An Adaptive Intrusion Detection Method for Wireless Sensor Networks

Hongchun Qu
Chongqing Industrial Networking Collaborative Innovation Center, College of Automation, Chongqing University of Posts and Telecommunications
Key Laboratory of Industrial IOT and Networked Control (Chongqing University of Posts and Telecommunications), Ministry of Education, Chongqing, China

Zeliang Qiu, Xiaoming Tang, Min Xiang, Ping Wang
College of Automation,
Chongqing University of Posts and Telecommunications
Chongqing, China

Abstract—Current intrusion detection systems for Wireless Sensor Networks (WSNs) which are usually designed to detect a specific form of intrusion or only applied for one specific type of network structure has apparently restrictions in facing various attacks and different network structures. To bridge this gap, based on the mechanism that attacks are much likely to be deviated from normal features and from different shapes of aggregations in feature space, we proposed a knowledge based intrusion detection strategy (KBIDS) to detect multiple forms of attacks over different network structure. We firstly, in the training stage, used a modified unsupervised mean shift clustering algorithm to discover clusters in network features. Then the discovered clusters were classified as an anomaly if they had a certain amount of deviation from the normal cluster captured at the initial stage where no attacks could occur at all. The training data combined with a weighted support vector machine were then used to build the decision function that was used to flag network behaviors. The decision function was updated periodically after training by merging newly added network features to adapt network variability as well as to achieve time efficiency. During network running, each node uniformly captured their status as feature vector at certain interval and forwarded them to the base station on which the model was deployed and run. Using this way, our model can work independently of network structure in both detection and deployment. The efficiency and adaptability of the proposed method have been tested and evaluated by simulation experiments deployed on QualNet. The simulations were conducted as a full-factorial experiment in which all combinations of three forms of attacks and two types of WSN structures were tested. Results demonstrated that the detection accuracy and network structure adaptability of the proposed method outperforms the state-of-the-art intrusion detection methods for WSN.

Keywords—Wireless sensor network; intrusion detection system; knowledge based detection; clustering algorithm; weighted support vector machine

I. INTRODUCTION

Wireless Sensor Network (WSN) is usually composed of many randomly distributed tiny wireless sensor nodes that collect and send sensory data in a coordinated way. In contrast to traditional wireless networks, inherent advantages such as lower cost and more convenient deployment have largely extended WSN application fields, e.g., health care monitoring [1], smart home [2] and military surveillance and reconnaissance [3]. The security of WSN for those crucial fields has been an important demand [4]. Since wireless sensor nodes usually are limited by power supply, computation capability and communication range [5], traditional encryption/decryption techniques that require an uninterrupted power supply to retain frequently key management and access control are unrealistic to be applied to WSN [6]. Thus, establishing an intrusion detection system (IDS) to meet the security requirements in WSN is essential.

In contrast with wired and ad hoc wireless networks, WSNs are susceptible to various forms of security threats due to their open and unreliable communication channel, dynamic topology structure as well as lacking central coordination [7]. In general, intrusion can be made by singular or multiple attacks. The singular attack, such as flooding attack, black hole attack, rushing attack and so on, occurs independently in WSNs during a certain interval. In flooding attack, a malicious node usually attempts to overwhelm processing capacity and energy of the sensor node as well as network bandwidth by constantly sending a stream of insignificant packets at a very short interval [8]. In black hole attack, an intruder tampers with packets by advertising itself as the possible shortest path to the destination node, which results in the fact that most of the packets are forwarded to the intruder [9]. In rushing attack, the attacker forwards RREQ (route request packets) packets immediately without processing after received them from other nodes, which results in high jitter of the entire network [10]. For the multiple attack situations in which various types of attacks occur simultaneously, the intrusion features might be blurred by the intertwined attacks. For example, if flooding attack and black hole attack occurring simultaneously, when the relay nodes received the flooding packets, they may forward these packets to the black hole attacker instead of keeping flooding them, which makes the attacks appearing reasonable and thus covers the intrusion. Since multiple-attacks is much likely to happen than single ones, it would be benefit to have an intrusion detection solution that is capable of handling multiple attacks [11].

In recent years, many intelligent intrusion detection systems that can only deal with singular form of malicious attacks have been developed for WSN. Athmani et al. [12] protected hierarchical WSN from black hole attack by
controlling packets transfer between sensor nodes and the base station. Although this lightweight scheme demonstrated significant improvement in energy saving, it was unable to defend flooding attack that aims to increase packets transfer between sensor nodes [13]. In order to minimize energy consumption in intrusion detection activities, Di Sarno and Garofalo [14] proposed a method in which status of node energy was only necessary to detect multi-layer flooding attacks. However, this cross-layer intrusion detection approach has not shown the ability to detect attacks in which energy consumption is irrelevant. For example, in a selective forwarding attack, a malicious node either forwards packets of a certain node or not does not significantly affect its energy consumption. Lim and Huie [15] introduced a Hop-by-Hop Cooperative Detection (HCD) method to decrease the probability of misbehavior forwarding while achieve more than 95% package delivery. However, the paper did not mention how to detect attacks that are not misbehavior forwarding relevant, such as flooding attack. Sarigiannidis et al. [16] presented an expert system, i.e., the RADS (rule-based anomaly detection system), based on an ultra-wideband (UWB) ranging-based detection algorithm. It seemed promising in detecting sybil attack in large-scale WSN with high detection rate and low false alarm rate, while no cooperation and data sharing between nodes are needed. However, no evidence has been shown that the RADS is able to detect unknown attacks. Obado et al. [17] calculated the number of hops on the shortest paths between a source node and a destination node as input to a Hidden Markov Model (HMM) Viterbi algorithm to identify wormhole attack. Although the HMM Viterbi algorithm reduced power consumption of the sensor nodes, it was unable to recognize other attacks that are path independent, such as flooding attack and rushing attack. Although these intrusion detection systems demonstrated merits in terms of detection capability or minimization in resource consumption, the bottleneck is that they just can detect singular threat. Researchers have been seeking available information that can be helpful to detect multiple attacks. According to Butun et al. [7], if a profile representing stochastic network behavior is generated in feature space based on the captured network traffic, malicious behaviors against a WSN could lead profile in the feature space to be deviated from the normal range and form different aggregations. Thus, different forms of attacks are much likely to have different shapes of aggregations in feature space.

In general, WSN has two types of network structures (topologies), i.e., hierarchical (cluster) network and flat network [18]. In the hierarchical network, nodes are organized into clusters according to their range of transmission. Each cluster has a cluster head that is responsible to transmit information to the base station. In the flat network, all nodes are identical in routing functions, i.e., transmitting packets in a multi-hop way [7]. Current intrusion detection systems usually take advantage of information of network structures to detect attacks [18]. Shamshirband et al. [19] proposed a cooperative multi-agent based fuzzy artificial immune system to detect DDoS (distributed denial-of-service attack), where the sink node and base station work together to choose the best strategy for discovering an impending attack. However, the authors did not detail the cooperative manner between the common nodes and the base station in the flat network as well as the implementation. Based on the mechanism that the residual energy of nodes around the sinkhole is much less than other nodes when a flat network is suffering sinkhole attack, Shafiei et al. [20] built a geostatistical hazard model and a distributed monitoring method to detect and defend sinkhole attacks. However, this strategy does not apply to hierarchical case, because it is very difficult to identify sinkhole attack launched in a cluster head when there is no significant difference in residual energy of nodes around the cluster head between normal and attacked situations. Therefore, the information of network structure may be helpful to form patterns in intrusion detection on one hand, it may also restrict the application scope of IDS [21] on the other hand. That means that how to efficiently use network structures while not be constrained by them, i.e., to make the IDS to be network structure independent, is tricky.

The aim of this research was to develop a network structure independent intrusion detection model for WSN. The proposed model employed a knowledge-based detection strategy in which the mechanism is based on the fact that different forms of attacks are much likely to have different shapes of aggregations in feature space. Specifically, we captured network traffics and projected them into feature space as profiles representing stochastic network behaviors, and then the shapes of aggregations of the profiles could be regarded as an indicator to flag network behavior as normal or abnormal. To achieve this goal, we firstly, in the training stage, used a modified unsupervised mean shift clustering algorithm to discover clusters from the profile in the feature space. Then the discovered clusters can be classified as an anomaly if they have a certain amount of deviation from the normal cluster (behavior) captured at the initial stage where no attacks could occur at all. The training data combined with a weighted support vector machine were then used to build the decision function that was used to flag network behaviors. The decision function was updated periodically after training by merging newly added network traffic to mitigate the impact of outliers and noise as well as improve detection accuracy. During network running, each node uniformly captured network traffics as profiles at certain interval and forwarded them to the base station on which the model was deployed and run. Using this way, our model can work independently of network structure in both detection and deployment.

The rest of this paper is organized as follows. Section 2 briefly describes related work. The proposed model is presented in Section 3. Simulation intended to evaluate the performance of the model is presented in Section 4. Section 5 summarizes this paper with indications of future work.

II. RELATED WORKS

A typical anomaly detection technique usually identifies behavior that has a certain amount of deviation from normal behavior as an anomaly. Garofalo et al. [22] utilized decision tree classification and lightweight detection techniques to achieve trade-off between high detection rate and energy saving. However, the paper did not give detail how to deal with unknown attacks not described in the reference dataset. A lightweight IDS was developed by using a wrapper based feature selection algorithm to remove redundant features and
employing a neural network based decision tree to optimize feature selection [23]. Although this detection paradigm increased the generalization ability by incorporating neural networks, its ability to identify unseen pattern was incomplete due to lacking updated decision function. A bio-inspired approach, i.e., the Watchdog based Clonal Selection Algorithm (WCSA), was implemented by Nishanthi and Virudhunagar [24]. It was successful in detecting known attacks but failed to detect unknown ones [4]. While these intrusion detection methods were featured as energy saving and high detection accuracy, they failed to detect “unknown” attacks. To address this problem, we used an unsupervised data mining method to classify an anomaly from normal behavior without any prior knowledge. In addition, the decision function was updated periodically to adapt to changes in network features over time to increase the generalization ability.

Improving detection accuracy can be achieved from two directions, i.e., increasing detection rate and decreasing false alarm rate. Salmon et al. [25] utilized a tailored Dendritic Cell Algorithm (DCA) derived from Danger Theory immune-inspired techniques in which different input signals can be categorized by DCA, i.e., the signals that caused damage were regarded as anomalous while others were classified as normal signals. Experimental data showed that DCA has high detection rate but no false alarm rate. An agent-based artificial immune system was developed by [26]. In their method, two types of agents, e.g., the dendritic cells agents and the T-cell agents, collaborated with each other to count danger value being regarded as indicators to detect malicious attacks. This scheme achieved low false alarm rate but still cannot obtain enough detection rate [19]. Accordingly, we used a weighted support vector machine to maximize the margin between clusters of normal and anomaly to minimize the classification error, which in turn effectively enhanced detection accuracy.

III. THE MODEL

In this model, network traffics are discretized by time slice defined as $\Delta t$ (Fig. 1). Each node captures and sends its status as a $d$-dimensional feature vector $x_i = (x_{i1}, x_{i2}, ..., x_{id})$ to the base station at interval of $\Delta t$, where $d$ is the number of feature types (see Table 3 for detail).

![Feature vector diagram](image)

Fig. 1. Network traffic. Each node captures its status at regular time interval $\Delta t$ as a feature vector. Network traffic is divided into training data and testing data based on time boundary $m$. The feature vectors extracted in a short period of time $[0, n]$ after WSN initialization are regarded as normal data.

![Diagram of processing steps](image)

Fig. 2. The schematic diagram of processing steps in KBIDS.

**Definition 1:** Network traffic is defined as a matrix $X_{\text{All}} = \{x_1, x_2, ..., x_{|\text{All}|}\}$ that contains all feature vectors recorded in interval $[0, t]$, where $\text{all} = N * t/\Delta t$, and $N$ is the total number of nodes.

**Definition 2:** Normal data is defined as a matrix $X_{\text{Nor}} = \{x_1, x_2, ..., x_{nr}\}$, $nr = N * n/\Delta t$, which contains all feature vectors captured at the initial stage $[0, n]$ where no attacks could occur at all.

**Definition 3:** Training data is defined as a matrix $X_{\text{Train}} = \{x_1, x_2, ..., x_{tr}\}$, $tr = N * m/\Delta t$, which contains all feature vectors captured at the training stage $[0, m]$.

**Definition 4:** Testing data is defined as a matrix $X_{\text{Test}} = \{x_{tr+1}, x_{tr+2}, ..., x_{te}\}$, $te = N * (t - m)/\Delta t$, which contains all feature vectors captured at the testing stage $(m, t)$.

The proposed method is performed at the base station and includes the following four steps (Fig. 2):

1) **Preprocessing:** Training data are normalized by the min-max normalization method.

2) **Training:** The normalized training data are grouped into a certain number of clusters by a modified mean shift clustering algorithm. These clusters are eventually merged into two clusters according to the distance between them and the center of other clusters. Each feature vector in the training data is tagged as normal or anomaly by comparing with the normal data and the result of clustering. Further, each feature vector is assigned a weight representing the distance between it and its cluster center. The training data with labels of weights are served as inputs to a weighted support vector machine to establish a decision function.

3) **Detecting:** The testing data are flagged as normal or anomaly by the decision function.

4) **Updating:** In the testing stage, the feature vectors that have been processed are merged into the training data to rebuild the decision function at specific an interval of $\Delta T = k\Delta t$, $k \in \mathbb{N}_+$.
The intrusion detection algorithm is deployed on the base station of a WSN and other nodes are only responsible for capturing and transmitting their own network status.

A. Preprocessing

In order to mitigate the effects of extreme value at one or several dimensions on final results as well as speed convergence of the algorithm [23], training data are normalized by the min-max normalization method. Giving the training data $X_{Train}$:

$$ X_{Train} = \begin{bmatrix} x_1^1 & x_1^2 & \cdots & x_1^d \\ x_2^1 & x_2^2 & \cdots & x_2^d \\ \vdots & \vdots & \ddots & \vdots \\ x_{tr}^1 & x_{tr}^2 & \cdots & x_{tr}^d \end{bmatrix} $$

A set of the minimum and the maximum values for each column of $X_{Train}$ are respectively obtained as $x_{tmin} = \{x_{1min}, x_{2min}, \ldots, x_{dmin}\}$ and $x_{tmax} = \{x_{1max}, x_{2max}, \ldots, x_{dmax}\}$. Each feature vector $x_t \in X_{All}$ is then normalized by (1):

$$ x_{tp} = \frac{x_t - x_{tmin}}{x_{tmax} - x_{tmin}} \tag{1} $$

Where, $x_{tp}$ is a normalized feature vector.

B. Training

The feature vectors (points) should be aggregated into a certain region in the feature space in the normal situation. But they would be deviated from the normal region while being attacked. From the perspective of feature space, different forms of attacks may result in significant difference in degree of deviation from the normal region, and thus generate several aggregations. The unsupervised mean shift clustering algorithm (MSCA) [27] can effectively discover different concentrated regions in the feature space to form arbitrary clusters, which represents the aggregation of the features resulted from attacks. In this step, the MSCA is employed to cluster training data, flag them as normal or anomaly and feeds them into the classifier for training.

Given $nr$ data points $x_t \in X_{got}$, $t = 1, 2, \ldots, nr$ on a $d$-dimensional space $\mathbb{R}^d$, the initial feature vector $x_t$ is continuously shifted by adding a shifting vector $m_h$, where $m_h$ can be calculated by (2) (please note that the shifting of $x_t$ results in changes in $m_h$). $x_t$ stops shifting when $m_h$ falls below a certain threshold.

$$ m_h = F(x_t) = \frac{\sum_{k=1}^{nr} x_{t_k} k'(\frac{x_t - x_{t_k}}{\|x_t - x_{t_k}\|^2})}{\sum_{k=1}^{nr} k'(\frac{x_t - x_{t_k}}{\|x_t - x_{t_k}\|^2})} - x_t \tag{2} $$

Where, $g(x) = -k'(x)$, $k'(x)$ and $h$ are respectively the derivative and the bandwidth of the kernel profile $k(x)$ which is defined by a multivariate normal kernel function $K(x)$:

$$ K(x) = (2\pi)^{-\frac{d}{2}}\exp\left(-\frac{1}{2}\|x\|^2\right) = c_{k,d}|x|^{\frac{d}{2}} \tag{3} $$

Where, $c_{k,d}$ is a normalization constant assuring $K(x)$ integrates to 1. During the process of the $x_t$ shifting, the points that it has traveled are regarded belong to the same cluster and the last one is regarded as the cluster center $c_t$. In the standard MSCA, all data points do the same work as $x_t$ did. However, in order to speed the convergence and promote precision of the standard MSCA, a modified version that recording the track of shifting of feature vectors is given as follows:

Each point $p_k$ on the track of $x_t$ and its shifting distances $d_k$ are recorded to form a similar set $s_t$:

$$ s_t = \{(p_1, d_1), (p_2, d_2), \ldots, (p_k, d_k)\}, k = 1, 2, \ldots, \tag{4} $$

For all feature vectors we have the similar sets:

$$ S = \{s_1, s_2, \ldots, s_t\}, i = 1, 2, \ldots, \tag{5} $$

The cluster centers are defined as:

$$ C = \{c_1, c_2, \ldots, c_t\}, j = 1, 2, \ldots, \tag{6} $$

Based on the track of $x_t$, the subsequent each feature vector $x_t \in x_{tr}, t = 2, 3, \ldots, tr$ is clustered by the following two steps:

Step 1: The Euclidean distances $d_{xtpk}$ between $x_t$ and each point $p_k$ in each similar set $s_t \subseteq S$ is calculated by (7). If $d_{xtpk}$ is less than the shifting distance $d_k$, the tuple $(x_t, d_{xtpk})$ is added to $s_t$; Otherwise, proceed to step 2.

$$ d_{xtpk} = \sqrt{\sum_{t=1}^{d} (x_t^t - p_k^t)^2} \tag{7} $$

Step 2: $x_t$ does the same work as $x_t$ did to generate a new similar set $s_t$ and a cluster center $c_t$. If $C$ has a cluster center $c_j$ which is equal to $c_t$, then merge $s_t$ into $s_j$: $s_j = s_t \cup s_j$; Otherwise, $c_t$ and $s_t$ are inserted to $C$ and $S$, respectively.

After the two steps are completed, the training data are grouped into several clusters. The cluster including the normal data is regarded as the normal cluster. The cluster whose cluster center is farthest from the normal cluster is classified as the abnormal cluster. The rest of clusters are then merged into either the normal or abnormal cluster based on the relative distance to them.

When a feature vector is received after time $n$, it is immediately classified into the nearest cluster thus can be flagged as normal or anomaly without extra training. In addition, to mitigate the effect of outliers or noise in clustering process on final decision and improve the detection accuracy, a weighted support vector machine (WSVM) was introduced to build a decision function (i.e. the optimal margin hyperplane classification) from the clustering results.

Given $tr$ data points:

$$ \{x_t, y_t\}_{t=1}^{tr}, x_t \in X_{Train}, y_t = \{-1, 1\} $$

Where $y_t$ denotes class labels (i.e. normal or anomaly), the classification is defined as:

$$ f(x) = \text{sign}(<w, x> + b) \tag{8} $$

Where $w$ is a weight vector and $b$ is the bias. In order to optimize $w$ and $b$, the WSVM requires the solution of the following optimization problem:
\[
\begin{align*}
\min \frac{1}{2}||w||^2 + \frac{1}{\nu_{tr}} \sum_{t=1}^{tr} r_t W_t, \quad 0 < \nu < 1 \\
s.t. (y_t < w . \phi(x_t) > +b) \geq 1 - \xi_t, \quad \xi_t \geq 0, \quad t = 1, 2, ..., \text{tr}
\end{align*}
\]  
(9)

Where, \( \nu \) is the penalty factor of misclassification. \( \xi_t \) is the slack parameter to control noise. Vectors \( x_t \) is mapped into a higher dimensional space by the function \( \phi. W_t \) is a weight, which represents the relative contribution of \( x_t \) to the decision function. \( W_t \) assigned to data point \( x_t \) is calculated by (10): 
\[
W_t = \exp(- \sum_{i=1}^{d} (x_t^i - c)^2)
\]  
(10)

where \( c \) is the cluster center of \( x_t \). Unlike the standard SVM, where all training data points in one class are equally important, WSVM reduces the effect of outliers and noise by setting different weights [28]. According to Lagrangian duality theory, the WSVM optimization problem in (9) is converted to a quadratic programming problem:
\[
\begin{align*}
\min \sum_{t=1}^{tr} a_t - \frac{1}{2} \sum_{t=1}^{tr} \sum_{k=1}^{tr} a_t a_k y_t y_k \phi(x_t) \phi(x_k) > \\
s.t. \sum_{t=1}^{tr} a_t y_t = 0, \quad 0 \leq a_t \leq \frac{1}{\nu_{tr}} W_t, \quad t = 1, 2, ..., \text{tr}
\end{align*}
\]  
(11)

where \( a_t \) is the Lagrangian parameter. The Karush–Kuhn–Tucker conditions of the SVM are defined as:
\[
\begin{align*}
& a_t y_t (< w . \phi(x_t) > + b) - 1 + \xi_t = 0, \quad t = 1, 2, ..., \text{tr} \\
& \left(\frac{1}{\nu_{tr}} W_t - a_t \right) \xi_t = 0, \quad t = 1, 2, ..., \text{tr}
\end{align*}
\]  
(12)

Finally, the optimal value of \( w \) and \( b \) are gained by:
\[
\begin{align*}
& w = \sum_{t=1}^{tr} a_t y_t \phi(x_t) \\
& b = y_k - \sum_{t=1}^{tr} y_t a_t \phi(x_k), \quad \forall k \in [1, \text{tr}]
\end{align*}
\]  
(13)

Hence, the decision function is obtained by:
\[
\begin{align*}
& f(x) = sgn(\sum_{t=1}^{tr} a_t y_t < \phi(x_t), \phi(x) > - \sum_{t=1}^{tr} y_t a_t < \phi(x_t), \phi(x_k) > + b_k) \\
& = \begin{cases} 
1 : \text{Normal} \\
-1 : \text{Anomalous}
\end{cases}
\end{align*}
\]  
(14)

### C. Detecting

In this step, each feature vector in testing data is flagged as normal or anomaly by the decision function (14). The feature vector is then merged into either normal or anomaly cluster according to the decision that has been made by the decision function. After that, it is used to update the decision function afterwards.

### D. Updating

In order to cope with the possible changes in network features over time, the decision function needs to be updated at an interval of time \( \Delta T \). In this step, the cluster centers are reevaluated by MSCA with updated training data, and the weight of each feature vector is adjusted as well. After that, the decision function is updated accordingly by WSVM.

The pseudo code of KBIDS is shown in Algorithm 1.

### IV. Simulation Experiments

In order to evaluate the performance of KBIDS, eight experiment scenario (Table 1) were simulated by QualNet on a PC with Inter(R) Core(TM) i7-4470k, 3.50GHz, 8GB memory (RAM). In a flat network, we randomly deployed 30 sensor nodes in a region with dimension of 1000(m) x 1000(m) and deployed the base station in the center of the region, as shown in Fig. 3(a). Ten percent of nodes were designated as malicious nodes that performed attack. Compared with the flat network, the hierarchical protocol is more suitable for large-scale networks.
networks in reducing node energy consumption and communication bandwidth [29]. Hence, the number of nodes in the hierarchical network was 100, and the number of attackers was 10. The relationship between nodes in the hierarchical network is given in Fig. 3(b).

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Attack type</th>
<th>Network structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>Black hole attack</td>
<td>Flat network</td>
</tr>
<tr>
<td>Case 2</td>
<td>Flooding attack</td>
<td>Flat network</td>
</tr>
<tr>
<td>Case 3</td>
<td>Rushing attack</td>
<td>Flat network</td>
</tr>
<tr>
<td>Case 4</td>
<td>Multiple attacks</td>
<td>Flat network</td>
</tr>
<tr>
<td>Case 5</td>
<td>Black hole attack</td>
<td>Hierarchical network</td>
</tr>
<tr>
<td>Case 6</td>
<td>Flooding attack</td>
<td>Hierarchical network</td>
</tr>
<tr>
<td>Case 7</td>
<td>Rushing attack</td>
<td>Hierarchical network</td>
</tr>
<tr>
<td>Case 8</td>
<td>Multiple attacks</td>
<td>Hierarchical network</td>
</tr>
</tbody>
</table>

For all types of network structure, the MAC layer and routing protocol of all devices were IEEE802.11 and Ad hoc On-demand Distance Vector Routing (AODV), respectively. Simulation time for each experiment scenario was set as 10000 seconds. The value of time $n$ was 100 seconds and $m$ was 5000 seconds. Network traffic flow was simulated by constant bit rate (CBR) with packets of 512 bytes. The mobility model of nodes was simulated random waypoint (RWP) model with pause time of 5 seconds and the maximum speed of 10m/s. The statistical data of the energy consumption was counted by MicaZ radio energy model [30]. The key parameters of the simulation experiments were presented in Table 2.

In each singular attack scenario, only one type of the three attacks, e.g., the black hole attack, flooding attack or rushing attack, was launched during 4000 and 7000 seconds. Unlike singular attack scenarios, the three attacks were simultaneously launched during 4000 and 7000 seconds in the multiple attacks scenarios. Each of these scenarios was replicated five times by setting different initial position of sensor nodes.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Flat network</th>
<th>Hierarchical network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation time</td>
<td>10000(s)</td>
<td>10000(s)</td>
</tr>
<tr>
<td>Field size</td>
<td>1000(m) x 1000(m)</td>
<td>1000(m) x 1000(m)</td>
</tr>
<tr>
<td>Total number of nodes</td>
<td>30</td>
<td>100</td>
</tr>
<tr>
<td>Number of attackers</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>Traffic type</td>
<td>CBR</td>
<td>CBR</td>
</tr>
<tr>
<td>Traffic size</td>
<td>512(B)</td>
<td>512(B)</td>
</tr>
<tr>
<td>Routing protocol</td>
<td>AODV</td>
<td>AODV</td>
</tr>
<tr>
<td>MAC protocol</td>
<td>IEEE802.11</td>
<td>IEEE802.11</td>
</tr>
<tr>
<td>Mobility model</td>
<td>RWP</td>
<td>RWP</td>
</tr>
<tr>
<td>Pause time</td>
<td>5(s)</td>
<td>5(s)</td>
</tr>
<tr>
<td>Max moving speed</td>
<td>10(m/s)</td>
<td>10(m/s)</td>
</tr>
</tbody>
</table>

For all types of network structure, the MAC layer and routing protocol of all devices were IEEE802.11 and Ad hoc On-demand Distance Vector Routing (AODV), respectively. Simulation time for each experiment scenario was set as 10000 seconds. The value of time $n$ was 100 seconds and $m$ was 5000 seconds. Network traffic flow was simulated by constant bit rate (CBR) with packets of 512 bytes. The mobility model of nodes was simulated random waypoint (RWP) model with pause time of 5 seconds and the maximum speed of 10m/s. The statistical data of the energy consumption was counted by MicaZ radio energy model [30]. The key parameters of the simulation experiments were presented in Table 2.

In each singular attack scenario, only one type of the three attacks, e.g., the black hole attack, flooding attack or rushing attack, was launched during 4000 and 7000 seconds. Unlike singular attack scenarios, the three attacks were simultaneously launched during 4000 and 7000 seconds in the multiple attacks scenarios. Each of these scenarios was replicated five times by setting different initial position of sensor nodes.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>numRequestRecvd</td>
<td>Number of route requests received</td>
</tr>
<tr>
<td>numRequestResent</td>
<td>Number of route requests resent because node did not receive a route reply.</td>
</tr>
<tr>
<td>numRequestRecvdAsDest</td>
<td>Number of route requests received by the destination.</td>
</tr>
<tr>
<td>numRequestInitiated</td>
<td>Number of route request messages initiated</td>
</tr>
<tr>
<td>numRequestRelayed</td>
<td>Number of route request messages forwarded by intermediate nodes.</td>
</tr>
<tr>
<td>numReplyInitiatedAsDest</td>
<td>Number of route replies initiated from the destination.</td>
</tr>
<tr>
<td>numReplyInitiatedAsIntermediate</td>
<td>Number of route replies initiated as an intermediate hop.</td>
</tr>
<tr>
<td>numReplyRecvd</td>
<td>Number of route replies received by the node.</td>
</tr>
<tr>
<td>numReplyRecvdAsSource</td>
<td>Number of route replies received as data source</td>
</tr>
<tr>
<td>numReplyForwarded</td>
<td>Number of route replies forwarded by intermediate hops</td>
</tr>
<tr>
<td>numDataInitiated</td>
<td>Number of data packets sent as the source of the data.</td>
</tr>
<tr>
<td>numDataRecieved</td>
<td>Number of data packets received as the destination of the data</td>
</tr>
<tr>
<td>numHops</td>
<td>Aggregate sum of the hop counts of all routes added to the route cache.</td>
</tr>
</tbody>
</table>
A feature vector (Table 3) which is the basic unit of information processing in KBIDS was constructed by capturing 13 types of features representing node status [31].

In order to test the efficiency of KBIDS, detection rate and false alarm rate were used as index of assessment [32]. Detection rate was calculated as the percentage of the numbers of successfully detected anomalies over the total numbers of anomalies. False alarm rate was calculated as the numbers of false alarm over the total numbers of normal data [18].

V. RESULTS

Simulation results of KBIDS were compared with mainstream intrusion detection methods, such as PCA-based centralized approach (PCACID) [33] K-Means, Mean Shift, Decision Trees (DT) and Logistic Regression (LR), to evaluate efficiency of detection and adaptively of network structure (Fig. 4). Results showed that in the eight experiment scenarios, the average detection rate and the false alarm rate of KBIDS were 97.854% and 1.875% with small standard deviation 0.922% and 1.069%, respectively, which demonstrated an obvious advantage than other mainstream methods. Although K-Means, Decision Trees and Logistic Regression obtained more than 92% average detection rate and less than 3% average false alarm rate at the same situation, their results had a large fluctuation, i.e., 4.831% and 4.291%, 1.327% and 1.348% as well as 3.922% and 2.547%. This exhibited their weak capability in dealing with various forms of attacks. In some cases, PCACID and Mean Shift achieved lower false alarm rate than KBIDS. However, they failed to detect anomaly constantly in all scenarios. Overall, KBIDS showed advantages over other mainstream detection algorithms in term of detection rate and false alarm rate. In addition, KBIDS achieved stable performance in all scenarios, particularly in scenarios with different network structure. This was a strong evidence showing that KBIDS is network independent.
As one of the key parameters, the length of time slice $\Delta t$ might be one important factor affecting results and network performance. Fig. 5 revealed the relationship between the variations in time slice $\Delta t$ and detection accuracy as well as energy consumption when the interval of updating $\Delta T$ was 300s. With the increasing of $\Delta t$, the average detection rate and the energy consumption of data transmission decreased gradually while the average false alarm rate rose steadily for these cases. Overall, the time slice $\Delta t=10s$ achieved a well trade-off between detection accuracy and energy consumption.

Since KBIDS employed a constant updating strategy to adapt network variability over time, the impact of update interval $\Delta T$ on the average running time (ART, i.e., the energy cost) of the updating step and the average detection accuracy cannot be ignored. When $\Delta t=10s$, although lower $\Delta T$ (20s) achieved high detection rate and low false alarm rate (Fig. 6), its cost, i.e., $\text{ART}=62.16s$ was much larger than $\text{ART}=1.34s$ where $\Delta T=100s$ where $\text{ART}=1.34s$. No surprise, the average detection rate and the average false alarm rate respectively decreased and increased with the increasing of $\Delta T$. However, the trend became stable after $\Delta T=500s$. Overall, $\Delta T=300s$ achieved a great balance between detection accuracy and energy cost, where $\text{ART}$ was about 3.83s which was much less than time slice $\Delta t=10s$.

In this algorithm, the computational complexity of the clustering step is $O(Tn^2)$, where $T$ is the average number of shifting, and $n$ is the number of feature vectors. The computational complexity of building WSVM is $O(dn^2)$ where $d$ is the dimension of feature vector. The computational complexity of the updating step is $O(M(T + d)n^2)$ where $M$ is the number of updates. Overall, the computational complexity of the algorithm can be approximated as $O(n^3)$, which is not significantly affected by the number of dimensions of feature vector. We acknowledge that with the increasing number of feature vectors, it is inevitable to increase the time cost for the updating step, which is the main source of energy consumption that delays the decision process. We used two strategies to solve this problem. First, decision can be made immediately after learning stage when WSVM has been established, no full training process is necessary at each decision making; Second, old feature vectors are removed from training data when new feature vectors come in, so that the size of the training data can be kept at a constant level.

VI. CONCLUSION

In this research, we developed KBIDS, a network structure independent intrusion detection model for WSN. KBIDS employed knowledge based detection strategy based on the
mechanism that attacks are much likely to be deviated from normal features and from different shapes of aggregations in feature space. In KBIDS, an unsupervised data mining method was used to classify an anomaly from normal behavior without any prior knowledge. In addition, the decision function was updated periodically to adapt to changes in network features over time to increase the generalization ability. Further, a weighted support vector machine was used to maximize the margin between clusters of normal and anomaly to minimize the classification error, which in turn effectively enhanced detection accuracy. During network running, each node uniformly captured their status as feature vector at certain interval and forwarded them to its neighbor. The based station runs the model to detect attack. Using this way, our model can work independently of network structure in both detection and deployment. Simulation experiments conducted on QualNet platform demonstrated that our model outperformed other mainstream algorithms in terms of detection efficiency, stability across different network structures and computational complexity. Sensitivity analysis gave insights into how model performance can be affected by some key parameters, thus future improvement can be directed.

ACKNOWLEDGMENT
This is a publication of dedicated Intelligent Manufacturing 2017 – Industrial Internet of Things for IPv6 validation. Funding was received from the National Natural Science Foundation of China (61374093) and Chongqing Research Program of Basic Research and Frontier Technology (cstc2017jcyjAX0453, cstc2015jcyjA40007).

REFERENCES

