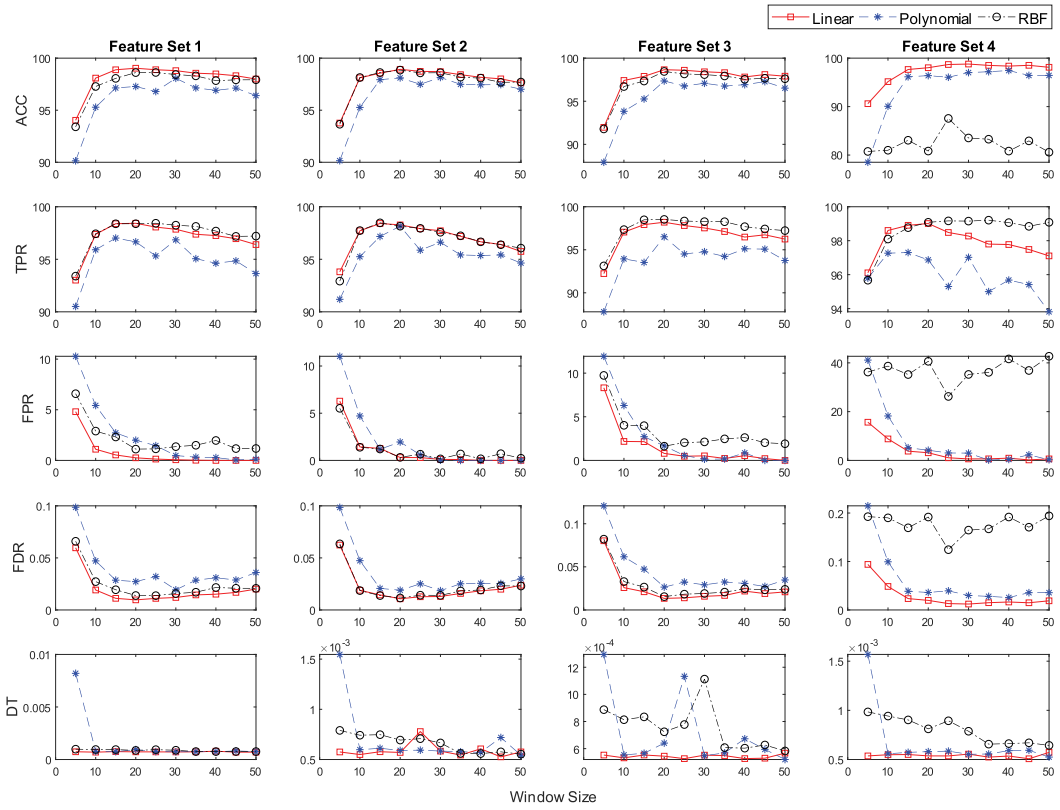

FIGURE 7. ACC, TPR, FPR, FDR, and τ comparison for each features set and kernel function obtained in Experiment 2.

TABLE 7. Performance evaluation of RBF kernel-based SVM.

Feature Set	Acc(%)	TPR	FPR	TDR	FDR	τ(ms)
Set1	81.6667	0.9667	0.3333	0.8167	0.1833	0.0187
Set2	91.6667	0.9667	0.1333	0.9167	0.0833	0.0062
Set3	55.0000	0.4333	0.3333	0.5500	0.4500	0.0056
Set4	82.5000	0.9833	0.3333	0.8250	0.1750	0.0053

for all types of SVM was similar for respective combinations of features. Mean and max took the longest time among all of them to classify 120 observations in the test set. On the other hand, mean, max, and median was the combination of features that required the shortest time from among all combinations of features to obtain final results.

2) EXPERIMENT 2

In the following experiments, the performance of the classifier was tested using a signal, S, of length 1000 elements obtained using a bigram technique (N-gram with N=2) as follows : $S = \{s_1, s_2, \ldots, s_r\}$. To obtain this signal, first, a number, r, was generated randomly between 1 and 1000. Then, signal $S_{norm} = \{s_1, s_2, \ldots, s_r\}$ was obtained through Poisson distribution with parameter λ_{norm}, and the signal $S_{intr} = \{S_{r+1}, S_{r+2}, \ldots, S_{1000}\}$ was generated with a Poisson distribution from parameter λ_{intr}. Finally, the two vectors, S_{norm} and S_{int} were concatenated to obtain one signal, $S = [S_{norm}, S_{intr}]$, of 1000 elements as shown in Figure 6(a).

It is assumed that each element s_i represent the signal element obtained at time t_i. The input test observation, I, to the classifier at time t_i is the window obtained from signal S of window size w, given as $I = \{s_t - w + 1, s_t - w + 2, \ldots, s_t\}$. Figure 6(b) shows an example of such a test signal obtained for $N = 1000$ and $w = 2$. The window size, w, is varied from 5, 10,. . ., 50 to obtain the results in Figure 7. Each column shows the results obtained for each feature set: 1, 2, 3, and 4, respectively. Each row illustrates the performance of the classifier in terms of ACC, TPR, FPR, FDR, and τ. $\lambda_{norm} = 2$ and $\lambda_{intr} = 4$ are used to obtain the training and testing signals.

Figure 7 shows that, for all types of classifier, i.e., linear, polynomial and RBF kernel-based classifiers, there was an increase in the performance efficiency of the classifier with an increase in the size of the window from 5 to 20. Further increases in the window size had no (or a slightly degrading) effect on performance. For instance, the accuracy of classifiers using feature sets 1 and 2 slightly decreased with an increase in window size. However, for feature sets 3 and 4, the performance was almost similar for all window sizes above 20. Nevertheless, the linear kernel-based classifier outperformed the other two classifiers in terms of accuracy. The polynomial kernel-based classifier reported the worst accuracy for features sets 1, 2 and 3. But using feature set 4, the RBF kernel-based classifier showed the lowest accuracy.

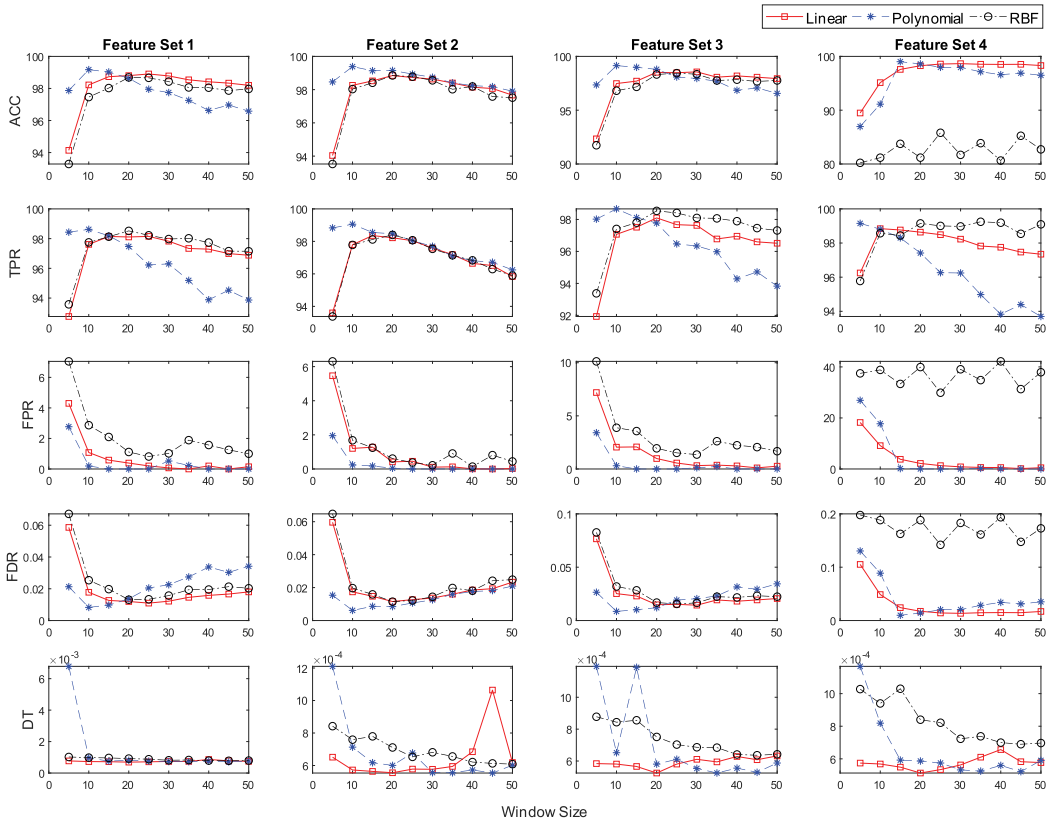

FIGURE 8. ACC, TPR, FPR, FDR, and τ comparison for each features set and kernel function obtained in Experiment 3.

The RBF kernel-based classifier outperformed the linear and polynomial kernel-based classifiers in terms of TPR. The worst results were reported by the polynomial kernel-based classifier. Similarly, the linear kernel-based classifier outperformed the two other kernels, in most cases, in terms of FPR, FDR, and τ. The polynomial kernel-based classifier showed the worst results.

3) EXPERIMENT 3

In this experiment, λ_{norm} is the same as the previous experiment; however, λ_{intr} was increased to 6. Interestingly, the performance of the polynomial kernel-based classifier improved to outperform the linear and RBF kernel-based classifiers in most cases, as shown in Figure 8. For instance, using feature set 1, the accuracy and TPR of the polynomial kernel-based classifier increased for window sizes lower than 20. Further increases in window size degraded the performance of the classifier to a lower level than that reported by the linear and RBF kernel-based classifiers. However, with feature sets 2, 3, and 4, the polynomial kernel-based classifier showed comparable performance to the other two classifiers. Similarly, the improved performance of the polynomial kernel-based classifier is reported in terms of FPR, FDR, and τ. Furthermore, it should be noted that the performance of all classifiers for all sets of features improved, compared to the results obtained in the previous experiment. Intuitively, an increase in λ_{intr} increases the efficiency of all classifiers for all sets of features.

4) EXPERIMENT 4

In this experiment, λ_{intr} was set to 1.5, a lower value than $\lambda_{norm} = 2$. All the other parameters were kept the same as in experiments 2 and 3. As we can see from Figure 9, the performance of the polynomial kernel-based classifier was worse compared to the other two classifiers in many cases. Moreover, the performance of the linear and RBF kernel-based classifiers are comparable to each other.

5) EXPERIMENT 5

In this experiment, λ_{intr} was further decreased to 1 to analyze the effect on the performance of the classifiers. The performance by all classifiers improved for all cases and classifier types, especially for the polynomial kernel-based classifier. These results strengthen the statement that an increase in the difference between λ_{norm} and λ_{intr} increases the performance of the classifiers, as shown in Figure 10. Furthermore, the linear kernel-based classifier can achieve the best results from among the three types of classifiers.

6) EXPERIMENT 6

Figure 11 illustrates the comparison results of the proposed SVM-based algorithm with neural network (NN), k-nearest

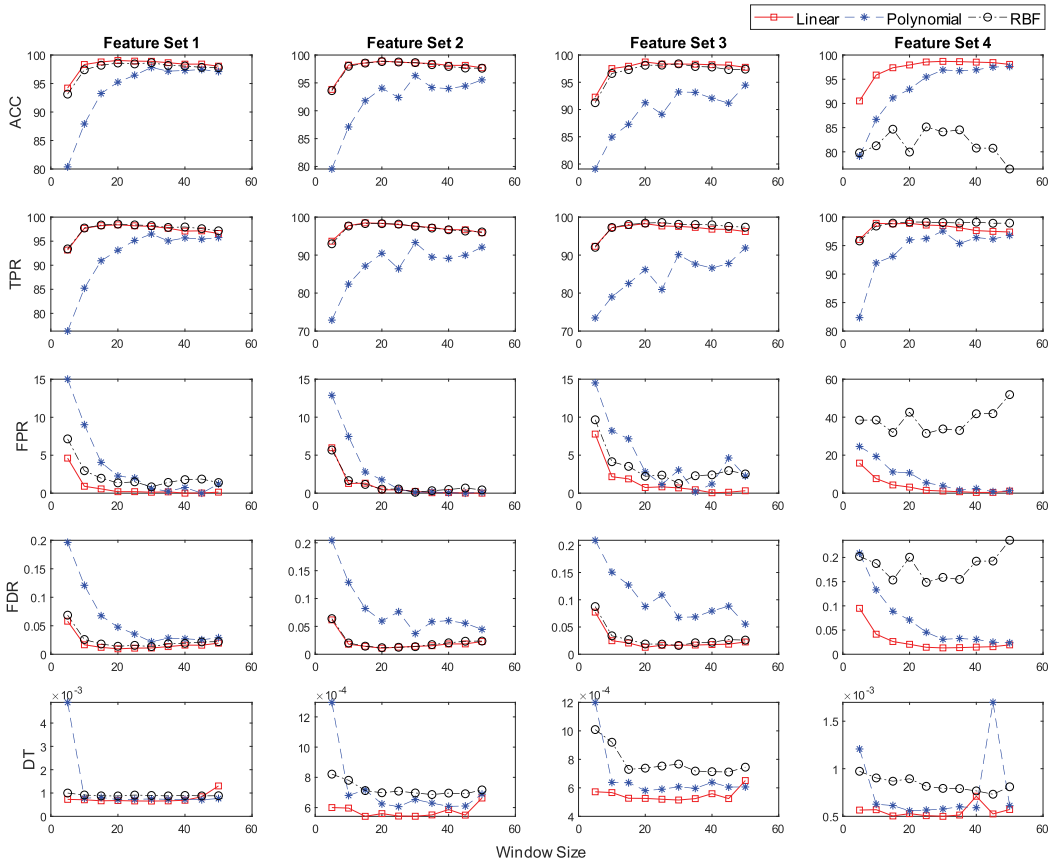

FIGURE 9. ACC, TPR, FPR, FDR, and τ comparison for each features set and kernel function obtained in Experiment 4.

neighbor (KNN) and decision tree (DT) algorithm. The SVM-based algorithm was implemented using three different kernels i.e., linear, polynomial and radial-basis function, illustrated by SVM(L), SVM(P) and SVM(R) in figure, respectively. The properties and parameters of each algorithm used in this experiment are given as follows; *Neural Network*, a feed forward network with input, hidden and output layers comprised of 2 or 3 (depending on the number of input dimensions), 2, and 1 node, respectively. A Levenberg-Marquardt (LM) technique was used in training phase to perform back-propagation; *k-nearest neighbor*, a classification KNN was used to classify an input sample by obtaining the Euclidean distance from 3 nearest neighbors; *Decision Tree*, a classification decision tree was fitted in training phase to perform classification of test samples; Finally, the SVM-based IDS was implemented with three different kernel functions: Linear, Polynomial and RBF, exclusively. All of these algorithms were implemented in MATLAB using built-in tools, *feedforwardnet, fitcknn, fitctree,* and *fitcsvm*. All the parameters, except given above, were set to default values to perform this experiment. The number of training samples per class were set to 40, i.e., 40 samples from normal class and 40 samples from intruded class, resulting in a total of 80 training samples. Whereas the testing set was composed of 1000 samples; 500 from normal class and 500 from intruded class. The parameters $\lambda_{norm} = 2.2$ and $\lambda_{intr} = 2.4$. The length of sample was set to 50 elements. A total of 100 experiments were performed whose results are given in figure in terms of accuracies obtained in each iteration of experiment. The feature sets are kept similar to the previous experiments. It can be observed from figure that in most cases, the SVM-based classifier perform better than other machine learning algorithms. The mean accuracies of SVM(L) and SVM(P) are higher than other algorithms irrespective of the feature set utilized. However, using SVM(R) while using Feature Set 1 and Feature Set 4 perform poor than other techniques. These results verify the results obtained in previous experiments. Furthermore, these results verify the efficiency of proposed IDS implemented with SVM-based classifier using linear and polynomial kernels.

7) EXPERIMENT 7

To prove the efficiency of the proposed algorithm, we present performance comparison in terms of accuracy and CPU time among the proposed algorithm, and the algorithms given in GA-SVM [30], A-IDS [57], and WFS-IDS [43]. The measure of accuracy shows how efficiently an algorithm classify an input signal as normal or intruded. To measure the lightweightness of an algorithm, the two parameters including consumed energy and elapsed time, can be used [58].

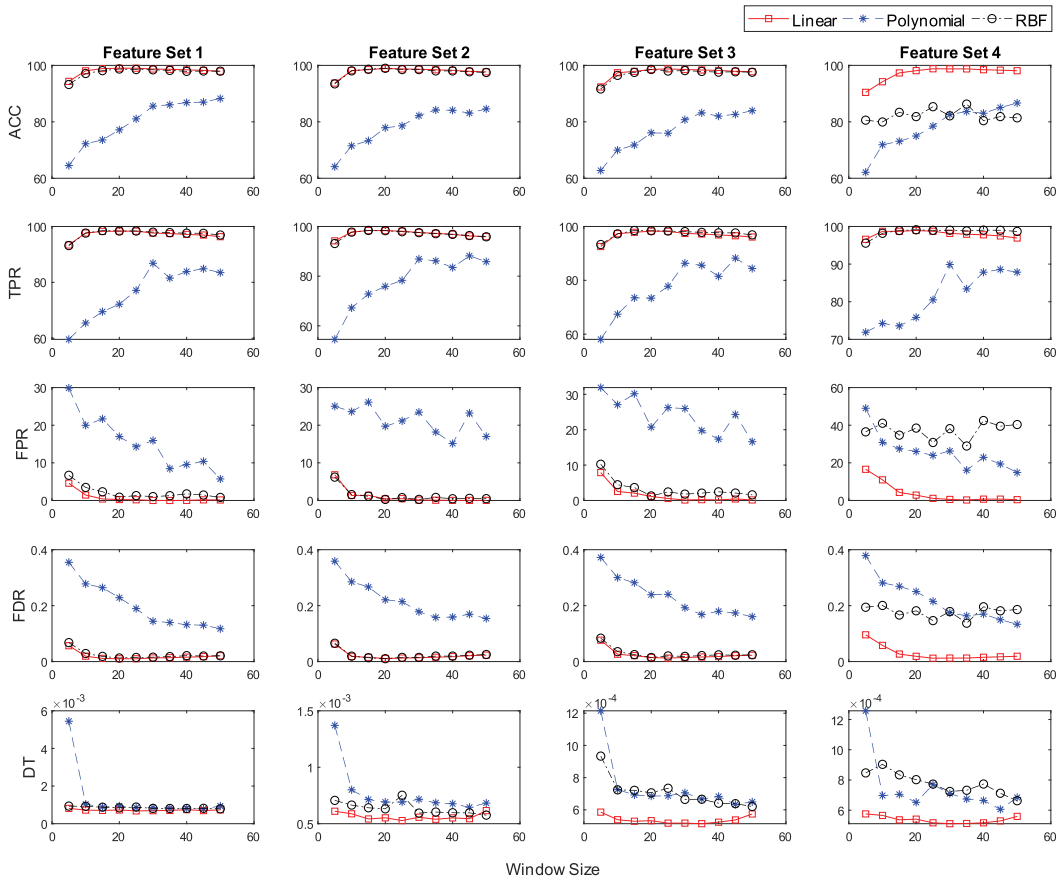

FIGURE 10. ACC, TPR, FPR, FDR, and τ comparison for each features set and kernel function obtained in Experiment 5.

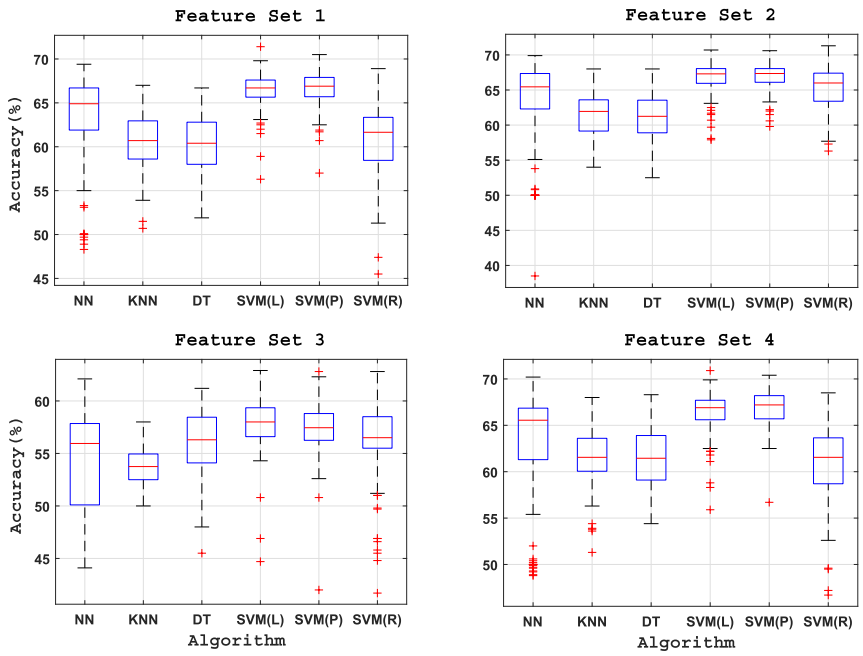

FIGURE 11. Comparative analysis of proposed SVM-based IDS with that implemented using NN, KNN and DT in terms of accuracies of 100 iterations of experiment 6.

In practice, these parameters are directly proportional to each other, given the similar computational resources. In other words, an algorithm taking longer time to execute will more likely consume larger amount of energy as compared to another algorithm with lower computation time. Unfortunately, to obtain the energy consumption analysis of an algorithm, a proper testbed and tools are required to obtain these measurements. In the paper, subsequently we choose the elapsed time as a performance index for the lightweight measurement of the algorithms because we use MATLAB version 2018b to simulate all the algorithms. More specifically, the MATLAB provides some functions to obtain the elapsed time to execute an algorithm. Taking advantage of these functions, we present here the elapsed time analysis of each algorithm to prove the lightweight property of proposed algorithm.

As defined in the introduction section of our paper, the lightweightness is not limited to the property of system requiring low computational resources. There are other requirements which should also be achieved in order to be a lightweight system such as being small, powerful, and flexible enough to be included in a node or network as a permanent element. However, the requirement of being able to perform using low computational resources, which aim to reduce energy consumption of the system, is an integral and the most important characteristic of lightweight IDSs. It should be noted that, the proposed IDS try to achieve all the above-mentioned properties while focusing mainly on the property of being able to perform with low computational resources. This property is proved using CPU time measure.

The CPU time is not very decisive factor in determining the 'lightweightness' of an algorithm. However, we are forced to use this measure because other researchers widely use the CPU time to prove the lightweightness of systems, for example [43], [58]–[61]. Moreover, the authors of [60] have also utilized MATLAB for performing the experiments to prove efficiency of the model. We also do not investigate a particular solution and we consider more general situation. Thus, similar to the method followed in [60], using MATLAB and performing simulation-based experiments is the only way that we can follow to perform these experiments and prove the efficiency of proposed algorithm.

We must accept the following arguments of previously published papers. The energy consumed is obtained as the product of power consumption and CPU time [59]. The power consumption being constant (using same computational resources), the energy consumption is directly related to the CPU time only. Furthermore, Lim *et al.* [58] state that the processing time somewhat depends on the implementation of the system. However, a large difference in time (seconds versus hours or even days) cannot be characterized by the implementation method exclusively. Moreover, Chen and Li [60] presented a comparison of energy consumption and CPU time of their model. The given results generate our claim. The energy consumed is in a direct relation with the CPU time. Even a very small difference in time (micro seconds) is reflected in the energy consumed measure of the system perfectly in direct relation with CPU time measure. These results show that, presenting only CPU time may not replicate the exact practical behavior of the system, however, it can be used to estimate the lightweightness trend of the system.

First, we present a performance comparison among the algorithms GA-SVM, A-IDS, WFS-IDS and the proposed algorithm by utilizing the generated dataset using Poisson distribution as explained in Section V-B. This dataset contains a total of 10 000 samples where the 60% samples are reserved for training the algorithm, with equal number of normal and intruded samples. The remaining 40% samples, with equal number normal and intruded samples, are used to test the algorithms. Each sample in this dataset is attributed in the form of three features; mean, maximum and median measures of the packet arrival rate as explained in the previous sections. We set the parameters of the proposed algorithm based on the results of experiments 1 to 6 as follows; we use the polynomial kernel-based SVM algorithm with kernel scale value 3 and select the features 1 and 3 from the dataset. However, the algorithms from GA-SVM, A-IDS, WFS-IDS have their respective feature selection procedures to select two most discriminative among three features. To obtain the accuracy given in Table 8, each algorithm is trained using the training set. After the training is finished, the testing subset is used to assess each algorithm.

As it can be seen from Table, the proposed algorithm achieves the highest accuracy of 98.35%. As the GA-SVM algorithm becomes a similar algorithm to the proposed algorithm in the testing phase (because only SVM part is utilized), the accuracy of this algorithm is 98.21%, almost equal to that of the proposed algorithm. However, the A-IDS and WFS-IDS get the accuracy of almost 97.9 and 97.08%, respectively. In terms of the CPU time, the proposed algorithm is also listed on the top with the minimum training, testing and total times as 16.3125, 0.0469, and 16.3594 seconds, respectively. The A-IDS comes next by taking 17.1719, and 0.6250 seconds for training and testing phases, respectively. However, the GA-SVM and WFS-IDS algorithms take much longer training time of almost 135 and 134 seconds. Nevertheless, the testing time of these algorithm is lower than that of A-IDS. However, the proposed algorithm has the lowest testing time as well.

The dataset used in previous comparative analysis was generated by using the MATLAB 2018b according to Poisson distribution with two different parameter values, each corresponding to one class of Normal and Intruded. There are some online datasets which the researchers normally have used to prove their proposed intrusion detection system [10]. Some famous datasets include the NSL-KDD and CICIDS217 both generated by Canadian Institute for Cybersecurity unit based at University of New Brunswick. These datasets are considered as benchmark for analysis of IDS by many researchers. However, it should be noted that the proposed IDS in this paper has two equally important parts; the attribute of the received data used to perform intrusion detection, and the

TABLE 8. Performance comparisons among the proposed algorithm with GA-SVM, A-IDS, and WFS-IDS using beget Dataset in terms of accuracy and CPU time.

Algorithm	Features Selected	Accuracy (%)	CPU Time (seconds)		
			Training Time	Testing Time	Total Time
Proposed	1, 3	98.35	16.3125	0.0469	16.3592
GA-SVM [30]	1, 3	98.21	135.0156	0.0625	135.0781
A-IDS [57]	1, 3	97.90	17.1719	0.6250	17.7969
WFS-IDS [43]	1, 3	97.08	134.2066	0.0544	134.261

TABLE 9. CICIDS2017 dataset features.

S. No	Features	S. No	Features
1	Destination Port	40	Max Packet Length
2	Flow Duration	41	Packet Length Mean
3	Total Fwd Packets	42	Packet Length Std
4	Total Backward Packets	43	Packet Length Variance
5	Total Length of Fwd Packets	44	FIN Flag Count
6	Total Length of Bwd Packets	45	SYN Flag Count
7	Fwd Packet Length Max	46	RST Flag Count
8	Fwd Packet Length Min	47	PSH Flag Count
9	Fwd Packet Length Mean	48	ACK Flag Count
10	Fwd Packet Length Std	49	URG Flag Count
11	Bwd Packet Length Max	50	CWE Flag Count
12	Bwd Packet Length Min	51	ECE Flag Count
13	Bwd Packet Length Mean	52	Down/Up Ratio
14	Bwd Packet Length Std	53	Average Packet Size
15	Flow Bytes/s	54	Avg Fwd Segment Size
16	Flow Packets/s	55	Avg Bwd Segment Size
17	Flow IAT Mean	56	Fwd Header Length
18	Flow IAT Std	57	Fwd Avg Bytes/Bulk
19	Flow IAT Max	58	Fwd Avg Packets/Bulk
20	Flow IAT Min	59	Fwd Avg Bulk Rate
21	Fwd IAT Total	60	Bwd Avg Bytes/Bulk
22	Fwd IAT Mean	61	Bwd Avg Packets/Bulk
23	Fwd IAT Std	62	Bwd Avg Bulk Rate
24	Fwd IAT Max	63	Subflow Fwd Packets
25	Fwd IAT Min	64	Subflow Fwd Bytes
26	Bwd IAT Total	65	Subflow Bwd Packets
27	Bwd IAT Mean	66	Subflow
28	Bwd IAT Std	67	Bwd Bytes
29	Bwd IAT Max	68	Init Win bytes forward
30	Bwd IAT Min	69	Init Win bytes backward
31	Fwd PSH Flags	70	act data pkt fwd
32	Bwd PSH Flags	71	min seg size forward
33	Fwd URG Flags	72	Active Mean
34	Bwd URG Flags	73	Active Std
35	Fwd Header Length	74	Active Max
36	Bwd Header Length	75	Active Min
37	Fwd Packets/s	76	Idle Mean
38	Bwd Packets/s	77	Idle Std
39	Min Packet Length	78	Idle Min

classifier part. As the first part, we have proposed to utilize the packet arrival rate of the receiving data. After that, the SVM-based classifier is used to classify the input sample as normal or intruded. Unfortunately, the benchmark datasets, which are available online, do not contain the required attribute of data, i.e. the packet arrival rate. This is why it is not possible to directly use this dataset to prove the efficiency of the proposed algorithm. To prove the efficiency of the proposed algorithm, however we generate a parallel synchronized dataset to the CICIDS2017. The steps of generating this beget dataset, which is used for further analysis, are shown by using a flowchart in Figure 12.

The CICIDS2017 dataset [62]is a benchmark dataset available online to assess and prove the cybersecurity algorithms such as intrusion detection systems (IDS) and intrusion

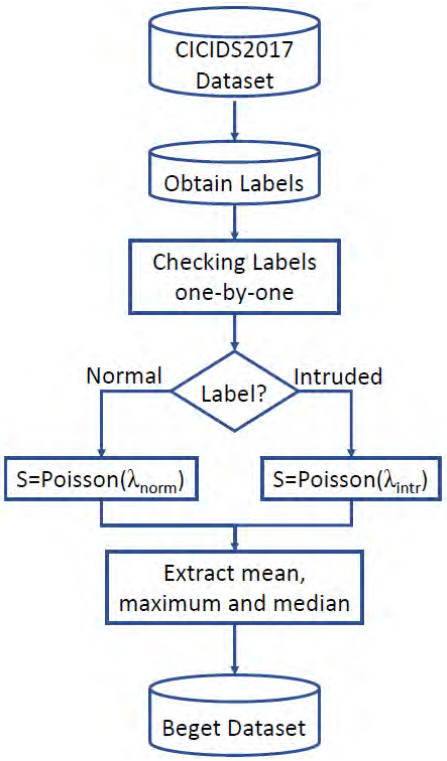

FIGURE 12. Procedure of generating Beget Dataset.

prevention systems (IPS). Generated in 2017, this dataset contains benign (normal) and the then up-to-date attacks. The dataset is composed of data obtained for 5 days, starting at 9 a.m., Monday, July 3, 2017 and ending at 5 p.m. on Friday July 7, 2017. On different days, different types of attacks were implemented in the network to obtain this dataset. We select a subset of CICIDS2017 dataset to assess and compare the proposed algorithm. The selected subset contained the samples obtained from network in the afternoon of Friday, July 7, 2017. The DDoS attack was implemented in the network at random points in time. This subset contained a total of 225,745 samples attributed in 78 features given in Table 9. We obtain the labels vector of CICIDS2017 dataset. Then we check labels one-by-one to obtain the beget dataset. If the i^{th} label of CICIDS2017 dataset is benign, we add a benign sample S, which is generated by Poisson distribution with parameter of normal class, λ_{norm}, as the i^{th} sample to beget dataset. Similarly, the i^{th} sample of beget dataset is obtained by Poisson distribution with parameter of intruded class, λ_{intr} if the i^{th} sample of CICIDS2017 dataset is intruded.

TABLE 10. Performance comparison among the proposed algorithm, GA-SVM, A-IDS, and WFS-IDS using CICIDS2017 dataset.

Algorithm	Features Selected	Accuracy (%)	CPU Time (seconds)		
			Training Time	Testing Time	Total Time
Proposed	1, 3	98.03	208.9063	2.2813	211.1876
GA-SVM [30]	1, 2, 4, 5, 6, 9, 11, 13, 15, 16, 17, 22, 37, 64	98.01	49953.1336	3.1300	49956.2636
A-IDS [57]	2, 5, 15, 16, 17, 22, 37, 64	89.76	1893.9001	4.0156	1897.9157
WFS-IDS [43]	2, 4, 5, 11, 15, 16, 17, 18, 22, 37, 55, 64	97.68	45651.0263	3.6600	45654.4863

In this way, we obtain a synchronized beget dataset with CICIDS2017 which has similar number and position (class label) of samples, as shown in Figure 12.

The data is divided into two subsets; training and testing data sets. The training data contain 60% of the samples including 86 729 benign samples and 48 718 intruded samples. Similarly, the testing data contain 49 000 benign and 41 298 intruded samples. This CICIDS2017 dataset is utilized for GA-SVM, A-IDS and WFS-IDS algorithms while the synchronized parallel generated beget dataset is utilized for the proposed algorithm. However, the number of training, testing and total samples as well as the labels remain same in both datasets. In this way, we present the performance comparison among the algorithms in Table 10 by using CICIDS2017 dataset.

The second column shows the features selected by each algorithm from the dataset. In the proposed algorithm, we use the beget dataset which has three features; mean, max and median. Based on the results from experiments 1 to 6, we select mean and median values for the proposed algorithm. The other techniques utilize respective feature selection algorithm to obtain the set of most discriminative features given in the second column of Table 10. The accuracy of the proposed algorithm is the highest at 98.03%. The worst performance is shown by the A-IDS algorithm with 89.76% accuracy. Similarly, the CPU execution time of the proposed scheme is the lowest with 208.9063, and 2.281 seconds for training and testing phases, respectively. Subsequently, these results prove the lightweightness property of the proposed algorithm with the given set.

A confusion matrix is a simple table layout used to understand or analyze the performance of a classifier under consideration. The actual labels of the samples should be known to obtain this table. Generally, each row represents the labels of actual class while the columns correspond to the instances in predicted class. In practice, one class is set as positive and the other as negative to fill up confusion matrix. In this paper, the Benign or Normal class is set as positive class and the Intruded class as negative class. In this table, each cell represents a specific measure among the TPs, FNs, FPs, and TNs instances in the order shown in Figure 13.

Figure 14 shows the confusion matrices obtained for (a) Proposed IDS, (b) GA-SVM, (c) A-IDS and WFS-IDS algorithms using the CICIDS2017 and related Benign datasets. The results of proposed, GA-SVM and WFS-IDS are comparable. However, the A-IDS shows comparatively poor performance. Moreover, the proposed IDS successfully

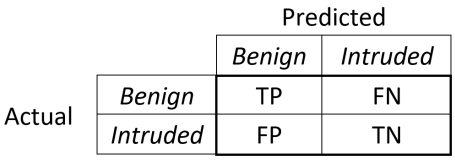

FIGURE 13. Confusion matrix.

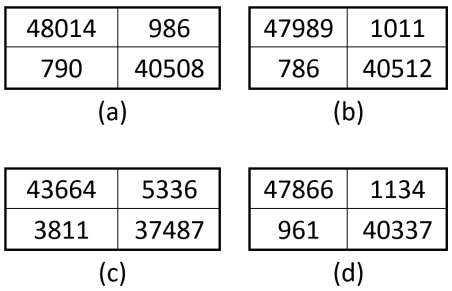

FIGURE 14. Confusion Matrices obtained utilizing CICIDS Dataset with (a) proposed, (b) GA-SVM, (c) A-IDS, and (d) WFS-IDS algorithms.

achieve highest TPs, TNs and lowest FNs instances. However, the FPs instances of proposed algorithm are slightly higher than those obtained by GA-SVM.

VI. DISCUSSION

The tradeoff for using only three features as opposed to '40 complex attributes' is related to the issue of feature selection and reduction. The advantage of using only three features is as following: First, the system processing time can be reduced due to low time consumption of single attribute acquisition from input data instead of multiple attributes. Secondly, extracting just 2 or 3 features from that single attribute takes lower time as compared to extraction of up to 40 features from the multiple attributes. Lastly, the complexity of SVM is also reduced because of utilizing a much lower number of dimensions (features) of input samples. Combining all these effects makes a big difference to the complexity of system. Furthermore, the feature selection step is omitted in the case of the proposed algorithm. These points can be considered as the positive effects of utilizing the proposed algorithm along with the proposed signal preprocessing model. Nevertheless, the feature selection and reduction techniques do not necessarily converge to global optimum, and sometimes end up selecting redundant features, which ultimately results in poor performance of classifier.

Apparently, the main drawback of the proposed algorithm is that it lacks the ability to detect intrusions which do not have concomitant (increasing or decreasing) effect on the traffic intensity of node. The algorithms which consider the complex 40 features may be able to detect more sophisticated intrusions. This issue is reserved for future works.

In this paper, we have considered the packet arrival rate, which follow the Poisson distribution, of the traffic intensity to the node. The Poisson distribution is just used for performance evaluation. We use this distribution because it had been offered in papers of authoritative journals [46]–[54]. It should be noted that, we do not use specific properties of CDF. Therefore, all expenses will be the same for any other traffic pattern followed in practice. However, the selected features such as mean, maximum and median, and the proposed detection scheme can be used irrespective of the type of distribution, given the condition that the intrusion or attack has an increasing or decreasing effect on the traffic intensity (packet arrival rate) to the node.

A. WHY NEED TO USE PACKET ARRIVAL RATE ATTRIBUTE?

As we have explained in Section III, the types of intrusions or attacks considered in this paper are the ones which influence the traffic intensity. Either the data rate is decreased (e.g. in case of packets flooding attack, jamming attack etc.) or increased (e.g. black hole attack, wormhole attack) whenever any of these intrusions occur. This means that if the IDS monitor the traffic intensity alone (or more specifically packet arrival rate measure), it may be able to detect these intrusions most of the time. This is our motivation of proposing an IDS which rely on packet arrival rate attribute exclusively to detect the intrusions. This claim is also supported by experimental results given in the paper.

B. WHY NEED TO EXTRACT MIN, MAX AND MEDIAN FEATURES?

The packet arrival rate is the only attribute of data which is used for intrusion detection in the network. Now, the question is why needed to extract features from this attribute? Why not use this attribute to detect an intrusion using a threshold value? If the packet arrival rate goes higher than the threshold value, it can be considered as intrusion and vice versa. The answer to these questions can be given in a single argument; the threshold value selection is not an easy and reliable way. To find out a threshold for any system needs a continuous monitoring of the network for a long time to get an estimate of the threshold value. Selecting a sub-optimal value of threshold would result in higher miss-detection or false-alarm instances. Moreover, it is believed that the network conditions are not consistent all the times. The nodes may observe variations in packet arrival rate depending on the network conditions i.e., network may be very busy or idle. Therefore, the threshold selection method is not favored for detection and classification application. On the other hand, the machine learning algorithms try to learn the characteristics of the network from a handful amount of historical data.

At this point, we have two choices; either directly use the only attribute (packet arrival rate attribute value) as the single input, or extract features (minimum, maximum and median values) from this attribute to give input to machine learning-based classifier. In former case, using the single input to the classifier may degrade the performance of system because of two reasons. First, there may be a single value which is included in both classes i.e., intrusion and non-intrusion classes. For instance, depending on the network condition, a specific packet arrival rate may or may not be resulted due to the intrusion in the network. Secondly, if a single value of packet arrival rate attribute is used, the classifier needs to perform detection every time we get a new value. Ultimately, the energy consumption is increased due to the utilization of computational resources more frequently. In this paper, the minimum, maximum and median values are obtained from the packet arrival rate attribute over a window of time to solve both the problems. For instance, a packet of data is arrived every t seconds to the node. If this single value is used, then the classifier will perform the classification task every t seconds. On the contrary, if we use a time window $T >> t$, then the classifier has to perform classification every T seconds. This leads to reduce the frequency of utilizing computational resources as well as the algorithm converge better as compared to using the first case of utilizing a single attribute alone.

C. WHY SVM?

The SVM is favored among other machine learning algorithms because of its efficient performance. The performance comparison among different machine learning algorithms given in the experimental results section confirms our claim. Furthermore, the lightweightness of the proposed algorithm is proved in the experimental results as well.

D. HOW DO WE GET LESS TRAINING AND TESTING TIMES?

The main reason that the proposed algorithm has the lowest training and testing time is that the other algorithms have additional feature selection properties. They try to select the best subset of features using complex optimization techniques. For instance, GA-SVM utilizes genetic algorithm to choose the best features among given set. Similarly, A-IDS and WFS-IDS algorithms analyze and select the best features using wrapper-based feature selection mechanism. A detailed explanation about these algorithms is out of scope of the current work, therefore, readers are suggested to refer these papers for more details. However, the major reason which reduces the computational time of the proposed algorithm is the elimination of the feature selection which is the part of training phase only.

The testing phase has no feature selection step, and hence, the difference between the testing time of all these algorithms is very low as given in Table 8 and 10. However, the small differences in training time are reported because of using different algorithms.

E. HOW DO WE GET BETTER ACCURACY?

The complexity of the classifier has a direct effect of the complexity and size of input vectors. The simple and small vectors with high discrimination power among different classes are easy to classify by classifier. On the other hand, the higher number of the dimensions with complex relation among features of input vector increases the challenge of classification for classifier. In authors' opinion, the input vector of only 2 dimensions with comparatively low complexity among features are the key factors which lead the classifier to obtain a highest accuracy of proposed IDS.

There are a few control values used in the proposed algorithm. First, the kernel parameter used in the support vector machine. Changing this value may affect the performance of classifier. An optimized value can be obtained by hit and trial. Another value which should be selected carefully is the time window size. As shown in the experimental results, increase in time window size may improve the performance to some extent. Further increase in the time window size may degrade the performance of classifier.

VII. CONCLUSIONS AND FUTURE WORKS

The IoT is a promising technology developed for applications ranging from small smart-home systems to large networks, such as smart grids. However, this vast network is exposed to different types of attacks, compromising its reliability. Furthermore, the limitations in the nodes, including memory, computational resources, and battery capacity, challenge network security. It is necessary to design a lightweight system that can efficiently improve the security of the IoT with the available resources.

This paper focuses on designing a lightweight IDS for anomaly detection in the IoT. A common type of attack, known as DDoS, is the target. The proposed IDS is focuses on two major issues; the attribute of the receiving data used to classify the signal and the machine learning based classifier. The only attributed considered in this paper is the packet arrival rate to the node. For classification purpose, an SVM-based classifier with input given in the form of two or three incomplex features is utilized. Through a series of experiments, we prove that these two factors (the packet arrival rate attribute and an SVM-based classifier) can be enough to detect the intrusion in IoT network.

Furthermore, we presented a comparative analysis of SVM-based classifier with other machine learning-based classifiers including NN, k-NN and DT to show the advantage of utilizing SVM in terms of accuracy over other techniques. For further proof, we also presented a comparison of proposed algorithm with other IDS proposed in literature. The results show that an SVM-based IDS can perform satisfactorily in detection of attacks. Also, the lightweightness measure of proposed algorithm is proven in terms of CPU time execution.

An investigation of various concomitant effects of attacks and increase in the scope of this IDS system to encompass other types of intrusions, where the effect of changing traffic intensity is not clearly pronounced or masked by intruders, is reserved for future works. Furthermore, concrete details of IDS implementation and intrusions mitigation are defined by application domains and strategy of security perimeter deployment. It is also a direction of our future work.

REFERENCES

[1] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in Internet of Things," *J. Netw. Comput. Appl.*, vol. 84, pp. 25–37, Apr. 2017.

[2] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in Internet of Things: The road ahead," *Comput. Netw.*, vol. 76, pp. 146–164, Jan. 2015.

[3] D. Singh, G. Tripathi, and A. J. Jara, "A survey of Internet-of-Things: Future vision, architecture, challenges and services," in *Proc. IEEE World Internet Things (WF-IoT)*, Mar. 2014, pp. 287–292.

[4] A. Meddeb, "Internet of Things standards: Who stands out from the crowd?" *IEEE Commun. Mag.*, vol. 54, no. 7, pp. 40–47, Jul. 2016.

[5] H. Sedjelmaci, S. M. Senouci, and T. Taleb, "An accurate security game for low-resource IoT devices," *IEEE Trans. Veh. Technol.*, vol. 66, no. 10, pp. 9381–9393, Oct. 2017.

[6] Y. Fu, Z. Yan, J. Cao, and O. Koné, and X. Cao, "An automata based intrusion detection method for internet of things," *Mobile Inf. Syst.*, vol. 2017, May 2017, Art. no. 1750637.

[7] M. Roesch *et al.*, "Snort—Lightweight Intrusion Detection for Networks," in *Proc. Lisa*, vol. 99, 1999, pp. 229–238.

[8] T. H. Hai, E.-N. Huh, and M. Jo, "A lightweight intrusion detection framework for wireless sensor networks," *Wireless Commun. Mobile Comput.*, vol. 10, no. 4, pp. 559–572, 2010.

[9] Y. Maleh and A. Ezzati, "Lightweight intrusion detection scheme for wireless sensor networks," *IAENG Int. J. Comput. Sci.*, vol. 42, no. 4, pp. 347–354, 2015.

[10] I. Sharafaldin, A. Gharib, A. H. Lashkari, and A. A. Ghorbani, "Towards a reliable intrusion detection benchmark dataset," *Softw. Netw.*, vol. 2017, no. 1, pp. 177–200, 2018.

[11] *KDD'99 Dataset*. Accessed: Mar. 10, 2018. [Online]. Available: http://www.unb.ca/cic/datasets/index.html

[12] S. U. Jan, V.-H. Vu, and I. Koo, "Throughput maximization using an SVM for multi-class hypothesis-based spectrum sensing in cognitive radio," *Appl. Sci.*, vol. 8, no. 3, p. 421, 2018.

[13] S. U. Jan, Y. D. Lee, J. Shin, and I. Koo, "Sensor fault classification based on support vector machine and statistical time-domain features," *IEEE Access*, vol. 5, pp. 8682–8690, May 2017.

[14] P. T. Noi and M. Kappas, "Comparison of random forest, K-nearest neighbor, and support vector machine classifiers for land cover classification using sentinel-2 imagery," *Sensors*, vol. 18, no. 1, p. 18, 2018.

[15] J. Deogirikar and A. Vidhate, "Security attacks in IoT: A survey," in *Proc. IEEE Int. Conf. I-SMAC (IoT Social, Mobile, Analytics Cloud) (I-SMAC)*, Feb. 2017, pp. 32–37.

[16] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[17] I. Tomić and J. A. McCann, "A survey of potential security issues in existing wireless sensor network protocols," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 1910–1923, Dec. 2017.

[18] M. D. Donno, N. Dragoni, A. Giaretta, and A. Spognardi, "DDoS-capable IoT malwares: Comparative analysis and Mirai investigation," *Secur. Commun. Netw.*, vol. 2018, Feb. 2018, Art. no. 7178164.

[19] V. Shakhov and I. Koo, "Depletion-of-battery attack: Specificity, modelling and analysis," *Sensors*, vol. 18, no. 6, p. 1849, 2018.

[20] F. Farahnakian and J. Heikkonen, "A deep auto-encoder based approach for intrusion detection system," in *Proc. IEEE 20th Int. Conf. Adv. Commun. Technol. (ICACT)*, Feb. 2018, pp. 178–183.

[21] S.-M. Lee, B.-D. Jeong, and S.-B. Suh, "Method of intrusion detection in terminal device and intrusion detecting apparatus," U.S. Patent 8 701 188, Apr. 15, 2014.

[22] J.-P. Ramirez-Paredes, E. A. Doucette, J. W. Curtis, and V. Ayala-Ramirez, "Sensor compromise detection in multiple-target tracking systems," *Sensors*, vol. 18, no. 2, p. 638, 2018.

[23] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018.
[24] D. Oh, D. Kim, and W. W. Ro, "A malicious pattern detection engine for embedded security systems in the Internet of Things," *Sensors*, vol. 14, no. 12, pp. 24188–24211, 2014.
[25] M. H. Ali, B. A. D. Al Mohammed, A. Ismail, and M. F. Zolkipli, "A new intrusion detection system based on fast learning network and particle swarm optimization," *IEEE Access*, vol. 6, pp. 20255–20261, 2018.
[26] M. Moukhafi, K. El Yassini, and S. Bri, "A novel hybrid GA and SVM with PSO feature selection for intrusion detection system," *Int. J. Adv. Sci. Res. Eng.*, vol. 4, pp. 129–134, May 2018.
[27] R. Vijayanand, D. Devaraj, and B. Kannapiran, "A novel intrusion detection system for wireless mesh network with hybrid feature selection technique based on GA and MI," *J. Intell. Fuzzy Syst.*, vol. 34, no. 3, pp. 1243–1250, 2018.
[28] E. Kabir, J. Hu, H. Wang, and G. Zhuo, "A novel statistical technique for intrusion detection systems," *Future Gener. Comput. Syst.*, vol. 79, pp. 303–318, Feb. 2018.
[29] L. Li, Y. Yu, S. Bai, Y. Hou, and X. Chen, "An effective two-step intrusion detection approach based on binary classification and κ-NN," *IEEE Access*, vol. 6, pp. 12060–12073, 2018.
[30] P. Tao, Z. Sun, and Z. Sun, "An improved intrusion detection algorithm based on GA and SVM," *IEEE Access*, vol. 6, pp. 13624–13631, 2018.
[31] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in IoT," *Sensors*, vol. 17, no. 9, p. 1967, 2017.
[32] F. Jiang *et al.*, "Deep learning based multi-channel intelligent attack detection for data security," *IEEE Trans. Sustain. Comput.*, to be published.
[33] L. khalvati, M. Keshtgary and N. Rikhtegar, "Intrusion detection based on a novel hybrid learning approach," *J. AI Data Mining*, vol. 6, no. 1, pp. 157–162, 2018.
[34] L. Han, M. Zhou, W. Jia, Z. Dalil, and X. Xu, "Intrusion detection model of wireless sensor networks based on game theory and an autoregressive model," *Inf. Sci.*, vol. 476, pp. 491–504, Feb. 2018.
[35] M. A. Abdullah, B. M. Alsolami, H. M. Alyahya, and M. H. Alotibi, "Intrusion detection of DoS attacks in WSNs using classification techniuqes," *J. Fundam. Appl. Sci.*, vol. 10, no. 4S, pp. 298–303, 2018.
[36] M. Ozay, I. Esnaola, F. T. Y. Vural, S. R. Kulkarni, and H. V. Poor, "Machine learning methods for attack detection in the smart grid," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 8, pp. 1773–1786, Aug. 2016.
[37] J. J. Q. Yu, Y. Hou, and V. O. K. Li, "Online false data injection attack detection with wavelet transform and deep neural networks," *IEEE Trans. Ind. Informat.*, vol. 14, no. 7, pp. 3271–3280, Jul. 2018.
[38] Y. He, G. J. Mendis, and J. Wei, "Real-time detection of false data injection attacks in smart grid: A deep learning-based intelligent mechanism," *IEEE Trans. Smart Grid.*, vol. 8, no. 5, pp. 2505–2516, Sep. 2017.
[39] S. Ahmed, Y. Lee, S.-H. Hyun, and I. Koo, "Feature selection–based detection of covert cyber deception assaults in smart grid communications networks using machine learning," *IEEE Access*, vol. 6, pp. 27518–27529, 2018.
[40] S. Ahmed, Y. Lee, S.-H. Hyun, and I. Koo, "Covert cyber assault detection in smart grid networks utilizing feature selection and Euclidean distance-based machine learning," *Appl. Sci.*, vol. 8, no. 5, p. 772, 2018.
[41] S. Teng, N. Wu, H. Zhu, L. Teng, and W. Zhang, "SVM-DT-based adaptive and collaborative intrusion detection," *IEEE/CAA J. Autom. Sinica*, vol. 5, no. 1, pp. 108–118, Jan. 2018.
[42] G. Helmer, J. S. K. Wong, V. G. Honavar, L. Miller, and Y. Wang, "Lightweight agents for intrusion detection," *J. Syst. Softw.*, vol. 67, no. 2, pp. 109–122, 2003.
[43] Y. Li, J.-L. Wang, Z.-H. Tian, T.-B. Lu, and C. Young, "Building lightweight intrusion detection system using wrapper-based feature selection mechanisms," *Comput. Secur.*, vol. 28, no. 6, pp. 466–475, 2009.
[44] A. Mosenia and N. K. Jha, "A comprehensive study of security of Internet-of-Things," *IEEE Trans. Emerg. Topics Comput.*, vol. 5, no. 4, pp. 586–602, Dec. 2017.
[45] P. Sarigiannidis, E. Karapistoli, and A. A. Economides, "Modeling the Internet of Things under attack: A G-network approach," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 1964–1977, Dec. 2017.
[46] A. Augustin, J. Yi, T. Clausen, and W. M. Townsley, "A study of LoRa: Long range & low power networks for the Internet of Things," *Sensors*, vol. 16, no. 9, p. 1466, 2016.
[47] B. Moon, "Dynamic spectrum access for Internet of Things service in cognitive radio-enabled LPWANs," *Sensors*, vol. 17, no. 12, p. 2818, 2017.
[48] Y.-B. Lin, S.-Y. Wang, C.-C. Huang, and C.-M. Wu, "The SDN approach for the aggregation/disaggregation of sensor data," *Sensors*, vol. 18, no. 7, p. 2025, 2018.
[49] J. Zhang, G. Han, and Y. Qian, "Queuing theory based co-channel interference analysis approach for high-density wireless local area networks," *Sensors*, vol. 16, no. 9, p. 1348, 2016.
[50] D. Xu *et al.*, "Maximizing throughput for low duty-cycled sensor networks," *Comput. Netw.*, vol. 139, pp. 48–59, Jul. 2018.
[51] J. Zhu, Y. Song, D. Jiang, and H. Song, "A new deep-Q-learning-based transmission scheduling mechanism for the cognitive Internet of Things," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2375–2385, Aug. 2018.
[52] J. H. Abawajy and M. M. Hassan, "Federated Internet of Things and cloud computing pervasive patient health monitoring system," *IEEE Commun. Mag.*, vol. 55, no. 1, pp. 48–53, Jan. 2017.
[53] Q. Ye and W. Zhuang, "Token-based adaptive MAC for a two-hop Internet-of-Things enabled MANET," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1739–1753, Oct. 2017.
[54] D. Uckelmann, M. Harrison, and F. Michahelles, "An architectural approach towards the future Internet of Things," in *Architecting the Internet of Things*. Berlin, Germany: Springer, 2011, pp. 1–24.
[55] K. S. Trivedi, *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*. Hoboken, NJ, USA: Wiley, 2008.
[56] C. Campbell and Y. Ying, *Learning with Support Vector Machines* (Synthesis Lectures on Artificial Intelligence and Machine Learning), vol. 5, no. 1. San Rafael, CA, USA: Morgan & Claypool, 2011, pp. 1–95.
[57] S. Aljawarneh, M. Aldwairi, and M. B. Yassein, "Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model," *J. Comput. Sci.*, vol. 25, pp. 152–160, Mar. 2018.
[58] T.-S. Lim, W.-Y. Loh, and Y.-S. Shih, "A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms," *Mach. Learn.*, vol. 40, no. 3, pp. 203–228, 2000.
[59] B. S. Khater, A. A. Wahab, M. Idris, M. A. Hussain, and A. A. Ibrahim, "A lightweight perceptron-based intrusion detection system for fog computing," *Appl. Sci.*, vol. 9, no. 1, p. 178, 2019.
[60] Y. Chen and S. Li, "A lightweight anomaly detection method based on SVDD for wireless sensor networks," *Wireless Pers. Commun.*, vol. 105, pp. 1235–1256, Apr. 2019.
[61] S. Zaman and F. Karray, "Lightweight IDS based on features selection and IDS classification scheme," in *Proc. IEEE Int. Conf. Comput. Sci. Eng.*, vol. 3, Aug. 2009, pp. 365–370.
[62] *Cicids2017 Dataset*. Accessed: Jan. 20, 2019. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2017.html

SANA ULLAH JAN (S'17–M'19) received the B.S. degree in electronic engineering from International Islamic University, Islamabad, Pakistan, in 2012. He is currently pursuing the combined M.S./Ph.D. degree with the University of Ulsan, South Korea. He served as a Lab Engineer with the University of Lahore, Islamabad Campus, Islamabad, Pakistan, from 2012 to 2014. His research interests include wireless sensor networks (WSNs), visible light communications (VLCs), applications of Bluetooth Low Energy (BLE), and machine learning-based sensor fault detection, isolation, and diagnosis applications.

SAEED AHMED (M'18) received the B.E. and M.E. degrees in electrical engineering from the University of AJK, Pakistan, in 2005 and 2010, respectively. He is currently pursuing the Ph.D. degree with the University of Ulsan, South Korea. He served as a Transmission Engineer at telecom industry for eight years and has a vast experience in planning, surveying, and deploying microwave and optical-fiber-based core and access PDH/SDH/SONET/DWDM networks. He joined the Mirpur University of Science and Technology (MUST), Mirpur AJK, Pakistan, as an Assistant Professor, in 2012. His research interests include energy-efficient resource allocation in cognitive radios, smart grid (SG) communication technologies, SG cybersecurity, and the Internet of Things (IoT).

VLADIMIR SHAKHOV (M'09) received the B.S. degree in mechanics and applied mathematics, the M.S. degree in mathematics, and the Ph.D. degree in computer science from Novosibirsk State University, Novosibirsk, Russia, in 1994, 1996, and 2000, respectively. He has been a Senior Researcher with the Institute of Computational Mathematics and Mathematical Geophysics, since 2000. He was a Senior Researcher with the Samsung Advanced Institute of Technology, Suwon, South Korea, and a Senior Software Developer at Intel Corporation, Novosibirsk. He visited Sungkyunkwan University, South Korea, multiple times. From 2014 to 2016, he was an Associate Professor with the Siberian State University of Telecommunications and Information Sciences. He is currently a Research Professor with the University of Ulsan, South Korea. His research interests include system performance analysis, the IoT security, and data analytics. He is the Vice Chair of the IEEE Russian Siberia Section.

INSOO KOO received the B.E. degree from Konkuk University, Seoul, South Korea, in 1996, and the M.S. and Ph.D. degrees from the Gwangju Institute of Science and Technology (GIST), Gwangju, South Korea, in 1998 and 2002, respectively. From 2002 to 2004, he was with the Ultrafast Fiber-Optic Networks (UFON) Research Center, GIST, as a Research Professor. In 2003, he was a Visiting Scholar with the Royal Institute of Science and Technology, Sweden. In 2005, he joined the University of Ulsan, where he is currently a Full Professor. His research interests include next-generation wireless communication systems and wireless sensor networks.