Hindawi
Advances in Fuzzy Systems
Volume 2018, Article ID 4071851, 12 pages
https://doi.org/10.1155/2018/4071851

Research Article

A Lightweight Intrusion Detection Method Based on Fuzzy Clustering Algorithm for Wireless Sensor Networks

Hongchun Qu[1,2] **Libiao Lei**[1,2] **Xiaoming Tang**[1,2] **and Ping Wang**[1,2]

[1]*Key Laboratory of Industrial IOT and Networked Control (Chongqing University of Posts and Telecommunications), Ministry of Education, Chongqing 400065, China*
[2]*Chongqing Industrial Networking Collaborative Innovation Center, College of Automation, Chongqing University of Posts and Telecommunications, Chongqing 400065, China*

Correspondence should be addressed to Hongchun Qu; hcchyu@gmail.com

Received 30 November 2017; Revised 17 April 2018; Accepted 2 May 2018; Published 6 June 2018

Academic Editor: Yasar Becerikli

For resource-constrained wireless sensor networks (WSNs), designing a lightweight intrusion detection technology has been a hot and difficult issue. In this paper, we proposed a lightweight intrusion detection method that was able to directly map the network status into sensor monitoring data received by base station, so that base station can sense the abnormal changes in the network. Our method is highlighted by the fusion of fuzzy c-means algorithm, one-class SVM, and sliding window procedure to effectively differentiate network attacks from abnormal data. Finally, the proposed method was tested on the wireless sensor network simulation software EXata and in real applications. The results showed that the intrusion detection method in this paper could effectively identify whether the abnormal data came from a network attack or just a noise. In addition, extra energy consumption can be avoided in all sensor monitoring nodes of the sensor network where our method has been deployed.

1. Introduction

WSNs are a typical mobile and self-organized network constrained by energy supply, computing power, and communication capabilities of individual node. They are usually deployed in areas where power supply and maintenance are limited. The typical fields for deployment are traffic monitoring, environmental surveillance, and military purposes [1, 2]. Compared to resource-rich wired networks, nodes in WSNs are vulnerable to attacks [3]. However, due to the limitations of WSNs, relying solely on intrusion defense to defend against attacks is far from enough. There has been a consensus that intrusion detection technology should be incorporated to resist attacks in WSNs protection [4]. The intrusion detection technology, specifically, the intrusion detection system (IDS), is a network security technology originally built for detecting vulnerability exploits against a target application or computer [2]. The IDS is a listen-only device, which means that it monitors traffic and reports its results to an administrator, but cannot automatically take action to prevent a detected exploit from taking over the system.

How to save energy consumption of nodes participating in intrusion detection activities is a key issue in IDS [5]. In [6], the author proposed a cluster-based IDS, in which the entire network was divided into several subnetworks. One node was selected as a management node and was responsible for monitoring the member nodes in each subnetwork. Although this method saved the energy of ordinary nodes in the subnetwork, the energy consumption in management nodes was still remarkably high. In [7], the author proposed an agent-based distributed and collaborative IDS, in which agents were deployed on individual nodes to detect the network status of other nodes within the communication range. When an agent node detected an abnormal activity but cannot clearly clarify whether it was an exception, then the node communicated with the neighboring agent nodes to collaboratively analyze and finally confirm whether it was an anomaly. In this scheme, energy consumption occurred not only in network status monitoring, but also in communicating with neighboring agent nodes.

In this paper, in order to reduce the energy consumption of sensor nodes participating in intrusion detection activities, we presented a lightweight intrusion detection method based on Fuzzy Clustering Algorithm (LIDFCM) for resource-constrained WSNs. This method is different from the common intrusion detection algorithms which directly handle status of network traffic. The innovation is that we map the network status into the sensor measurements received by the base station, so that the abnormal change of measurements can reflect the change of the network status. In order to distinguish the source of abnormal data, our intrusion detection method can effectively identify potential attacks in abnormal data by the fusion of fuzzy c-means algorithm, one-class SVM, and sliding window procedure (the related operations of temporal correlation of sensor monitoring data [8]). It is an efficient and lightweight intrusion detection method.

The rest of this paper is organized as follows. Section 2 introduces related work of intrusion detection system. We introduce the model proposed in this paper in Section 3. Section 4 is simulation experiments. Results and discussion are shown in Section 5. Section 6 summarizes our work.

2. Related Works

Intrusion detection has been a hot and difficult problem in the research of WSNs. According to various classification criteria, intrusion detection technology can be classified into different categories. In this section, we review the mainstream categories of detection methodologies used in current intrusion detection systems, which are, respectively, anomaly-based detection, misuse-based detection, and specification-based detection [2].

2.1. Anomaly-Based Detection. Anomaly-based intrusion detection can be functionally categorized into three groups: statistical-based methods, knowledge-based methods, and machine learning based methods. Anomaly-based intrusion detection methods often require positive samples as training sets, to learn normal behavior patterns and establish normal activity models. When intrusion detection system detects that activities deviated significantly from the normal network behavior activities, a network attack much likely occurred. Common techniques based on anomaly detection include probabilistic statistics, data mining, neural networks, and others [2]. The advantage of anomaly-based intrusion detection is that it can detect unknown network attacks. However, the disadvantage is that the false positive rate is high [9]. For example, Doumit et al. [10] proposed an IDS based on self-organized criticality (SOC) and hidden Markov model (HMM), which can be regarded as anomaly detection. In the IDS, probability transition matrix used for HMM modeling was calculated by SOC. When the change of the network state was detected, the algorithm calculated the probability that the state-changing occurred under the normal behavior HMM and compared with the system predefined HMM threshold. If it was higher than the threshold, the IDS claimed that the state-changing was due to cyberattacks. Tian et al. [11] proposed an anomaly detection algorithm based on SVM and robust classification method to detect select forwarding attacks. In this model, SVM performed better than traditional machine learning methods that need large samples. In addition, it can find the best tradeoff between model complexity and learning ability given limited sample information, which brought the method a better generalization ability. Su et al. [6] proposed a clustering-based intrusion detection method, in which a group of member nodes monitored cluster head nodes and determined the abnormal behavior of cluster heads through alarm threshold (X). At the same time, cluster head nodes monitored the member nodes instead of deploying intrusion detection systems on each node to save sensor energy. However, this solution is subject to limitation that only static sensor networks can be applied.

2.2. Misuse-Based Detection. Misuse detection is based on the principle of pattern matching, in which the attack library established by collecting abnormal operations is used to detect network attacks. During detection, network activities undergo a matching process with the established network intrusion library. If the matching is successful, the network activity is considered as an intrusion and corresponding alarm information is then generated [2]. Misuse detection can often achieve a high detection rate, but only works in the condition where attack patterns are previously established in the library, which means that some omissions are inevitably expected [12, 13]. Common misuse detection techniques include Petri net analysis, neural networks, expert systems, state transition analysis, and feature analysis. For example, Da Silva et al. [14] deployed the intrusion detection system on sensor nodes. The system monitored the behavior of the attacked nodes and established attack rules. For example, retransmission rule was mainly used to detect black hole attacks and selective forwarding attacks. The experimental results showed that this method was very effective in detecting the corresponding attacks defined in the rules, but less effective in detecting attacks excluded by the rules. Han et al. [15] proposed a data mining tool based network attack feature extraction (SigSniffer). The tool firstly captured packet conditions in the attacked network through Packet Sensor, then performed simple processing, and finally sent it to Signature Miner for initial mining where potentially useful rules were stored in the Signature Set. Associated Signature Miner further mined features stored in the Signature Set by using association analysis, which were added to the Rule Set for intrusion detection. The authors claimed that the feature extracted by their methods had a high degree of recognition. Cheng et al. [16] proposed a distributed intrusion detection system based on rule matching for industrial control systems. The network probes were distributed into an industrial network to collect information of packets around its domain and send them to the corresponding detection module. The detection module performed in-depth extraction of features, and then compared it against the black and white lists extracted from known attacks to identify potential cyber-attacks. However, each detection module was independent in their method, thus ignoring of attacks that were multiple subnet relevant might be expected.

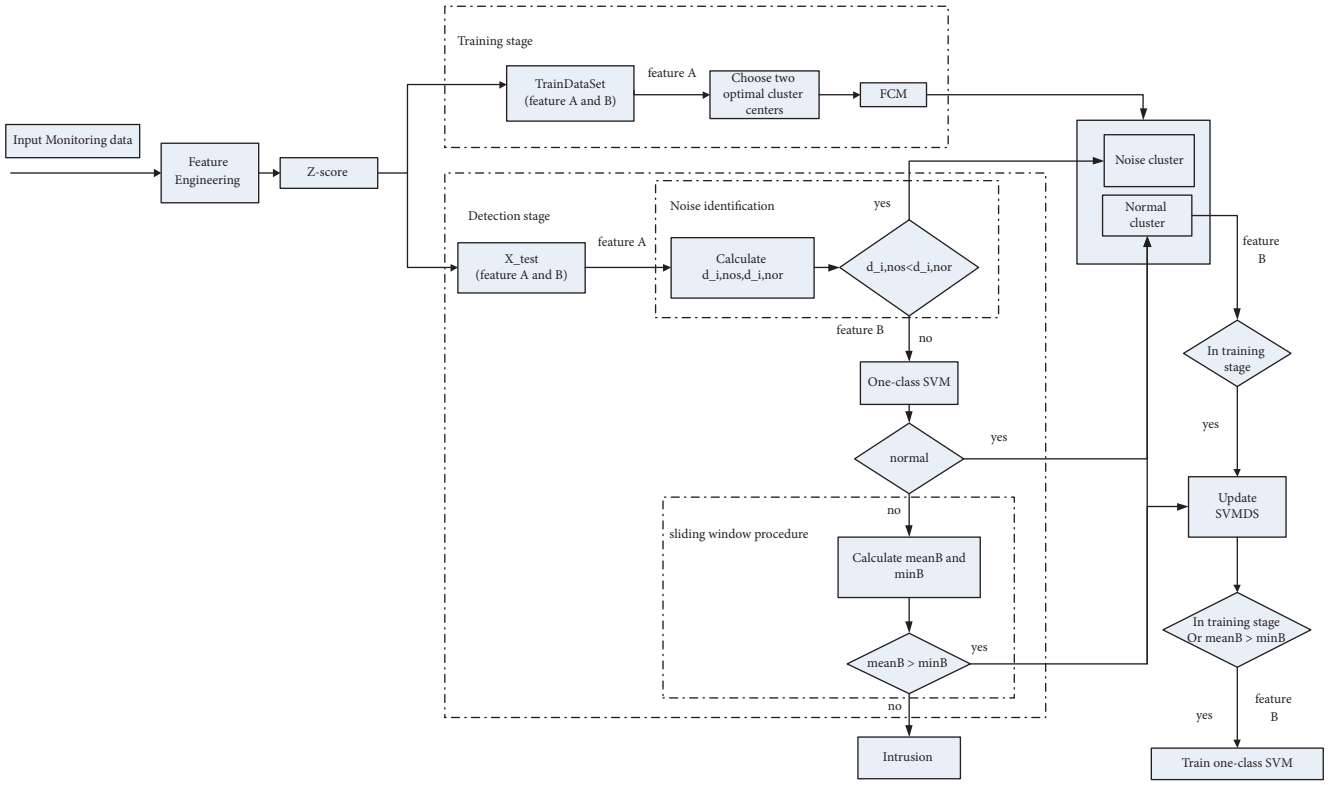


Figure 1: Structure diagram of the proposed intrusion detection model.

2.3. Specification-Based Detection. The basic idea of specification-based detection method is to use a set of specification and constraints to describe the correct operation of a program or protocol, which has been usually employed to monitor the behavior of a system or a network [2]. The way of establishing a normal behavior pattern in specification-based intrusion detection is different to that of anomaly-based intrusion detection, which is obtained through the actual operation of objects and continuously revised through repeated training. However, specification-based intrusion detection obtains the pattern by artificially abstracting the expected behavior of the objects. When object behavior deviates significantly from the abstract behavior, it is considered that an abnormality has occurred [17]. Specification-based intrusion detection has the merit of both misuse-based and anomaly-based detection; that is, on the one hand, it is able to detect unknown attacks and the false positive alarm rate is lower than that of anomaly detection-based technologies. On the other hand, it has a considerable high detection rate for known attacks [18]. The reliable performance of specification-based detection is at the cost of computation, that is, a set of specification and constraints need to be defined [19–28]. For example, Le et al. [27] proposed a specification-based IDS for attacks targeting routing protocols for low power and lossy networks (RPL). The intrusion detection system defined the legitimate operations of the RPL protocol to monitor the system's various behaviors. Experimental results showed that the intrusion detection system had a higher detection rate and a lower false positive rate. In [28], the author proposed a specification-based intrusion detection system for the medical cyberphysical system (MCPS). In this intrusion detection system, the author defined the behavioral guidelines for the normal operation of devices, then translated them into a normal machine state, and inputted into the model for training. During the operation of devices, their states were always monitored by the intrusion detection system. When devices' behavior deviated from the predefined state, an alarm was triggered. Results in comparison with another similar two anomaly-based detection schemes showed a better detection rate.

3. The Model

In this section, we detail the main steps adopted in our model that are, respectively, the feature engineering, the training phase of fuzzy c-means clustering algorithm (FCM), the selection of optimal clustering center and one-class SVM model, and the detection stage of noise identification, the sliding window procedure of network attack identification, as shown in Figure 1.

3.1. Feature Engineering and Z-Score. We assume that a wireless sensor network contains N monitoring nodes. Each node in the network sends a set of collected monitoring data, that is, measurements (such as temperature, humidity, and voltage) to the base station in a fixed time interval. Some data packets may be lost due to various reasons,

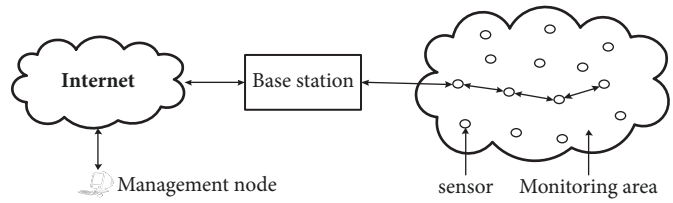


Figure 2: Wireless sensor network architecture diagram.

such as network congestion, sensor node failure, or network attacks. We assume that the packet loss caused by non-network attacks is within a reasonable range. Base station actually receives s group of data packets at each time interval: $Q = \{Q_1, Q_2, \ldots, Q_s\}$. Each group of data has u-dimensional attributes. The typical structure of a wireless sensor network is shown in Figure 2.

A common form of network intrusion in sensor networks is that attacked nodes discard packets (or a large number of invalid packet is accumulated in the network to cause network congestion). As a result, the number of measurements received by the base station is far less than the actual number of nodes but has no effect on the value of measurements. This is the reason why most anomaly detection algorithms characterized by measurements cannot identify communication destructive network attacks. We map the network traffic conditions into monitoring data in base station. When the network conditions change significantly, they are automatically reflected in the monitoring data, so that the base station can perceive the changes in the network status. However, the change in monitoring data is also likely caused by noise data. Therefore, we define detection feature A and detection feature B down below to solve this problem.

Definition (detection feature A). The standard deviation of each dimension attribute of a measurement at each time interval $A = \{\sigma_1, \sigma_2, \ldots, \sigma_u\}$.

$$\sigma_z = \sqrt{\frac{\sum_{k=1}^{s} (Q_{zk} - g_z)^2}{s}}, \quad z = \{1, 2, \ldots, u\} \tag{1}$$

where g_z is the mean of each dimension attribute of the measurement at each time interval. $g_z = \sum_{k=1}^{s} Q_{zk}/s$, $z = \{1, 2, \ldots, u\}$.

Definition (detection feature B). A modified from g_z, $B = \{e_1, e_2, \ldots, e_u\}$.

$$e_z = \frac{\sum_{k=1}^{s} Q_{zk}}{N}, \quad z = \{1, 2, \ldots, u\}. \tag{2}$$

Since the standard deviation of a set of measurements containing noise is expected to be significantly different to the standard deviation of a set of measurements without noise, the detection feature A is able to distinguish between the noise and nonnoise data. The impact of most attacks on a wireless sensor network is reflected in the reduction in the number of data packets. Therefore, the standard deviation of normal measurements and the standard deviation of measurements containing cyberattacks in each time interval are much likely similar, which means that the application of feature A alone is not able to distinguish between normal data and intrusion. To bridge this gap, the network traffic relevant feature, i.e., feature B, is applied to recognize network intrusion from normal data. The usage of feature A and feature B is shown in Figure 3.

In a sensor network, the ranges of monitoring data from which different objects are sensed are usually different. If the input features of a machine learning algorithm have different range and are not normalized, the convergence speed of the algorithms is significantly slow for most machine learning algorithms. In addition, the prediction accuracy is also decreased. This paper adopts Z-score normalization (formula (3)) to standardize the data after feature engineering.

$$y_{zk} = \frac{Q_{zk} - g_z}{\sigma_z} \tag{3}$$

3.2. Training Stage

3.2.1. Fuzzy c-Means Clustering Algorithm and Selection of Optimal Clustering Center. Given n sample points $X=\{x_1, x_2, \ldots, x_n\}$, in the feature space, the standard FCM algorithm [29] classifies X into c fuzzy cluster groups as follows.

Step 1. Initialize the acceptable error of objective function ε and the fuzziness index m.

Step 2. Randomly initialize the membership matrix $U=[u_{ji}] \in R^{cn}$, where u_{ji} represents membership from sample point i^{th} (i=1,2,. . .,n) to the j^{th} (j = 1,2,. . .,c) cluster center. Membership must satisfy the following conditions:

$$\sum_{j=1}^{c} u_{ji} = 1, \quad \forall i = 1, 2, \ldots, n; \; u_{ji} \in [0, 1], \; \forall i, j \tag{4}$$

$$\sum_{i=1}^{n} u_{ji} > 0, \quad \forall j = 1, 2, \ldots, c \tag{5}$$

Step 3. For the k^{th} iteration, update the clusters centers using (6) and update the partition matrix using (7).

$$c_j = \frac{\sum_{i=1}^{n} {u_{ji}}^m x_i}{\sum_{i=1}^{n} {u_{ji}}^m} \tag{6}$$

$$u_{ji} = \frac{1}{\sum_{k=1}^{c} \left(\left\| x_i - c_j \right\| / \left\| x_i - c_k \right\| \right)^{2/(m-1)}} \tag{7}$$

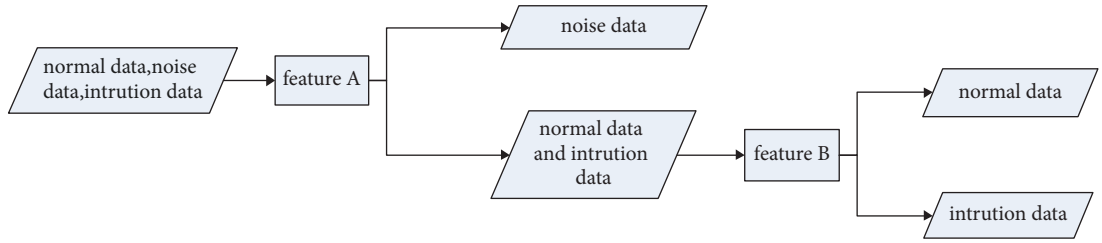


Figure 3: The function of feature A and feature B.

where c_j represents the j^{th} cluster center; $\|x_i - c_j\|$is the Euclidean distance between the i^{th} sample point and the j^{th} cluster center.

Step 4. For the k^{th} iteration, calculate the value of the objective function using (8). If $|J^{(k)} - J^{(k-1)}| \leq \varepsilon$, then stop. Otherwise continue doing Steps 3 and 4.

$$J = \sum_{i=1}^{n}\sum_{j=1}^{c}\left(u_{ji}\right)^{m}\left\|x_i - c_j\right\|^2, \quad 1 \leq m < \infty \tag{8}$$

In our model, the FCM is used to cluster the data generated in the initialization stage of the sensor network. The advantage is that it is not necessary to manually label the data as a training set to train the system. Considering the sensitivity of the standard FCM to center location, we adopt the approach proposed by Rodriguez et al. [30] to choose local cluster centers, in which two assumptions are made: (1) a cluster center itself has a higher density than the surrounding neighbors, and (2) the distance between one high-density point and other high-density points is relatively large. Therefore, we choose local cluster centers based on the criteria that a point is likely to be a cluster center if its density is larger than all points within the radius of the d_c; then it is labeled as the local cluster center. For feature space containing n sample points $X=\{x_1, x_2, \ldots, x_n\}$, the density of a point x_i can be defined in

$$\rho_i = \sum_{j=1}^{n}\exp\left(-\left(\frac{\left\|x_i - x_j\right\|}{d_c}\right)^2\right) \tag{9}$$

where d_c is the cutoff distance. In this context, we calculate the distance ($\|x_i - x_j\|$) between any two points and sort these distances from small to large. d_c is the value of the first q percentile of the sorted distance ($q\epsilon[1, 2]$) [30].

Since detection feature A is calculated as the standard deviation of the measurements, the global noise cluster center is selected as the center with the largest standard deviation, while the global normal cluster center should be the one with the smallest standard deviation. Then the noise and the normal cluster center are used as the initial cluster center of the FCM algorithm. The membership matrix is calculated using the two cluster centers, which are used to establish the normal and noise cluster with the assistance of FCM and feature A. Detection feature B of the normal cluster is inputted into the support vector machine data set (named SVMDS) and used to train the one-class SVM model.

3.2.2. One-Class SVM. The one-class SVM [31] is essentially an unsupervised classification method. The method firstly performs statistical modeling by mapping normal data into high-dimensional space H ($x \rightarrow \emptyset(x)$) through a kernel function in the training stage, then establishes a hyperplane that separates the normal data from the abnormal ones (the origin in high-dimensional feature space) as much as possible. This process aims at maximizing the distance between nonnormal data and the hyperplane while trying to identify the normal data as much as possible. Given the hyperplane $\omega \cdot \emptyset(x) - b = 0$, the quadratic programming problem that needs to be solved for the training sample points $X=\{x_1, x_2, \ldots, x_l, \}$, $x_i \in R^l$ is

$$\min_{\omega, b, \varepsilon_i} \quad \frac{1}{2}\|\omega\|^2 + \frac{1}{vl}\sum_{i=1}^{l}\varepsilon_i - b \tag{10}$$

$$\text{s.t.} \quad \left(\omega \cdot \emptyset\left(x_i\right)\right) \geq b - \varepsilon_i, \quad \varepsilon_i \geq 0, \ i = 1, 2, \ldots, l. \tag{11}$$

where ε_i is a relaxation variable whose role is to penalize the misclassified sample points; $v\epsilon(0, 1)$ controls the distance between aggregation area of sample points and the maximum-interval hyperplane, as well as the distance between the nonnormal data and the maximum-interval hyperplane.

Next, the Lagrange multiplier method is used to transform the quadratic programming problem into the optimization problem of solving dual variables. The Lagrange function is constructed as follows:

$$\begin{aligned} L\left(\omega, b, \varepsilon, \alpha, \gamma\right) = {} & \frac{1}{2}\|\omega\|^2 + \frac{1}{vl}\sum_{i=1}^{l}\varepsilon_i - b \\ & - \sum_{i=1}^{l}\alpha_i\left[\left(\omega \cdot \emptyset\left(x_i\right)\right) - b + \varepsilon_i\right] \\ & - \sum_{i=1}^{l}\gamma_i\varepsilon_i \end{aligned} \tag{12}$$

where $\alpha_i \geq 0$ and $\gamma_i \geq 0$ are Lagrange factors. Letting the partial derivatives of L with respect to ω, ε_i, b to be zero, we have

$$\omega = \sum_{i=1}^{l}\alpha_i\emptyset\left(x_i\right) \tag{13}$$

$$\sum_{i=1}^{l}\alpha_i = 1 \tag{14}$$

$$\alpha_i = \frac{1}{vl} - \gamma_i \tag{15}$$

Substituting (13), (14), (15) into (12), we get the dual problem of quadratic optimization problem as

$$\min_{\alpha_i} \quad \left(\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j K\left(x_i \cdot x_j\right)\right) \tag{16}$$

$$\text{s.t.} \quad 0 \le \alpha_i \le \frac{1}{vl}, \ \sum_{i=1}^{l}\alpha_i = 1 \tag{17}$$

where $K(x_i \cdot x_j)$ is kernel function. Support vectors (x_k) are defined as training sample points with $\alpha_i > 0$; the hyperplane parameter b can be expressed

$$\mathrm{b} = \sum_{i=1}^{l}\alpha_i K\left(x_i, x_k\right) \tag{18}$$

In summary, the decision function (f(x)) is defined as

$$\mathrm{f}\left(\mathrm{x}\right) = \mathrm{sgn}\left(\sum_{i=1}^{l}\alpha_i K\left(x_i, \mathrm{x}\right) - b\right) \tag{19}$$

The decision function is used in the detection stage. The decision function takes a test sample point x as input, if f(x)> 0, the sample is classified as the normal (no intrusion presents), otherwise it is regarded as the nonnormal, which means that the test sample might or might not contain intrusion. The latter requires further process to identify. They are described in detail in the next stage.

In the training stage, the optimal global cluster center is selected by the idea of density algorithm, and the FCM is initialized. Then training set is divided into normal clustering and noise clustering by the FCM, and the intrusion detection identifier one-class SVM is trained by normal clustering. The training stage is shown in Algorithm 1.

3.3. Detection Stage. The first step is the identification of noise data in detection stage. The distance from each sample entry $\mathrm{x}_{\mathrm{new}}$ to the normal cluster (d_i, nor, see (20)) and the noise cluster (d_i, nos, see (21)) are, respectively, calculated by the detection feature A. If the former is greater than the latter, $\mathrm{x}_{\mathrm{new}}$ is noise. Otherwise, the detection feature B of $\mathrm{x}_{\mathrm{new}}$ is input of one-class SVM model. If the result is normal, $\mathrm{x}_{\mathrm{new}}$ is normal. But if it is determined as nonnormal, i.e., $\mathrm{x}_{\mathrm{new}}$ is suspicious (named $\mathrm{x}_{\mathrm{suspicious}}$); then we make the final decision on $\mathrm{x}_{\mathrm{suspicious}}$ with sliding window procedure.

$$\mathrm{d\_i, nor} = \frac{\sum_{\mathrm{k}=1}^{\mathrm{n1}}\left\|x_i - x_{nork}\right\|}{n1} \tag{20}$$

$$\mathrm{d\_i, nos} = \frac{\sum_{\mathrm{k}=1}^{\mathrm{n2}}\left\|x_i - x_{nosk}\right\|}{n2} \tag{21}$$

where i is the i^{th} entry to be detected; *nor* is normal cluster; *nos* is noise cluster; *nork* is the k^{th} element in the normal cluster; the *nosk* is the k^{th} element in the noise cluster; *n1* is the sum of the numbers of all elements in the normal cluster; and *n2* is the sum of the numbers of all elements in the noise cluster.

Sliding window procedure uses the temporal correlation between monitoring data to clarify suspicious samples. Temporal correlation of monitoring data means that the values of measurements obtained in adjacent moments are typically consistent; i.e., the mean or variance of measurements tends to be temporally correlated. Sliding window procedure is defined as the following four steps. Firstly, we define a sliding window p, which represents p records from $\mathrm{x}_{\mathrm{suspicious}}$ to the first p records of feature B in X. For example, if x_{10} is suspicious, the sliding window p includes entries $\{\mathrm{x}_{(10-\mathrm{p})}, \mathrm{x}_{(10-\mathrm{p}+1)}, \ldots, \mathrm{x}_9\}$. Secondly, we choose the normal samples from these records named normalB and find the minimum of the first dimension value of these normal samples named minB. Thirdly, the average value, meanB, calculated the first dimension values between $\mathrm{x}_{\mathrm{suspicious}}$ and these normal samples. Finally, If meanB is greater than or equal to minB, $\mathrm{x}_{\mathrm{suspicious}}$ is predicted as normal; otherwise $\mathrm{x}_{\mathrm{suspicious}}$ is classified as an attack. Sliding window procedure is given in Algorithm 2.

When the sliding window procedure determines that the suspicious sample is normal data, it indicates that the environment is slowly changing. The SVMDS needs to be updated and the one-class SVM model is retrained to adapt to changes occurred in the environment. Specifically, the $\mathrm{x}_{\mathrm{new}}$ is appended to SVMDS and the first entry of SVMDS is discarded to keep the total length of the SVMDS unchanged. Finally, the one-class SVM is retrained using the updated SVMDS.

4. Simulation Experiments

We simulated a sensor network with artificial attacks on EXata [32], a wireless sensor network simulation software platform, to test the proposed method. We choose EXata because its considerable accuracy compared with real network environments. The network environment simulated by EXata ran on a personal computer with an Inter (R) Core i5-6300HQ, 2.30GHz, 8GB memory (RAM). The wireless sensor network was deployed by randomly placing 50 sensor nodes (10% of which were randomly set as attack sources) in an area of 100 (m) x 100 (m) and one base station node (Figure 4, node 51 is the base station). The MAC layer and routing protocols of all the nodes were IEEE802.11 and AODV (ad hoc on-demand distance vector routing), respectively. The total time setting of each simulation run was 15000 seconds, in which the sensor node transmitted three types of measurement data (temperature, humidity, and voltage) to the base station in wireless multihop manner at an interval of 10 seconds. The three types of environmental measurements were collected from the Berkeley Research Laboratory (IBRL, Intel Berkeley Research Laboratory) [33]. Black hole attacks and flood attacks were selected for network attacks. The attack ran time was 3000 to 4000 seconds and 6000 to 7000 seconds,

Input: Training set X={$x_1, x_2, \ldots, x_n$, },q
Output: Normal clustering, noise clustering and one-class SVM model
(1) use feature A of X
(2) lc = [] /∗ storage local cluster center
(3) gnorc = 0 /∗global normal cluster center
(4) gnosc = 0 /∗ global noise clustering center
(5) $k = 0$
(6) calculate d = $\{d_{i,j}\}_{n*n}$ /∗ calculate the distance between any two points
(7) d_c = cutoffdis(q) /∗ cutoff distance, equation (9)
(8) $\rho = (\rho_i)_n$ /∗ equation (9)
(9) **for** $i = 1$ to n **do**
(10) find ρ_{tmpMax} in the field of x_i /∗the field is a circle with radius d_c and center x_i
(11) **if** $\rho_i = \rho_{tmpMax}$ **then**
(12) lc.append(x_i) /∗ x_i is a local cluster center and stored
(13) $k = k+1$
(14) **end if**
(15) **end for**
(16) **for** $i = 1$ to k **do**
(17) **if** x_i is min **then** /∗ x_i in lc
(18) gnorc = x_i
(19) **end if**
(20) **if** x_i is max **then** /∗ x_i in lc
(21) gnosc = x_i
(22) **end if**
(23) **end for**
(24) initialize FCM with gnorc and gnosc
(25) X is divided into normal clustering data and noise clustering data by the FCM
(26) use feature B of normal clustering data
(27) input feature B of normal clustering data to SVMDS
(28) train one-class SVM by SVMDS
(29) **return** normal clustering, noise clustering and one-class SVM model

Algorithm 1: **Training Stage**.

Input: Feature B of suspect sample $x_{suspicious}$, size of sliding window p
Output: Category of the suspect sample $x_{suspicious}$
(1) extract first p record data of the suspect sample
(2) use first dimension value of the record data and $x_{suspicious}$
(3) select feature B of predicted normal data from the record data (named normalB, the size is k)
(4) **for** $i = 1$ to k **do**
(5) find minB /∗ the smallest value of feature B of predicted normal data
(6) **end for**
(7) $$\text{meanB} = \frac{(\text{normalB} + x_{suspicious})}{k+1}$$
(8) **if** meanB ≥ minB **then**
(9) the $x_{suspicious}$ is normal
(10) **else then**
(11) the $x_{suspicious}$ is a network intrusion sample
(12) **end if**
(13) **return** category of the suspect sample $x_{suspicious}$

Algorithm 2: Sliding window procedure.

respectively. The simulation was replicated for 10 times, in which the location of each sensor node was randomly set.

In order to verify the validity of LIDFCM proposed in this paper, support vector machine (SVM), Bayesian algorithm (Naïve Bayes), and PCACID algorithm proposed in [34] were compared with our method. The training set and test set were the data of the IBRL data set after passing the EXata platform and calculated the detection features A and B, including normal data, noise data, and network attack data. The SVM ("rbf", C=1, $\delta = 0.5$) and Naïve Bayes were

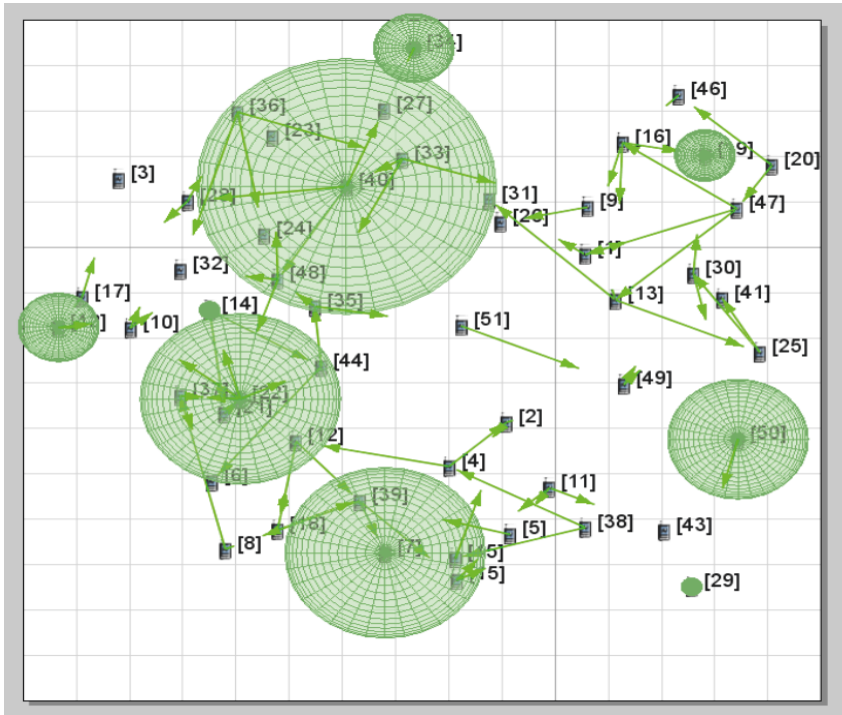


Figure 4: EXata sensor network simulation.

provided by the Python sklearn library. PCACID was written in Python 3.6. Because SVM model and Naïve Bayes model were supervised learning algorithm, the training set of SVM and Naïve Bayes algorithm contained normal data, noise data, and partial network attack data. The training set of our model contained only normal data and noise data. The PCACID algorithm was similar to one-class SVM and the training set contained only normal data. The number of training sets of four algorithms was respectively 200, and the rest were test sets contained normal data, noise data, and network attack data. All algorithms ran on Intel(R) Core(TM) i5-6300HQ, 2.30GHz, 8GB memory (RAM) personal mobile computers. The specific environment was the PyCharm integration software running on win10.

In addition, we tested our method on the IPv6-based wireless sensor network verification platform to verify the practical value of our method. The verification platform, a centralized deployment mode as shown in Figure 5(a), was developed by another work group in the laboratory based on configuration software and ran in the Windows 7 environment. Each laboratory deployed two sensor nodes, where one node was located in the laboratory near the window, and the other was in the laboratory near the aisle, such as the red triangle in Lab_9. Nodes were placed between four laboratories in the aisle, such as the red square between Lab_8, Lab_9, Lab_10, and Lab_11. The verification platform had a total of 40 sensor nodes with communication modules (each node had temperature, humidity, and voltage sensors), where the MAC layer and routing protocols were IEEE802.11 and AODV. Gateway node was, at Lab_0, responsible for collecting measurements of each sensor. Then the measurements were sent verification platform monitoring located in Lab_0.

Table 1: Utility function parameters.

parameters	description
UF	utility function value
DR	Detection rate
FAR	False alarm rate
FPR	False positive rate
ρ	$\rho = 0.75$
β	$\beta = 1$
θ	$\theta = 1$

In verification, the sensor node was networked through the verification platform. After all the nodes were successfully networked, gateway node collected measurements every 10 seconds. Then it was sent to the verification platform and stored in Mysql database for intrusion detection by our method. The total time setting of each run was 18000 seconds, where the attack time of black hole attacks and flood attacks was 5000 to 6000 seconds and 9000 to 10000 seconds respectively. The verification was replicated for 3 times. Figure 5(b) was the management interface of the verification platform and (c) was an illustration of the verification platform.

The utility function (UF) [35, 36] is used to evaluate our method using (22). Table 1 is the relevant parameters.

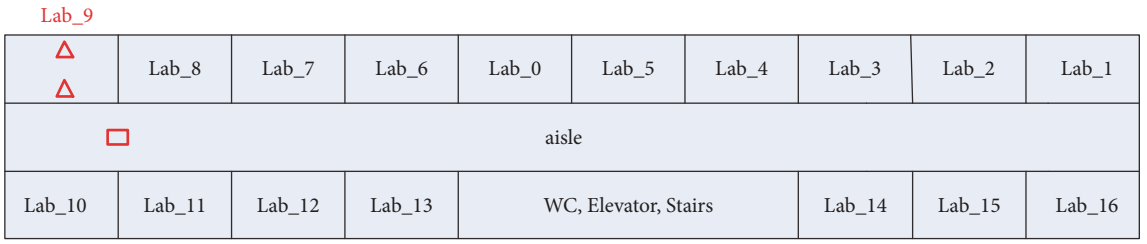
$$\mathrm{UF} = \rho * \mathrm{DR} - \beta * \mathrm{FAR} - \theta * \mathrm{FPR} \tag{22}$$

5. Results And Discussion

In EXata's virtual wireless sensor network, the average results of the abnormal data detection utility values of the four algorithms are shown in Table 2.

Table 2: Anomaly detection average UF of SVM, Naïve Bayes, PCACID and LIDFCM.

model	SVM	Naïve Bayes	PCACID	LIDFCM
UF	73.82%	71.86%	72.64%	74.80%



(a)

(b)

(c)

Figure 5: The IPv6-based wireless sensor network verification platform, where (a) is the deployment of the IPv6-based wireless sensor network verification platform, (b) is the management interface of the verification platform, and (c) is an exhibition of the verification platform.

In the result, we can see that the utility value of LIDFCM is 74.80%, which is better than 73.82% of SVM, 71.86% of Naïve Bayes, and 72.65% of PCACID. The reason why the detection efficiency of our method is relatively good is that, on one hand, the detection of feature A helps to reduce the impact of noise on network attack detection, because feature A makes the normal data and network attack data very similar, helping to reduce influence of noise data. On the other hand, sliding window procedure applied in the detection stage helps to improve the detection rate of normal data. In addition, most of the traditional traffic-based network intrusion detection algorithms directly classify anomaly data as network intrusion, but sometimes noise data is hidden behind. Therefore, these methods are not suggested to be directly applied to intrusion detection of measurements. However, our method is able to detect network attack from measurements with noise. Overall, the average UF value of our method for network attack detection is 74.61%.

The sliding window procedure is mainly to use the temporal correlation of measurements to further process suspicious samples. The sliding window p in sliding window procedure is closely related to the temporal correlation of the monitoring data, and the size of p is a crucial factor that powers the efficiency of our method. However, correctly choosing appropriate sliding window size p is tricky and application specific. On one hand, it obviously does not meet the definition of temporal correlation of measurements that is a too large p value. On the other hand, a too small p value might seriously decay the performance of the detection system. For example, a suspicious entry x_{new} could be recognized as a network attack when the previous entry is noise and the value of p is 1. Therefore, it is necessary to analyze the impact of the size of sliding window p on the detection efficiency. Figure 6 shows the effect of different value of p on average UF. When temporal correlation of measurements is employed (p=1), the increasing UF is 3.73% for anomaly detection, and that is 3.72% for intrusion detection. When p is increased to 5, the UF value of anomaly detection reaches 74.80%, and the UF value of intrusion detection is 74.61%. When p was larger than 5, they remain relatively stable with the increase of p.

In this paper, the time interval for the base station to receive monitoring data is 10 s, and five historical values are extracted (total time=sliding window∗time interval=50 seconds). During this time interval, changes in temperature, humidity, and voltage are very small. That is to say, for environmental monitoring objects, it is reasonable to use the total time of 50 seconds as the adjacent time in the definition of temporal correlation. It is helpful to use sliding window procedure in our method. What is more, the computational complexity of introducing sliding window procedure is O(n), where n is the number of suspicious samples of detection in the one-class SVM model. Overall, the computational

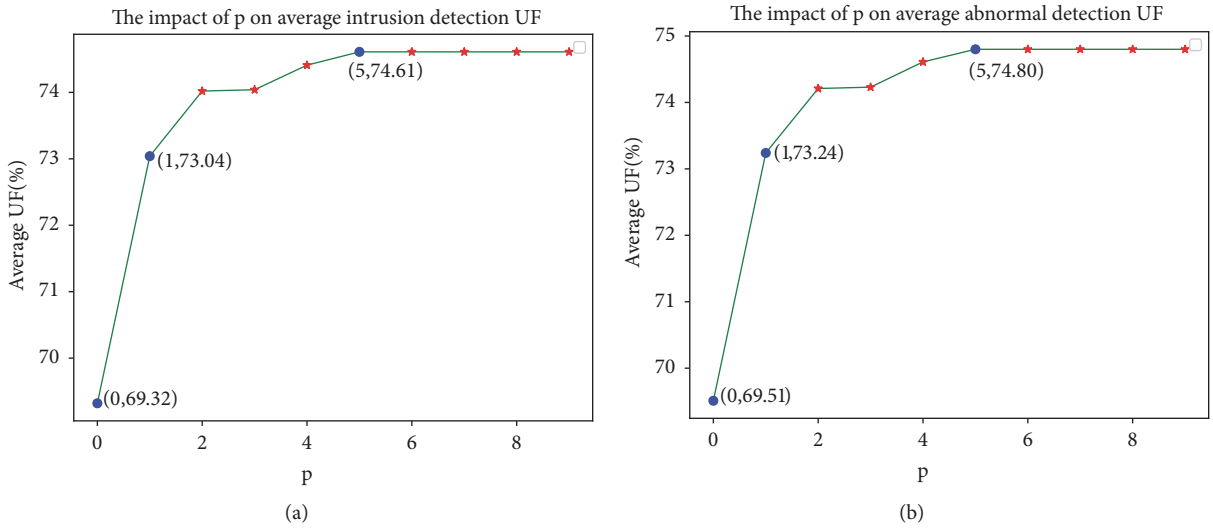


FIGURE 6: Effects of sliding window size p on the detection UF, where (a) is the impact of the value of p on the intrusion detection, which is measured as the utility function value. (b) is the effect on average abnormal detection UF value.

complexity of the method is not significantly increased when the sliding window process is introduced. Therefore, considering the temporal correlation of monitoring data and detection effect, the final value of the sliding window p chosen in this paper is 5.

In different application scenarios, the size of the sliding window p should be dynamically adjusted according to the time interval size and detection effect. The principle is to meet the definition of temporal correlation in the application scenario, that is, the monitoring data in the sliding window does not change much. Next, select the window p with the largest detection utility value or the smallest window p with the same utility value.

Although introducing the sliding window procedure does not significantly increase the computational cost, a better parameterized one-class SVM is very helpful to reduce the number of calls for sliding window procedure, which made our method more attractive. Figure 7 shows the relationship between the number of calls for sliding window procedure and the parameter v of the one-class SVM (see (10)). When a sample entry to be tested is regarded as suspicious by the one-class SVM model, the detection system calls the sliding window procedure to further investigate. Therefore, the result of the one-class SVM directly affects the number of calls for the sliding window procedure. The parameter v is very sensitive to model for different training set. When $v = 0.02$, 0.03 or 0.04, the number of calls for sliding window procedure is the minimized, which is approximately 28. When $v = 0.5$, which has been used by a lot of similar algorithms, the number of calls for sliding window procedure is 377. Therefore, we suggest that for different datasets, one can adjust the parameters v of one-class SVM by using different parameter optimization method, such as common function of GridSearchCV, and apply the optimal one-class SVM to our method in order to reduce the calls for sliding window procedure.

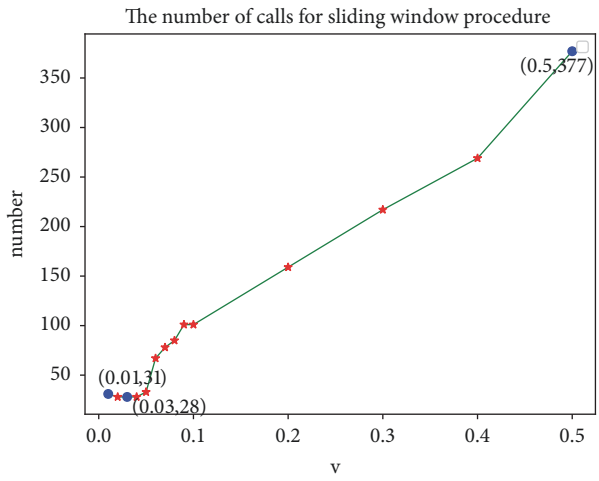


FIGURE 7: The number of calls for sliding window procedure is affected by the one-class SVM parameter v.

The updates of SVMDS and one-class SVM model also affect the number of calls for sliding window procedure. In general, temporal correlations of data entries in measurements apply only in a small time scale. When new samples keep coming, they are much likely to be deviated from the historical samples stored in the SVMDS. These new samples are treated as abnormal due to violation of temporal correlation, which inevitably causes calls of sliding window procedure. Experimental results show that when the SVMDS is not updated, the number of calls for sliding window procedure is 28. After updating SVMDS and retraining the one-class SVM model, the calls is 12 times, which significantly improves the efficiency of the one-class SVM detection.

In verification stage of the IPv6-based wireless sensor network verification platform, experimental results show that the UF both the anomaly detection and intrusion detection in

the real data set are 74.77%, which is very close to the result of the EXata simulation. It not only shows that the reliability of the EXata simulator, also demonstrates the efficiency of our method.

6. Conclusion

In this paper, we proposed a lightweight intrusion detection method for wireless sensor networks. Our approach is a two-stage sensor network intrusion detection method. In the training stage, we use density-based approach to find high-quality local density cluster centers, from which the global optimal normal cluster center and noise cluster center are selected. Then the FCM is initialized to generate the normal cluster and noise cluster, in which the normal cluster and feature B are used as input to the one-class SVM for training. Finally, the possible intrusion is detected by the one-class SVM and sliding window procedure by using feature B. Experiments show that our method can not only identify anomalies, but also identify whether anomalies are network attacks, which is more practical than most anomaly detection algorithms. In addition, the intrusion detection system is deployed at the base station, which does not require sensor nodes to send detection features, achieving high efficiency of detection and prolonging the lifespan of battery powered sensor networks. At the same time, we combine the sensor monitoring data received by the base station at each time interval into one data. In the age of the Internet of Everything where the data volume is exploding, this solution can effectively reduce the training difficulty of the model.

In spite of positive results, our method still needs to be improved in the diversity of intrusion detection. The intrusion detection method proposed in this paper is very effective for communication destructive network attacks and needs to be gradually improved in order to detect multiple attacks in the future.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

Funding was received by the Chongqing Research Program of Basic Research and Frontier Technology (cstc2017jcyjAX0453 and cstc2015jcyjA40007).

References

[1] C. Chong and S. P. Kumar, "Sensor networks: evolution, opportunities, and challenges," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1247–1256, 2003.

[2] I. Butun, S. D. Morgera, and R. Sankar, "A survey of intrusion detection systems in wireless sensor networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 266–282, 2014.

[3] A. Perrig, J. Stankovic, and D. Wagner, "Security in wireless sensor networks," *Communications of the ACM*, vol. 47, no. 6, pp. 53–57, 2004.

[4] T. Anantvalee and J. Wu, "A Survey on Intrusion Detection in Mobile Ad Hoc networks," in *Wireless Network Security*, pp. 159–180, Springer, 2007.

[5] S. Rajasegarar, C. Leckie, M. Palaniswami, and et al, "Distributed Anomaly Detection in Wireless Sensor Networks," in *Proceedings of the IEEE Singapore International Conference on Communication Systems (IEEE '06)*, 2006.

[6] C. C. Su, K. M. Chang, Y. H. Kuo, and M. F. Horng, "The new intrusion prevention and detection approaches for clustering-based sensor networks," in *Proceedings of the IEEE Wireless Communications and Networking Conference*, New Orleans, La, USA, 2005.

[7] Y. Zhang and W. Lee, "Intrusion detection in wireless ad-hoc networks," in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom '00)*, pp. 257–283, ACM, Boston, Mass, USA, August 2000.

[8] S. Kamal, R. A. Ramadan, and F. El-Refai, "Smart outlier detection of wireless sensor network," in *Proceedings of the International Conference on Recent Advances in Computer Systems*, Hail, Saudi Arabia, November 2015.

[9] R. Mitchell and I.-R. Chen, "A survey of intrusion detection techniques for cyber-physical systems," *ACM Computing Surveys*, vol. 46, no. 4, article 55, 2014.

[10] S. Doumit and D. Agrawal, "Self-organized criticality & stochastic learning based intrusion detection system for wireless sensor networks," in *Proceedings of the IEEE Military Communications Conference, 2003. MILCOM 2003.*, pp. 609–614, Boston, MA, USA.

[11] J. Tian, M. Gao, and S. Zhou, "Wireless sensor network for community intrusion detection system based on classify support vector machine," in *Proceedings of the IEEE International Conference on Information and Automation (ICIA '09)*, pp. 1217–1221, Zhuhai, China, June 2009.

[12] R. A. Kemmerer and G. Vigna, "Intrusion detection: A brief history and overview," *The Computer Journal*, vol. 35, pp. 27–30, 2002.

[13] T. S. Sobh, "Wired and wireless intrusion detection system: classifications, good characteristics and state-of-the-art," *Computer Standards & Interfaces*, vol. 28, no. 6, pp. 670–694, 2006.

[14] A. P. R. Da Silva, M. H. T. Martins, and B. P. S. Rocha, "Decentralized intrusion detection in wireless sensor networks," in *Proceedings of the 1st ACM international workshop on Quality of service security in wireless and mobile networks, ACM*, pp. 16–23, 2005.

[15] H. Han, X. L. Lu, and L. Y. Ren, "Using data mining to discover signatures in network-based intrusion detection," in *Proceedings of the 2002 International Conference on Machine Learning and Cybernetics*, vol. 1, pp. 13–17, Beijing, China, 2002.

[16] H. Y. Li, L. J. Lo, K. S. Chen, K. S. Wong, and K. P. Chang, "Robin sequence: review of treatment modalities for airway obstruction in 110 cases," *International Journal of Pediatric Otorhinolaryngology*, vol. 65, no. 1, pp. 45–51, 2002.

[17] C.-Y. Tseng, P. Balasubramanyam, C. Ko, R. Limprasittiporn, J. Rowe, and K. Levitt, "A specification-based intrusion detection system for AODV," in *Proceedings of the 1st ACM Workshop on Security of Ad Hoc and Sensor networks (in association with 10th ACM Conference on Computer and Communications Security)*, pp. 125–134, October 2003.

[18] J. P. Amaral, L. M. Oliveira, J. J. P. C. Rodrigues, G. Han, and L. Shu, "Policy and network-based intrusion detection system for IPv6-enabled wireless sensor networks," in *Proceedings of the*

2014 1st IEEE International Conference on Communications (ICC '14), pp. 1796–1801, June 2014.

[19] B. Sun, L. Osborne, Y. Xiao, and S. Guizani, "Intrusion detection techniques in mobile ad hoc and wireless sensor networks," *IEEE Wireless Communications Magazine*, vol. 14, no. 5, pp. 56–63, 2007.

[20] L. Mostarda and A. Navarra, "Distributed intrusion detection systems for enhancing security in mobile wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 4, no. 2, pp. 83–109, 2008.

[21] Y. Wang, X. Wang, B. Xie, D. Wang, and D. P. Agrawal, "Intrusion detection in homogeneous and heterogeneous wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 7, no. 6, pp. 698–710, 2008.

[22] G. Li, J. He, and Y. Fu, "Group-based intrusion detection system in wireless sensor networks," *Computer Communications*, vol. 31, no. 18, pp. 4324–4332, 2008.

[23] V. Bhuse, A. Gupta, and A. Al-Fuqaha, "Detection of masquerade attacks on wireless sensor networks," in *Proceedings of the 2007 IEEE International Conference on Communications (ICC'07)*, pp. 1142–1147, June 2007.

[24] M. V. De Sousa Lemos, L. Barroso Leal, and R. Holanda Filho, "A new collaborative approach for intrusion detection system on wireless sensor networks," in *Novel Algorithms and Techniques*, Springer, 2010.

[25] S. Shin, T. Kwon, G.-Y. Jo, Y. Park, and H. Rhy, "An experimental study of hierarchical intrusion detection for wireless industrial sensor networks," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 4, pp. 744–757, 2010.

[26] T. M. Mubarak, S. A. Sattar, A. Rao, and M. Sajitha, "A collaborative, secure and energy efficient intrusion detection method for homogeneous WSN," *Communications in Computer and Information Science*, vol. 192, no. 3, pp. 102–110, 2011.

[27] A. Le, J. Loo, K. K. Chai, and M. Aiash, "A specification-based IDS for detecting attacks on RPL-based network topology," *Information (Switzerland)*, vol. 7, no. 2, article no. 25, 2016.

[28] R. Mitchell and I.-R. Chen, "Behavior rule specification-based intrusion detection for safety critical medical cyber physical systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 1, pp. 16–30, 2015.

[29] J. C. Bezdek, R. Ehrlich, and W. Full, "FCM: the fuzzy c-means clustering algorithm," *Computers & Geosciences*, vol. 10, no. 2-3, pp. 191–203, 1984.

[30] A. Laio and A. Rodriguez, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, 2014.

[31] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and J. Piatt, "Support vector method for novelty detection," in *Proceedings of the 13th Annual Neural Information Processing Systems Conference (NIPS '99)*, pp. 582–588, December 1999.

[32] J. Davis and S. Magrath, "A survey of cyber ranges and testbeds," Defence Science and Technology Organisation Edinburgh (Australia) Cyber and Electronic Warfare Div, 2013.

[33] S. Suthaharan, M. Alzahrani, S. Rajasegarar, C. Leckie, and M. Palaniswami, "Labelled data collection for anomaly detection in wireless sensor networks," in *Proceedings of the 6th International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP '10)*, pp. 269–274, Brisbane, Australia, December 2010.

[34] M. A. Livani and M. Abadi, "A PCA-based distributed approach for intrusion detection in wireless sensor networks," in *Proceedings of the International Symposium on Computer Networks and Distributed Systems (CNDS '11)*, pp. 55–60, February 2011.

[35] S. Shamshirband, A. Patel, N. B. Anuar, M. L. M. Kiah, and A. Abraham, "Cooperative game theoretic approach using fuzzy Q-learning for detecting and preventing intrusions in wireless sensor networks," *Engineering Applications of Artificial Intelligence*, vol. 32, pp. 228–241, 2014.

[36] J.-Y. Huang, I.-E. Liao, Y.-F. Chung, and K.-T. Chen, "Shielding wireless sensor network using Markovian intrusion detection system with attack pattern mining," *Information Sciences*, vol. 231, pp. 32–44, 2013.