An intrusion detection mechanism for IPv6-based wireless sensor networks

Min Wei, Chunmeng Rong, Erxiong Liang and Yuan Zhuang

Abstract
With the advancement of IPv6 technology, many nodes in wireless sensor networks realize seamless connections with the Internet via IPv6 addresses. Security issues are a significant obstacle to the widespread adoption of IPv6 technology. Resource-constrained IPv6 nodes face dual attacks: local and Internet-based. Moreover, constructing an active cyber defense system for IPv6-based wireless sensor networks is difficult. In this article, we propose a $K$-nearest neighbor-based intrusion detection mechanism and design a secure network framework. This mechanism trains an intrusion detection algorithm using a feature data set to create a normal profile. The intrusion detection algorithm uses the normal profile to perform real-time detection of network traffic data to achieve rapid detections connecting many devices in a wireless sensor network. In addition, we develop a test platform to verify this mechanism. Experimental results show that this mechanism is appropriate for IPv6-based wireless sensor networks and achieves a low false-positive rate and good intrusion detection accuracy at an acceptable resource cost.

Keywords
IPv6, wireless sensor network, $K$-nearest neighbor, intrusion detection, security

Date received: 7 June 2021; accepted: 22 December 2021

Handling Editor: Yanjiao Chen

Introduction
In recent years, the scale of wireless sensor networks (WSNs) has grown rapidly. Internet of Things (IoT) systems have many types of networks connecting various system entities. Different networks need to be combined to provide the necessary network connectivity between the entities attached to each network. Entities must be interoperable to operate seamlessly in different networks. However, currently available heterogeneous network protocols in WSNs are typically applicationspecific. Network-level solutions are required among WSNs and between wired and wireless networks to provide seamless communications and interactions among different network types. IPv6 facilitates information exchange, peer-to-peer connectivity, and seamless communication between different IoT systems.

IPv6 technology can have a large number of address resources, automatic address configuration, and good mobility. The use of IPv6 technology in WSNs is inevitable, especially when nodes in WSNs are required to connect to the Internet using IPv6 technology seamlessly. Numerous standardization work has been completed in Internet Engineering Task Force (IETF) to enable the use of IPv6 technology in WSNs. For IPv6 communication on IEEE 802.15.4 devices, IETF proposed IPv6 over low-power wireless personal area networks (6LoWPANs). These documents focus on the standardization of the IPv6 head compression, neighbor discovery, time-slotted channel hopping (TSCH), and so on.

Creative Commons CC BY: This article is distributed under the terms of the Creative Commons Attribution 4.0 License (https://creativecommons.org/licenses/by/4.0/) which permits any use, reproduction and distribution of the work without further permission provided the original work is attributed as specified on the SAGE and Open Access pages (https://us.sagepub.com/en-us/nam/open-access-at-sage).
WSN features and frameworks have been significantly changed because of access to the Internet via IPv6 technology, which may lead to new network threats and attacks. Moreover, the WSN is in an unknown environment with limited resources and hidden attacks. Therefore, it is necessary to study the security issues of IPv6-based WSNs.

Intrusion detection is a mechanism that detects network attacks by analyzing activities in a network or system. Once an attack is detected, an intrusion detection system (IDS) records relevant information about the attack. However, current intrusion detection mechanisms against multiple attacks support multiple protocols but are still in development stages. Therefore, an intrusion detection mechanism, considering the overall security of IPv6-based WSNs, should be investigated further.

The contributions of this article are as follows:

1. A common intrusion detection framework for IPv6-based WSNs is developed. Based on this framework, a security framework consisting of an intrusion detection console as the core, a traffic generation module, a traffic capture module, a feature processing module, and an intrusion detection module is proposed, and the coordination mechanism and workflow of each module are designed.

2. Methods of collecting and processing security feature data are described for IPv6-based Internet and IPv6-based wireless networks. The IPv6-based WSNs features for intrusion detection are specified in this article. A set of lightweight intrusion detection algorithms based on K-nearest neighbors (KNNs) is implemented for IPv6-based WSNs, and the algorithms are stable and can be used effectively in the intrusion detection console.

3. A test platform is developed to verify the proposed mechanism. The laboratory 6LoWPAN node and gateway are used to build an IPv6-based WSN and verify the proposed mechanism’s feasibility on the IPv6-based wireless. Compared with other schemes, the proposed mechanism can effectively reduce the false positive rate (FPR) of intrusion detection and achieves good detection efficiency and ACC.

This article is organized as follows. Section “Related work” reviews several existing works related to security and IDSs for WSNs. Section “Intrusion detection framework” describes the intrusion detection framework. Section “Intrusion detection mechanism” proposes a lightweight intrusion detection algorithm for IPv6-based WSNs based on KNN. Section “Verification and result analysis” develops a test platform to verify the proposed mechanism, and research results are analyzed and discussed. Finally, in section “Verification and result analysis,” we conclude the study with a summary.

**Related work**

Research on the security of Internet protocol (IP)-based WSNs has attracted much attention. In terms of standardization, IPv6 has supported Internet protocol security (IPsec) for WSNs. In 2020, RFC 5570 proposed an optional method for encoding packet sensitivity labels on IPv6 packets. The encoding provided multilevel network security services for network layer traffic in IPv6 environments. RFC 8750 updated an encapsulated security payload to generate a nonce using values provided in the encapsulating security payload sequence number to avoid sending initialization vector. RFC 4301 updated security architecture for IP. In 2019, RFC 8598 proposed two configuration payload attribute types for Internet key exchange protocol version 2 (IKEv2), adding support for private domain name system (DNS) domains.

Meanwhile, IP-based wireless sensor networks are usually resource-constrained mainly because nodes are attacked locally and on the Internet. Therefore, lightweight security mechanisms are necessary in this regard. Cao et al. designed a lightweight security D2D (Device to Device) system using multiple sensors on mobile devices. In the research by Raz and Magnusson, a lightweight IKE was proposed, and IKEv2 was adapted.

An active defense system can monitor a network and respond to detected attacks in real-time. Therefore, it is necessary to develop an active defense system for IPv6-based WSNs to deal with security issues and detect attacks in real-time.

There are several research studies on IDSs for specific attacks. In 2013, Shahid first proposed the IDS SVELTE for routing attacks by changing routing information against IPv6 WSNs, with intrusion detection using the widely used and accepted the opnet 14.5 simulators to simulate in turn WSN generation of data sets for normal flow and attack flow. Althubaity et al. proposed a hybrid specification-based IDS to protect the RPL (IPv6 routing protocol for low-power and loss networks) topology in 6TiSCH networks from any manipulation on the rank value to establish rank attack or on the routing metric to perform rank attack based on the objective function. Amaran and Mohan proposed Optimal Multilayer Perceptron (OMLP) with Dragonfly Algorithm (DA) for intrusion detection in
WSN. The OMLP model has high accuracy and detection rate. Choudhary and Taruna\textsuperscript{16} proposed a technique which is based on the frequency analysis onsite to find intrusion into the network; the data from these dedicated sensors are stored in a fuzzy analytical engine for inference. Jiang et al.\textsuperscript{17} proposed SLGBM, an intrusion detection method for wireless sensor networks. A LightGBM algorithm is utilized to detect different network attacks. Sharma et al.\textsuperscript{18} proposed a supervised machine learning-based IDS for RPL-based cyber–physical systems, that is capable of detecting several attacks.

Similarly, there are some research studies on IDSs for specific protocols. Moustafa et al.\textsuperscript{19} proposed an integrated intrusion detection technology. Message queuing telemetry transport (MQTT) protocols are used in IoT systems, and the AdaBoost ensemble learning method was developed using decision tree (DT), naive Bayesian, and artificial neural networks (ANNs). Verma and Ranga\textsuperscript{20} used an ensemble learning-based network IDS framework to detect routing attacks on the IPv6-based routing protocols for low-power and lossy networks. Shen et al.\textsuperscript{21–23} proposed IDSs for malware, which can suppress malware diffusion in IoT networks. Zhou et al.\textsuperscript{24} proposed a malware detection model based on game theory in WSNs. Liu et al.\textsuperscript{25–28} proposed a series of methods for virtual resource security detection in sensor edge cloud.

Existing research studies focus on specific protocols or attacks, and they can achieve effective intrusion detection. However, an intrusion detection mechanism, considering the overall security of IPv6-based WSNs, merits further investigation. Furthermore, the intrusion detection mechanism should be designed considering all IPv6-based WSN frameworks.

**Intrusion detection framework**

Traditional WSNs are typically stand-alone, not connected to any other external networks. They are usually composed of low-power, lossy networks, and many resource-constrained nodes, forming a closed wireless mesh network. To use IPv6 technology in WSNs, IETF proposed a 6LoWPAN protocol stack based on IEEE 802.15.4. The protocol stack has six layers, where its bottom layer adopts the IEEE 802.15.4 standards of the PHY and MAC layers. For implementing a seamless connection between the MAC and network layers, an adaption layer is added between the MAC and network layers to handle header compression, fragmentation, reassembly, and mesh route forwarding.

6LoWPAN nodes and gateways form an IPv6-based WSN through the 6LoWPAN protocol stack. When one or more gateways of the IPv6-based WSN access the Internet, an extended IPv6-based WSN forms. The network is connected to the intrusion detection console to form an intrusion detection framework. Figure 1 shows the IPv6-based WSN intrusion detection framework, including the intrusion detection console, the IPv6-based Internet side, and the IPv6-based wireless side.

For the IPv6-based Internet side, a normal server and a malicious server can generate an original packet, and the traffic generated by the servers constitutes normal and abnormal activities. The traffic sent by the servers is forwarded to a PC or a portable device via a router.

For the IPv6-based wireless network side, each IPv6-based node forwards a packet to an IPv6-based border router through the IPv6-based route node, and finally, the IPv6-based border router uploads it to the gateway. Each node is configured with the CoAP/
MQTT protocol and is connected to the gateway via the CoAP/MQTT proxy. An intrusion detection device is a tool for constructing and collecting security feature data for the intrusion detection mechanism. It can sniff packets from its neighbors and construct security feature packets.

The intrusion detection console logically includes five functional modules: a traffic generation module, traffic capture module, feature processing module, intrusion detection module, and intrusion response module. Their specific functions are as follows:

1. Traffic generation module: the traffic generation module includes a server on the IPv6-based Internet side and a sensor node on the IPv6-based wireless network side. These devices are responsible for generating original packets for intrusion detection.
2. Traffic capture module: the traffic capture module includes packet capture tools in the intrusion detection console and intrusion detection devices. The IPv6-based Internet side captures the traffic of an ingress router, and the IPv6-based wireless network side captures the security feature packets forwarded by the gateway to the Internet.
3. Feature processing module: the feature processing module is a feature extraction tool in the intrusion detection console. After capturing the original traffic, it is stored in a local database in the intrusion detection console. Feature extraction tools and feature processing algorithms help realize feature statistics and selection.
4. Intrusion detection module: the intrusion detection module stores processed feature data in a CSV (comma-separated values) file in the intrusion detection console, using it as an input to the intrusion detection module. This module trains the intrusion detection mechanism to form the normal profile (NP) of the intrusion detection model. The NP is used to detect and classify the real-time flow data into normal flow or abnormal flow in real-time.
5. Intrusion response module: the intrusion response module prevents organizational attacks by managing the network, such as taking malicious nodes offline or restoring normal network behaviors.

Intrusion detection mechanism

In this section, an intrusion detection mechanism is proposed for an IPv6-based WSN based on KNN. Figure 2 shows its specific workflow.

Three steps are involved in the proposed intrusion detection mechanism:

1. Security feature data collection and processing: on the IPv6-based Internet side, original packets generated by the gateway are collected and stored in the database. On the IPv6-based wireless network side, the intrusion detection device constructs a security feature message and eventually forwards it to the gateway. The packet capturing tool captures the packet from the gateway and stores it in the database. The feature processing module will perform feature extraction on packets stored in the local database to generate traffic features and generate security feature data after performing statistics on the traffic features.
2. Data standardization and feature selection: the feature processing module standardizes security

Figure 2. Intrusion detection mechanism workflow.
feature data and uses feature selection algorithms to screen appropriate security features. Finally, the feature processing module creates a security feature data set for training the intrusion detection algorithms.

3. Algorithm training and intrusion detection: the intrusion detection module trains the algorithm, generating an intrusion detection model. In addition, the intrusion detection module needs to be regularly updated to adapt to network changes. The intrusion detection module also detects new security feature data. When the detected traffic flow is abnormal, the intrusion response module is responsible for processing the abnormal node.

Security feature data collection and processing

The network traffic on the IPv6-based Internet side uses a packet capturing tool to capture original packets of the entry router to form a packet capture (pcap) file. The pcap file needs to be processed to generate a record for each message sent and received. Implicit information related to normal and abnormal activities is recorded. Those records are further processed and transformed into security feature data for online analysis by the intrusion detection algorithm. IPv6-based Internet side security features are divided into HTTP-based features, traffic-based features, and transaction-based features. Table 1 shows the HTTP-based features, Table 2 shows the traffic-based features, and Table 3 shows the IPv6-based Internet side transaction-based features.

The IPv6-based wireless network side security features include RPL-based features, application layer-based features, 6top-based features, transaction-based

Table 1. HTTP-based features.

| Feature | Description |
|---------|-------------|
| sbytes | Source to destination transaction bytes |
| dbytes | Destination to source transaction bytes |
| sttl | Source to destination time-to-live |
| dttl | Time-to-live in destination |
| sloss | Retransmission or drop the source packet |
| dloss | Retransmission or drop the destination packet |
| service | http, ftp, smtp, ssh, dns, ftp-data, irc |
| sload | Source bits per second |
| dload | Destination bits per second |
| spkts | The number of packages from a source to a destination |
| dpkts | The number of packages from a source to a destination |
| swin | Source TCP window |
| dwin | Destination TCP window |
| stcpb | Source TCP sequence number |
| dtcpb | Destination TCP sequence number |
| smeansz | The average value of the packet size sent by a source |
| dmeansz | The average value of the packet size sent by a destination |
| Sjit | Source jitter |
| Djit | Destination jitter |
| Sintpkt | Source packet arrival time |
| Dintpkt | Destination packet arrival time |
| tcprtt | TCP connection establishment round trip time (RTT) |

RTT: round trip time.

Table 2. Internet side traffic-based features.

| Feature | Description |
|---------|-------------|
| Srcip | Source IP address |
| sport | Source port |
| dstip | Destination IP address |
| dport | Destination port |
| proto | Protocol |
| dur | Total time |
| state | Indicate status and related agreements |
| stime | Start time |
| etime | End time |

IP: Internet protocol.

Table 3. Internet side transaction-based features.

| Feature | Description |
|---------|-------------|
| ct_state_ttl | Assign a number to each state according to the specified value range of the source/destination TTL |
| flw_http_mthd | The number of streams of Get and Post methods in HTTP |
| is_ftp_login | If the FTP session is accessed through a user and password, it returns 1; otherwise, it returns 0 |
| ct_ftp_cmd | The number of commands in the FTP session |
| ct_srv_src | The number of connections containing the same service and source address in 100 connections |
| ct_srv_dst | The number of connections containing the same service and destination address in 100 connections |
| ct_dst_ltm | The number of connections with the same destination address among 100 connections |
| ct_src_ltm | The number of connections with the same source address among 100 connections |
| ct_src_dport_ltm | The number of connections with the same source address and destination port in 100 connections |
| ct_dst_sport_ltm | The number of connections with the same destination address and source port in 100 connections |
| ct_dst_src_ltm | In 100 connections, the number of connections with the same source and destination |
features, and TSCH-based features. Table 4 shows the RPL-based features, Table 5 shows the application layer-based features, Table 6 shows the TSCH-based features, and Table 7 shows the 6top-based features, and Table 8 shows the transaction-based features.

Application layer-based features include IP address and the port numbers of a source and destination and protocol. Transaction-based features are generated based on the interaction of flow identifiers created in a time window to maintain online detection of malicious activities. This includes traffic statistics, such as the number of connections in a fixed period. A flow identifier and session time are sequentially stored by the packet capturing tool after obtaining the header information of the original packet. According to the timestamp of the captured packets, the packets are grouped and processed in a fixed collection cycle to generate traffic features in the collection cycle.

**Table 4.** RPL-based features.

| Feature       | Description                                    |
|---------------|------------------------------------------------|
| RplTxDAO      | The number of transported DAO packets          |
| RplTxDIO      | The number of transported DIO packets          |
| RplRxDAO      | The number of received DAO packets             |
| RplRxDIO      | The number of received DIO packets             |
| rplChurnParentSet | Times to change parent set                   |
| rplChurnRank  | Times to change rank                          |
| DAO: Destination Advertisement Object; DIO: Destination Oriented Directed Acyclic Graph Information Object.

**Table 5.** Application layer-based traffic features.

| Feature          | Description                                      |
|------------------|-------------------------------------------------|
| appGenerated     | The number of generated packets                 |
| appReachesDagroot| The number of packets reaching the root          |
| AppRelayed       | The number of relayed packets                   |

**Table 6.** TSCH-based features.

| Feature            | Description                               |
|--------------------|-------------------------------------------|
| numSharedCells     | The number of shared cells                |
| NumTxCells         | The number of transported cells            |
| NumRxCells         | The number of received cells               |
| numDedicatedCells  | The number of dedicated cells              |
| TschRxEB           | The number of received beacon              |
| TschTxEB           | The number of transported beacon           |

**Table 7.** 6top-based features.

| Feature              | Description                                      |
|----------------------|-------------------------------------------------|
| 6topRxAddResp        | Times for received response to add a time cell   |
| 6topRxDelResp        | Times for received response to delete a time cell|
| 6topRxDelReq         | Times for received request to delete a time cell |
| 6topTxDelResp        | Times for transported response to delete a time cell|
| 6topTxAddReq         | Times for transported request to add a time cell |
| 6topRxAddReq         | Times for received request to add a time cell    |
| 6topRxReCells        | Received cell relocated command                |
| 6topTxReCells        | Transformed cell relocated request             |
| 6topTxReBund         | Transformed bund relocated request             |
| 6topTxAddResp        | Transformed response to add a time cell         |

**Table 8.** IPv6-based WSN side transaction-based features.

| Feature                | Description                               |
|------------------------|-------------------------------------------|
| droppedMacRetries      | Dropped because of exceeded               |
| FailedEnqueue          | Preamble Retrans Max                      |
| droppedData            | Dropped because of data failed to enqueue |
| droppedNoRoute         | Dropped because of no route               |
| droppedQueueFull       | Dropped because of queue full             |
| droppedNoTxCells       | Dropped because of no transported cell     |
| aveSixtopLatency       | 6top average latency                      |
| AveLatency             | Average latency                           |
| aveQueueDelay          | Average queue delay                       |
| AveHops                | Average hops                              |
| probableCollisions     | Collision times                           |
| chargeConsumed         | Charge consumed                           |
| txQueueFill            | Times for transporting a queue full       |
| dataQueueFill          | The data queue is full                    |
| NumTx                  | The number of transported packets          |

**Feature data standardization and feature selection**

The generated security feature data set is denoted by $X$ with $n$ feature data. The dimension of each feature data is denoted by $q$. Equations (1) and (2) represent the security feature data set $X$ and the sample $x_i$ in the data set, respectively

$$X = \{x_1, x_2, \ldots, x_i, \ldots, x_n\} \quad (1)$$

$$x_i = \{x_{i1}, x_{i2}, \ldots, x_{iq}\} \quad (2)$$

**Standardization**

Box–Cox transformation. Correlation analysis and machine learning algorithms have a default requirement that data follow the normal distribution. However, in reality, data seldom follow the normal distribution.
Box–Cox transformation can reduce unobservable errors and predict the correlation of variables to a certain extent. Therefore, before performing the feature correlation analysis, we use the Box–Cox transformation to bring the data as close to the normal distribution as possible.

\[
X_{\text{Box–Cox}(\lambda)} = \begin{cases} 
\frac{x^{\lambda} - 1}{\lambda}, & \lambda \neq 0 \\
\ln(x), & \lambda = 0 
\end{cases} 
\]

(3)

It can be seen from equation (3) that the final form of Box–Cox transformation is determined by \(\lambda\):

1. When \(\lambda = 0\), Box–Cox transformation is a logarithmic transformation.
2. When \(\lambda = -1\), it is equivalent to a reciprocal transformation.
3. When \(\lambda = 0.5\), it is equivalent to a square transformation.

**Kolmogorov–Smirnov test.** Kolmogorov–Smirnov test is used to determine the normal distribution of features. It involves the degree of consistency between the eigenvalue distribution and the completely theoretical continuous distribution.

Equation (4) is the cumulative distribution function \(F_n(x)\)

\[
F_n(x) = \begin{cases} 
0, & x < x_1 \\
k, & x_k \leq x < x_{k+1}, k = 1, 2, \ldots, n - 1 \\
1, & x \geq x_n 
\end{cases} 
\]

(4)

Equation (5) is the Kolmogorov distribution function \(f(x)\)

\[
f(x) = \frac{\sqrt{2\pi}}{x} \sum_{n=1}^{\infty} \frac{e^{-\frac{(2\pi x)^2}{4n^2}}}{n^n} 
\]

(5)

**Feature correlation analysis.** Correlation analysis is a statistical evaluation technique used to determine the relationship between features. This technique is used to study the relationship between the features of the training set and test set.

**Pearson’s correlation coefficient.** Pearson’s correlation coefficient (PCC) is used to study feature correlation between the training set and test set, without considering labels or categories. PCC is a measure of the strength and direction of the linear correlation between two features.

Equation (6) is the PCC between features \(f_1\) and \(f_2\)

\[
PCC(f_1, f_2) = \frac{\sum_{i=1}^{n} (x_{if1} - \bar{x}_{f1})(x_{if2} - \bar{x}_{f2})}{\sqrt{\sum_{i=1}^{n} (x_{if1} - \bar{x}_{f1})^2 \sum_{i=1}^{n} (x_{if2} - \bar{x}_{f2})^2}} 
\]

(6)

In equation (6), \(x_{if1}\) and \(x_{if2}\) are the values of features \(f_1\) and \(f_2\), respectively.

The calculated value of PCC can vary from +1 to 0 to −1. A positive value of PCC indicates that two features are positively related, whereas a negative value of PCC indicates that two features are negatively related.

**Gain ratio.** The gain ratio is used to classify the correlation between features, and it considers the corresponding instance labels. The analysis aims to find features that distinguish between normal traffic instances and attack traffic instances.

Splitting information is the potential information generated by splitting the security feature data set \(X\) into \(m\) blocks. Equation (7) is used to calculate the splitting information.

\[
\text{SplitInfo}(X) = -\sum_{j=1}^{m} \frac{|X_j|}{|X|} \times \log_2 \left( \frac{|X_j|}{|X|} \right) 
\]

(7)

In equation (7), \(X\) represents the security feature data set with \(n\) instances, and \(m\) represents the number of results corresponding to the feature \(f\).

The average information entropy required to classify an instance is expressed in equation (8)

\[
G(X) = -\sum_{i=1}^{k} p_i \log_2 (p_i) 
\]

(8)

In equation (8), \(p_i\) represents the probability that an instance in the data set belongs to the class \(i\). \(k\) represents the number of label categories in the data set.

Based on the feature \(f\), \(X\) is divided into \(i\) different groups, and the expected information gain \(E(f)\) is defined in equation (9)

\[
E_f = \sum_{i=1}^{m} \frac{G(X)}{X} X_{1i} + X_{2i} + X_{3i} + \cdots + X_{mi} 
\]

(9)

Therefore, the information gain before and after splitting can be calculated using equation (10)

\[
\text{Gain}(f) = G(X) - E(f) 
\]

(10)

The gain ratio is defined as the ratio between the information gain and split information

\[
\text{GainRatio}(f) = \frac{\text{Gain}(f)}{\text{SplitInfo}(X)} 
\]

(11)

**Intrusion detection algorithm**

The proposed intrusion detection algorithm proposed is an anomaly detection method for a single classification problem. It is a variant of the KNN algorithm, which aims to solve the shortcomings of the KNN algorithm
with high computation and lazy learning. In IPv6-based WSN intrusion detection, the intrusion detection algorithm needs to distinguish between normal traffic and abnormal traffic. The key assumption of the proposed intrusion detection algorithm is that normal data points appear in dense neighborhoods and abnormal data points are far from neighbors.

**Quantification method of grid structure.** Each data object is quantified into a p-dimensional space. The p-dimensional space is divided into a continuous square grid space composed of a fixed size. Assume the diagonal of the hypercube grid to be d/2.

For the data dimension p = 2 in the data set, quantify each data object into a two-dimensional (2D) space. The 2D space is divided into a continuous square grid space composed of a fixed size; the grid structure diagram is shown in Figure 3. \( \text{Cube}(u_1, u_2) \) represents the grid at the intersection of row \( u_1 \) and column \( u_2 \). The grid set of the nearest neighbors of the grid is represented in equation (12).

\[
\text{Neighbor}(\text{Cube}(u_1, u_2)) = \{\text{Cube}(v_1, v_2) | v_1 = u_1 \pm 1, v_2 = u_2 \pm 1, \text{Cube}(v_1, v_2) \neq \text{Cube}(u_1, u_2)\} \tag{12}
\]

In 2D space, the length of the grid is \( L \), and \( L = d/2\sqrt{2} \). In the p-dimensional space, the diagonal length of the hypercube is \( \sqrt{p}L \), so in the p-dimensional space, \( L = d/2\sqrt{p} \), and the hypercube is represented as \( \text{Cube}(u_1, u_2, ..., u_q) \), representing the grid at coordinate \( (u_1, u_2, ..., u_q) \). Equation (13) represents the neighbors of the hypercube.

\[
\text{Neighbor}(\text{Cube}(u_1, u_2, ..., u_q)) = \{\text{Cube}(v_1, v_2, ..., v_q) | v = u \pm 1, \text{Cube}(v_1, v_2, ..., v_q) \neq \text{Cube}(u_1, u_2, ..., u_q)\} \tag{13}
\]

The grid structure has the following geometric properties:

1. The distance between any data objects in a grid is at most \( d/2 \).
2. \( p \in \text{Cube}(u_1, u_2, ..., u_q) \), \( q \in \text{Cube}(v_1, v_2, ..., v_q) \) where the distance between any data object \( p \) and \( q \) is at most \( d \).
3. \( x \in \text{Cube}(u_1, u_2, ..., u_q) \) and detection area \( DR(x) \) be a hypercube with data \( x \) as the center and \( d \) as the diagonal length. \( \text{Cube}(u_1, u_2, ..., u_q) \) can cover the detection area, as shown in Figure 3.

**Algorithm training.** The training process of the intrusion detection algorithm involves using the generated security feature data set to adjust the parameters of the algorithm (a KNN classifier) to meet the requirements of intrusion detection. The proposed intrusion detection algorithm analyzes the relationship between the data and the label in the security feature data set. Thus, the algorithm can learn to infer the affiliation of new data. In the training process, the security feature data are projected into the grid structure.

The maximum and minimum values of the \( i \)th feature are \( \max_i \) and \( \min_i \). In this case, the \( i \)th dimension boundary of the grid structure is limited by \( \max_i \) and \( \min_i \).

If the training data remain unchanged, the boundary can be fixed. However, the IDS needs to update the training data regularly while retraining the model. Therefore, it is necessary to set aside appropriate redundant space and leave a margin at the current boundary for the online update, the grid structure can then capture data outside the current boundary.

In addition, the coefficient \( c \) is introduced to translate the data, where \( c > |\min_i|, \min_i = \{\min_i | i = 1, 2, ..., q\} \); the entire feature space is transferred to the positive coordinate space to avoid negative component values in the security data set.

For the hypercube position in which the data \( x_i \) is located as \( [u_i] \), where \( u_i = (x_i + c)/l \), the binary code \( 2^b \) is used to represent the hypercube, with the hypercube position encoded as \( \text{pos} = u_i \leq (q - i) \bmod \text{pos} \). The binary code must be able to cover the hypercube position. The maximum hypercube position is \( (\max_i + c)/l \), where \( \max = \{\max_i | i = 1, 2, ..., q\} \). Therefore, \( (\max_i + c)/l \leq 2^b \).

**Intrusion detection.** Equation (14) is defined as the alternative detection area of test data. The alternative detection area is shown in Figure 3

\[
\text{DR}(X) = \{\text{Cube}(v_1, v_2, ..., v_q) | v = u_i, u_i + c_i\} \tag{14}
\]

In equation (14)
The intrusion detection rules are described as follows:

- If there are at least \( k \) data points in the grid, the test data falling into the grid are always normal.
- If the data points in the grid are less than \( k \), determine the number of data points in the detection area DR to determine whether the number of data points in the replacement DR is greater than \( k \). If it is greater, the test data are normal.
- If the data in DR are less than \( k \), continue to analyze the data in the \( \text{Neighbor}(\text{Cube}) \); if the number of data points in \( \text{Neighbor}(\text{Cube}) \) is greater than \( k \), the test data are normal; if the number of data points in \( \text{Neighbor}(\text{Cube}) \) is less than \( k \), the test data are abnormal.

**Verification and result analysis**

This section describes the test platform for verifying and analyzing the intrusion detection mechanism. Figure 4 shows the test platform, which is built with an IPv6-based WSN platform independently developed by our laboratory. The platform has obtained the IPv6 Ready Phase-2 Logo, which designates the consistency of the IPv6 protocol and device interoperability. The platform includes one PC, one 6LoWPAN gateway, and fifteen 2.4 GHz band 6LoWPAN nodes.

The UNSW-NB15\textsuperscript{30} data set is used as the feature data set of the IPv6-based Internet side. The raw network packets of the UNSW-NB15 data set were created using the IXIA PerfectStorm tool in the Cyber Range Lab of the Australian Centre for Cyber Security for generating a hybrid of real modern normal activities and synthetic contemporary attack behaviors.

The 6TiSCH Simulator\textsuperscript{31} is used to simulate normal activities and attack behaviors of IPv6-based nodes. After the simulation, a DAT file is generated as the feature data set of the IPv6-based wireless network side. The simulation has 50 nodes, 5\textsuperscript{k} slot frames, a slot frame length of 101 cells, and a cell duration is 10 ms.

**Data set analysis**

The security feature data set is divided into training data set and test data set. The skewness, kurtosis, and PCC of the data set are analyzed.

Figure 5 shows the skewness of IPv6-based wireless network side data set. In the training data set, features 20, 23, 25, 33, 39, 40, and 42 are positively skewed, and features 39 and 41 are negatively skewed. The training data set and the test data set have almost equal skewness, and it can be inferred that they have similar distributions.

Figure 6 shows the kurtosis of IPv6-based wireless network side data set. Most of the features of the training data set and the test data set have flat kurtosis. Features 19, 20, 38, 40, 42, and 43 have positive kurtosis.
kurtosis. The training data set and the test data set have similar kurtosis.

The PCC of IPv6-based wireless network side data set is shown in Figure 7, most of the correlations between the features remain balanced, there is no excessive correlation, and no correlation, such features are acceptable features. Acceptable-related features account for more than 70%, and the data set has good correlation.

**Overhead analysis**

The intrusion detection algorithm should be as light as possible to ensure that it can maintain optimal network performance when used in a resource-constrained environment such as IPv6-based WSNs. Therefore, the overhead of the intrusion detection algorithm is evaluated. Assume that retraining the algorithm necessitates \( n_1 \) feature data, \( q \) feature dimension of the data, and NP size. The number of non-zero hypercubes is \( S \). The overhead of the proposed algorithm is discussed in this section.

**Computational complexity.** Computational complexity determines the detection efficiency of an algorithm. The intrusion detection algorithms mainly include training the algorithm and intrusion detection, both of which are completed in the intrusion detection console.

The computational complexity of projecting each training data to the hypercube is \( O(n_1) \) and of counting and sorting data in the hypercube is \( O(n_1 \log(n_1)) \). Address encoding requires \( O(n_1) \). Therefore, the total computational complexity of the learning process is \( O(n_1 \log(n_1)) \).

In the detection process, each data will generate computational complexity from \( n_1 \log(n_1) + 1 \) to \( n_1 \log(n_1) + 2^q - 1 \); therefore, the computational complexity of the detection process is \( O(n_1 \log(n_1)) \).

The results show that the computational complexity of each function varies linearly or logarithmically with

---

Figure 7. PCC of feature data set.
the number of data \( n_1 \). This indicates that the training and detection efficiency of the algorithm is stable, and it can run effectively on the intrusion detection console.

**Communication overhead.** The communication overhead generated during the detection process mainly includes the feature data collected by the intrusion detection device and sent to the gateway and the intrusion response.

A packet payload sent by the intrusion detection device to the gateway is 4 bytes. The feature data sending period is \( T \). The number of nodes is \( N \), the node data message payload is 4 bytes, and the data sending period is 60 s. Intrusion detection devices need to send \( n_1 \) feature data to retrain the algorithm.

During the intrusion response process, the offline command message length is 14 bytes, and the broadcast offline message length is 12 bytes. In a training cycle, the communication overhead of the feature data is 4 \( n_1 \) bytes. Without considering packet forwarding, the total overhead in the network is at least 
\[
4NT(n_1/N)/60 + 4n_1 = n_1T/15 + 4n_1 \text{ bytes.}
\]
This result shows that the communication overhead of feature data accounts for \( 60/(T + 60) \) of the total network communication overhead during the training process. \( T \) is typically much larger than the node data transmitting period, so the communication overhead is acceptable.

**Storage overhead.** Since feature data of the IPv6-based WSN changes constantly, the NP of the intrusion detection model changes accordingly, and the NP is updated online. Therefore, the intrusion detection console stores only the NP, the length of the position-coding unit is \( hp \) bits, and the amount of data in the hypercube is \( \log(n_1) \) bits at most. Therefore, the total storage overhead is 
\[
(hp + \log(n_1))S.
\]

**Performance analysis**

First, the feasibility of the intrusion detection mechanism is verified. When an attacking node is detected, the intrusion detection console records the device address of the attacking node in an offline command message. Then, it sends a message to the gateway to take the attacking node offline and then broadcasts the node’s offline information to other nodes to update the network topology.

The algorithm’s intrusion detection performance and efficiency were evaluated in terms of ACC, the FPR, receiver operating characteristic (ROC) curve, and CPU running time. ACC is the percentage of all normal and abnormal records that are correctly detected. FPR is the percentage of incorrectly identified abnormal records. The ROC curve represents the relationship between the true positive rate (TPR) and FPR, reflecting the algorithm’s overall performance. scikit-learn\(^{32}\) is a Python module comprising a wide range of state-of-the-art machine learning algorithms for medium-scale supervised and unsupervised problems. We compared the proposed model’s performance to intrusion detection models trained on this study’s feature data using DT,\(^{33}\) ANN,\(^{34}\) random forest (RF),\(^{35}\) KNN,\(^{36}\) AdaBoost,\(^{37}\) logistic regression (LR), and Bayesian algorithms;\(^{38}\) these algorithms are from scikit-learn.

Two experiments were conducted to verify the intrusion mechanism’s performance further. In both experiments, the security feature dimension was \( q = 10 \). These security features are drawn from the top four and top six features in descending order of the gain ratios of the IPv6-based Internet side and the IPv6-based wireless network side, respectively.

Experiment 1 was conducted for preliminary verification of intrusion detection performance. The coefficient \( L \) is set to a fixed value, and ROC is tested to observe the algorithm’s overall performance. In Experiment 1, the algorithm ran 100 times independently, and the experimental results in terms of Area Under Curve (AUC), as shown in Figure 8(a). AUC = 0.87, which indicates that the classifier learning effect is good. Figure 8(b) shows the detection time of a single sample, which is 0.12–0.14 ms. It can meet the requirement of real-time detection. The floating of detection time is caused by judgment conditions in the intrusion detection process.

Experiment 2 tests the robustness of intrusion detection capability and compares and analyzes it with other algorithms. The length of the grid \( L \) is set in [0.86, 1.02]; ACC, FPR, and detection time are calculated in each algorithm run.

Figure 9 shows the experimental results in terms of ACC except for the sixth experiment. The results show that ACC is stable at approximately 90%, which shows the algorithm’s effectiveness. The ACC of the proposed algorithm proposed close to DT, AdaBoost, and RF and better than most of the comparison algorithms.

Figure 10 shows the experimental results in terms of FPR. The proposed algorithm results are close to those of DT, AdaBoost, and RF. It does not exceed 25% and can even achieve 6% FPR, which is better than most comparison algorithms. The FPR indicates that the NP has well expressed the behaviors of nodes in the network, and the algorithm is robust.

The experimental results in terms of detection time are shown in Figure 11. The results of the proposed algorithm proposed are close to those of the LR, DT, and RF algorithms. It can achieve timely detection within 2 ms.

Experiment 1 and Experiment 2 results show that the proposed intrusion detection algorithm’s AUC is
0.87 and ACC is stable around 0.9, which indicates that the classifier has a good learning effect and effective intrusion detection. The proposed algorithm’s FPR is less than 0.25, which indicates that the NP well expressed the behaviors of nodes in the WSN. The algorithm’s detection time of a single sample is stable within 0.12–0.14 ms. The overall detection time of the algorithm is stable within 2 ms, indicating that the algorithm is highly efficient. In addition, the detection time of the intrusion detection mechanism meets the requirement of timely detection.

Analysis and experimental results show that the algorithm proposed in this research can effectively reduce the FPR of intrusion detection, achieving good detection efficiency and ACC. In addition, the inexpensiveness of the intrusion detection mechanism allows for the realization of the real-time detection of malicious attacks in IPv6-based WSNs. The proposed intrusion detection algorithm can achieve better detection performance than other comparison algorithms.

**Conclusion and future work**

This research proposed an intrusion detection framework and mechanism for an IPv6-based WSN. The mechanism is lightweight and efficient, and the NP of the intrusion detection model is trained using the feature data set. The intrusion detection algorithm uses the NP to perform real-time detection of traffic data to achieve rapid detection after a significant number of
devices are connected in the network. In addition, a test platform was developed to verify the effectiveness and performance of the intrusion detection mechanism. Experimental results have shown that implementing the proposed intrusion detection mechanism is reasonable and can be used in IPv6-based WSNs.

The intrusion detection mechanism can only detect active threats; it cannot detect threats in advance. Furthermore, in the face of fake malicious behavior, the attack (threat) source cannot be traced, resulting in some false positives. In the future, we will build the 6TiSCH platform to further verify the proposed intrusion detection mechanism. The intrusion detection algorithm will need to collaborate with expert systems to analyze security situations and prevent attacks in advance. Furthermore, more in-depth research on the nature of networks and attack mechanisms will be critical for developing a comprehensive intrusion detection mechanism.

Declaration of conflicting interests
The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.
Funding
The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work is supported by The National Key Research and Development Program of China (2018YFB1702202) and The Chongqing Talent Plan Project (cstc2021ycjh-bgzxm0206).

ORCID ID
Min Wei https://orcid.org/0000-0003-4326-3946

References
1. Chakrabarti S, Montenegro G, Droms R, et al. IPv6 over low-power wireless personal area network (6LoWPAN) ESC draft code points and guidelines. Internet requests for comments RFC editor RFC 8066, 2017, https://datatracker.ietf.org/doc/html/rfc8066
2. Thubert P, Nordmark E, Chakrabarti S, et al. Registration extensions for IPv6 over low-power wireless personal area network (6LoWPAN) neighbor discovery. Internet requests for comments RFC editor RFC 8505, 2018, https://datatracker.ietf.org/doc/rfc8505/
3. Bornmann C and Shemby Z. Block-wise transfers in the constrained application protocol (CoAP). Internet requests for comments RFC editor RFC 7959, 2018, https://datatracker.ietf.org/doc/rfc7959
4. Feng S, Shi H, Huang L, et al. Unknown hostile environment-oriented autonomous WSN deployment using a mobile robot. J Netw Comput Appl 2021; 182: 103053.
5. Zhang H, Shen S, Cao Q, et al. Modeling and analyzing malware diffusion in wireless sensor networks based on cellular automaton. Int J Distr Sens N. Epub ahead of print November 2020. DOI: 10.1177/150147720927944.
6. StJohns M, Atkinson R and Thomas G. Common architecture label IP6v security option (CALIPSO). Internet requests for comments RFC editor RFC 5570, 2020, https://datatracker.ietf.org/doc/rfc5570
7. Migault D, Guggemos T and Nir Y. Implicit initialization vector (IV) for counter-based ciphers in encapsulating security payload (ESP). Internet requests for comments RFC editor RFC 8750, 2020, https://datatracker.ietf.org/doc/rfc8750
8. Kent S and Seo K. Security architecture for the Internet protocol. Internet requests for comments RFC editor RFC 4301, 2020, https://datatracker.ietf.org/doc/rfc4301
9. Pauly T and Wouters P. Split DNS configuration for the Internet key exchange protocol version 2 (IKEv2). Internet requests for comments RFC editor RFC 8598, 2019, https://www.ietf.org/rfc/rfc8598.html
10. Cao M, Wang L, Xu H, et al. Sec-D2D: a secure and lightweight D2D communication system with multiple sensors. IEEE Access 2019; 7: 33759–33770.
11. Raza S and Magnusson RM. TinyIKE: lightweight IKEv2 for Internet of Things. IEEE Internet Things 2019; 6: 856–866.
12. Raza S, Wallgren L and Voigt T. SVELTE: real-time intrusion detection in the Internet of Things. Ad Hoc Netw 2013; 11: 2661–2674.
13. Wazid M and Das AK. A secure group-based blackhole node detection scheme for hierarchical wireless sensor networks. Wireless Pers Commun 2017; 94: 1165–1191.
14. Althubaity A, Ji H, Gong T, et al. ARM: a hybrid specification-based intrusion detection system for rank attacks in 6TiSCH networks. In: Proceedings of the 2017 22nd IEEE international conference on emerging technologies and factory automation (ETFA), Liassol, 12–15 September 2017, pp.1–8. New York: IEEE.
15. Amaran S and Mohan RM. An optimal multilayer perceptron with dragonfly algorithm for intrusion detection in wireless sensor networks. In: Proceedings of the 2021 5th international conference on computing methodologies and communication (ICCMC), Erode, India, 8–10 April 2021, pp.1–5. New York: IEEE.
16. Choudhary V and Taruna S. An intrusion detection technique using frequency analysis for wireless sensor network. In: Proceedings of the 2021 international conference on computing, communication, and intelligent systems (ICCCIS), Greater Noida, India, 19–20 February 2021, pp.206–210. New York: IEEE.
17. Jiang S, Zhao J and Xu X. SLGBM: an intrusion detection mechanism for wireless sensor networks in smart environments. IEEE Access 2020; 8: 169548–169558.
18. Sharma M, Elmiligi H and Gebali F. A novel intrusion detection system for RPL-based cyber-physical systems. IEEE Can J Electr Comput Eng 2021; 44(2): 246–252.
19. Moustafa N, Turnbull B and Choo KKR. An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of Internet of Things. IEEE Internet Things 2019; 6: 4815–4830.
20. Verma A and Ranga V. ELNIDS: ensemble learning based network intrusion detection system for RPL based Internet of Things. In: Proceedings of the 2019 4th international conference on Internet of Things: smart innovation and usages (IoT-SIU), Ghaziabad, India, 18–19 April 2019, pp.1–7. New York: IEEE.
21. Shen S, Huang L, Zhou H, et al. Multistage signaling game-based optimal detection strategies for suppressing malware diffusion in fog-cloud-based IoT networks. IEEE Internet Things 2018; 5: 1043–1054.
22. Shen S, Hu K, Huang L, et al. Quantal response equilibrium-based strategies for intrusion detection in WSNs. Mob Inf Syst 2015; 2015: 179839.
23. Shen S, Hu K, Huang L, et al. Optimal report strategies for WBANs using a cloud-assisted IDS. Int J Distrib Sens N 2015; 2015: 184239.
24. Zhou H, Shen S and Liu J. Malware propagation model in wireless sensor networks under attack–defense confrontation. Comput Commun 2020; 162: 51–58.
25. Liu J, Wang X, Shen S, et al. A Bayesian Q-learning game for dependable task offloading against DDoS attacks in sensor edge cloud. IEEE Internet Things 2021; 8: 7546–7561.
26. Liu J, Shen S, Yue G, et al. A stochastic evolutionary coalition game model of secure and dependable virtual service in sensor-cloud. Appl Soft Comput 2015; 30: 123–135.
27. Liu J, Yu J and Shen S. Energy-efficient two-layer cooperative defense scheme to secure sensor-clouds. *IEEE Trans Inf Foren Sec* 2017; 13(2): 408–420.

28. Liu J, Wang X, Shen S, et al. Intelligent jamming defense using DNN Stackelberg game in sensor edge cloud. *IEEE Internet Things*. Epub ahead of print 9 August 2021. DOI: 10.1109/JIOT.2021.3103196.

29. Massey FJ. The Kolmogorov-Smirnov test for goodness of fit. *J Am Stat Assoc* 1951; 46(253): 68–78.

30. Moustafa N. *UNSW-NB15 dataset*. IEEE DataPort, 2019, https://ieee-dataport.org/documents/unswnb15-dataset

31. Municio E, Daneels G, Vucinic M, et al. Simulating 6TiSCH networks. *T Emerg Telecommun T* 2019; 30: e3494.

32. Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: machine learning in Python. *J Mach Learn Res* 2011; 12: 2825–2830.

33. Breiman L, Friedman J, Olshen R, et al. *Classification and regression trees*. Belmont, CA: Wadsworth, 1984.

34. Hinton GE and Salakhutdinov RR. Reducing the dimensionality of data with neural networks. *Science* 2006; 313(5786): 504–507.

35. Breiman L. Random forests. *Mach Learn* 2001; 45: 5–32.

36. Yang Y, Slattery S and Ghani R. A study of approaches to hypertext categorization. *J Intell Inf Syst* 2002; 18(2): 219–241.

37. Freund Y and Schapire RE. A decision-theoretic generalization of on-line learning and an application to boosting. *J Comput Syst Sci* 1997; 55: 119–139.

38. MacKay DJC. Bayesian interpolation. In: Smith CR, Erickson GJ and Neudorfer PO (eds) *Maximum entropy and Bayesian methods*. Dordrecht: Springer, 1992, pp.39–66.