LDES: detector design for version number attack detection using linear temporal logic based on discrete event system

Abhay Deep Seth¹ · Santosh Biswas¹ · Amit Kumar Dhar¹

Published online: 17 March 2023 © The Author(s), under exclusive licence to Springer-Verlag GmbH, DE 2023

Abstract
The Internet Engineering Task Force (IETF) has defined routing protocols for low power and lossy networks (RPL) for constrained devices. RPL constructs DODAGs (destination oriented directed acyclic graphs), to optimize routing. RPL ensures acyclic topology with the DODAG version number. However, the control message’s DODAG version number is not authenticated. So, RPL is vulnerable to network resource attack known as DODAG Version Number (DVN) attack. DVN attack creates a packet delay, packet loss, cyclic topology, etc., in the network. This paper proposes a method for detecting DODAG version number attacks. Several existing schemes to defend against the DVN, such as cryptographic techniques, trust-based, threshold-based and mitigation are computationally intensive or require protocol modification. DVN does not change the packet format or sequence of packets, but can still perform attacks and hence fall under the category of stealthy attacks, which are difficult to detect using traditional intrusion detection system’s (IDS). Discrete-event system (DES) based IDS have been applied in the literature for stealthy attacks that achieve low overhead, low false alarm rate, etc. However, the construction of DES-based IDS for network protocol may lead to errors, as modelling is manual. The resulting IDS, therefore, is unable to guarantee its correctness. This paper proposes linear temporal logic (LTL) based DES paradigm to detect DVN. LTL-based paradigm facilitates formal verification of the DES-based IDS, and hence the correctness of the scheme is ascertained. The proposed technique is simulated using the Contiki cooja simulator. When the percentage of spiteful nodes in the network increases, the true positive rate, and packet delivery rate drops, while the false positive rate and control message overhead increase. The memory requirement for sending the packets and verifying the nodes is minimal. The LTL-based IDS has been formally verified using NuSMV to ensure the correctness of the framework.

Keywords Routing protocol for low power and lossy networks (RPL) · Internet Engineering Task Force (IETF) · DODAG version number attack (DVN) · Destination oriented directed acyclic graphs (DODAG)

1 Introduction
The Internet of Things (IoT) has been overgrowing. It connects a vast network of machines or objects over the Internet. These interconnected machines or objects such as sensors, actuators may consist of resource-constrained devices [17,36]. Various applications like smart hospitals, smart industrial monitoring, smart agriculture, smart irrigation, smart fighting, use IoT devices. In the IoT framework, a wide range of constrained devices with minimal memory, power, and processing capacities construct a low power and lossy networks (LLNs) [24]. LLNs have lossy links and low throughput. Therefore, LLN requires a routing protocol that can fulfill the need of LLN devices. However, there are already many existing routing protocols like OSPF (Open Shortest Path First) Version 2, RIP (Routing Information First) Version 2, DYMO (Dynamic Mobile ad hoc network On-demand routing), that are designed for other networks like wireless, ad-hoc, MANET. Many researchers have applied these routing protocols for LLN devices, but such protocols lack the routing criteria (such as routing state, lossy response, control cost, link cost, node cost) required for LLN devices.

¹ Indian Institute of Technology Bhilai, GEC Campus, Sejbahar, Raipur, Chhattisgarh, India

Abhay Deep Seth
abhays@iitbhilai.ac.in

Santosh Biswas
santosh@iitbhilai.ac.in

Amit Kumar Dhar
amitkdhar@iitbhilai.ac.in
Thus, Routing Protocol for LLNs (RPL) [6,11,13] has been designed that passes all criteria associated with LLNs. RPL based network is open to various security attacks because of their limited potential and open and unguarded distribution. Major security attacks against RPL include internal attacks, e.g., Rank attack, DIS (DODAG Information Solicitation) flooding attack, topological inconsistency attack, and external attacks like DoS (Denial-of-Service), hijacking of IoT device, password-based attacks. A variety of security mechanisms have been taken into consideration for RPL [7]. All these mechanisms can secure or guard the network against external attacks [9,10,12], but it is also vital to secure the network from internal attacks [14,25,26,33,35].

This paper focuses on a resource attack referred to as DODAG Version Number attack. This attack falls into the category of stealthy attacks that are difficult to be detected by standard IDS. A spiteful node alters the version number by illegally incrementing the version number value in the respective field of the DIO control message format. Once the new increased version number of the DIO message format is received, this modification forces the rebuild of the new DODAG. The parent node sends DIO messages to their child nodes. In this way, the DIO messages are exchanged to build the new DODAG. The new DODAG formation can cause loss of reserved energy, unavailability of channels, and even form loops in the routing topology. There is no technique available that can assure the integrity of the DIO field, i.e., version number. Various trust-based [18], cryptography-based approaches (such as key exchange, signature-based approaches, hashing) have been proposed to ensure the security of version numbers in RPL networks. However, these approaches deliver the protocol as heavy-weight for LLN devices. Only a few strategies target to detect version number attacks but require protocol modification and hardware upgradation. [5,8]. Traditional signature and anomaly IDS [12,31,34] show high false alarms as stealthy attacks like DODAG version number attacks do not change the syntax of a network packet or result in a large statistical change of the network parameters. Therefore, in this paper, we apply the Discrete Event System (DES) based modeling to detector design.

The remaining paper is organized as follows. Section 2 presents the operation of the RPL protocol along with the version number attack. Section 3 consists of related work for the detection of version number attacks. The overview of detector design using LTL-based scheme is discussed in Section 4. The proposed scheme is discussed in Section 5. It gives a detailed working principle of the proposed IDS. It consists of an LTL-based DES framework and concerns constructing a DES detector for a version number attack. Section 6 contains the performance evaluation of various simulation scenarios and verifies the DES-based IDS using the NuSMV tool. In Section 7, counterexample for showing the importance of LTL is shown. The concluding remarks are given in Section 8.

2 Routing protocol low power and lossy network (RPL)

2.1 Overview of RPL

RPL is a routing protocol for Low Power and Lossy Network, developed as a distance-vector routing protocol. The RPL protocol is primarily for IoT devices with limited capabilities and wireless sensors. The devices (nodes) use RPL to
construct a Destination Oriented Acyclic Graph (DODAG) [28] for communication. The DODAG consists of three different kinds of nodes: parent (in-between nodes), child (leaf nodes), and root (default gateway), as illustrated in Fig. 1. Each device in a DODAG tree has a rank that helps to reduce upward traffic flow while increasing downward traffic flow. In DODAG, the root node has rank zero, and the parent node’s rank is always lesser than its child to avoid loop creation. The RPL supports three kinds of traffic schemes. In P2P, the communication between two LLN devices occurs, whereas, in P2M, the traffic flows from a root node to LLN devices. In M2P, communication from LLN device(s) to root occurs.

The RPL consists of multiple DODAG networks, where the number of distinct wireless sensor nodes are linked to a DODAG’s gateway node, i.e., root [5]. Each DODAG in the network can be distinguished by some fields - DODAG version number, rank values, DODAG ID, and instance ID. To maintain and establish the DODAG routing, RPL uses ICMPv6 control messages: DODAG information object (DIO), DODAG Advertisement Object (DAO), DODAG Advertisement Object Acknowledgement (DAO-ACK), and DODAG Information Solicitation (DIS). DIO advertises the information to maintain and build the DODAG. The DODAG root begins the construction of DODAG towards the upward route by broadcasting DIO frames. The nodes select the sender as a parent on receiving the DIO frame, and the receiving node notifies the updated data to neighboring nodes in the next DIO. All nodes has an upward route towards the root on completing the DODAG construction process. A rank value is assigned to each node in DODAG with respect to the root node, which denotes the location of a node. A node selects the preferred parent from the parent list, which acts as a default gateway. A node selects the preferred parent to forward a packet towards the root. Upon unsuccessful transmission, the node selects any non-preferred parent to forward the packet, one after the other.

Figure 1 depicts the DODAG construction using RPL. The root node (node A) delivers the DIO frame consisting of the DODAG’s data. Node C, B joins DODAG upon receipt of the DIO frame and responds to node A with the DAO frame. Then, node B forwards a DIO frame containing the latest DODAG data. On receipt of the DIO frame from node B, node D joins the DODAG. After joining DODAG, node D responds to node B with a DAO packet. Node B receives a DIS request response from node E, as no node has joined the DODAG. After node B joined the DODAG, node B sends the DIO frame to node E to join the DODAG. Now, after joining DODAG, node E responds with a DAO frame to node B. After receiving the frames, node B will sum up all the data and forward the DAO frame to its selected parent node A. Finally, node A attains all data about the nodes from their DAO frames to construct a downward route. Similarly, the remaining nodes will join the DODAG.

In RPL, the trickle algorithm transmits the DIO periodically. A rank rule is used to avoid loop creation where a parent node always has a rank lesser than its child node. DODAG loops may occur when DODAG is cyclic. When inconsistencies arise in the DODAG (e.g., due to less power or lossy communication condition, nodes vanish from the network), RPL activates repair mechanisms. Two different repair mechanisms are available: (i) Global repair and (ii) Local repair. Whenever the preferred parent is unavailable, a local repair is triggered. RPL uses a local repair mechanism to find an alternate path to route the packets. This mechanism fails if multiple inconsistencies occur and activate the global repair mechanism. A global repair mechanism increments the DODAG version number to rebuild the whole DODAG tree. The DIO control message consists of the version number. A node checks its current version number with the new version number of the received DIO from its parent nodes. If the latest version number value is large, a node ignores its current rank value, resets the clock, and rebuilds a new DODAG. A global repair procedure in a new DODAG assures an acyclic topology.

### 2.2 Threat model

The RPL protocol is exposed to a large variety of attacks. The characteristics of LLN networks such as resource constraints, lack of infrastructure, dynamic topology, unreliable links etc. make them particularly vulnerable against attacks. The RPL integrates local and global repair mechanism as well as loop avoidance and detection techniques. Even though an attacker is able to bypass security at link layer by exploiting a vulnerability as well as can configure faulty node to disturb the network functioning. Therefore, RPL based attacks targeting the exhaustion of network resources like Flood-
The few prominent RPL related attacks discussed are as follows.

- **Flooding attack:** This attack generates huge amount of traffic to make nodes and links unavailable. An adversary node present either externally or internally in the network can perform flooding attack. The adversary nodes exhaust the resources available of all the network nodes. Using DIS message, an adversary node can perform flooding is known as HELLO flood attack. In RPL networks, an attacker can either broadcast DIS messages to its neighboring nodes which have to reset their trickle timer, or, unicast DIS message to a node which has to reply with a DIO message. In both cases, this attack leads to network congestion and saturation of the RPL nodes.

- **Routing table overhead attack:** The RPL protocol is proactive protocol. It means that the RPL router nodes build and maintain routing tables when the storing mode is enabled for those nodes. The principle of this attack is to advertise false routes using DAO message to saturate the routing table of the targeted node. It prevents the build of new legitimate routes and impacts network functioning. It may also result in memory overflow.

- **Increased rank attack:** In this attack, the rank value of a RPL node is increased intentionally in order to generate loops in the network. The rank value is associated to each node and corresponds to its position in the DODAG structure according to the root node. As previously mentioned, the node rank always increase in the downward direction to preserve the acyclic DODAG. The rank of child node must be greater than rank value of its parents node. If a node wants to change its rank value, it has to update its parent list by removing the nodes having a higher rank than its new rank value. An adversary node advertises higher rank value than the one it is supposed to have. Loops are formed when its new preferred parent was in its prior sub-DODAG and only if the attacker does not use loop avoidance mechanisms. It drains the node batteries power and creates congestion in the RPL network.

- **Version number modification attack:** The version number acts as an indicator for global repair mechanism and carried in DIO message. The root node in the network uses a version number to manage the global repair process. The version number assures that all nodes in the DODAG are present with the routing state. In this attack, spiteful node illegally increments the version number field value in the DIO frame format [3]. A spiteful node forwards the DIO frame to all of its neighbors. When the neighboring nodes receive the DIO frame from the spiteful node with an incremented version number value, they form the latest DODAG tree. The latest DODAG tree formation creates a loop in the topology, resulting in a waste of node energy, increased overhead, and many more.

In this paper, version number modification attack has been considered. The version number modification attack is explained through an example. The network topology in RPL comprises of one root node, several normal nodes and spiteful node facilitate for version number attack. Figure 2a shows the impact of DODAG Version Number Attack. The network consists of 8 nodes where there is one DODAG root (node A), one spiteful node (node F), and the remaining are legitimate nodes. In DODAG, only node A is allowed to update the Version number parameter in DIO message format. Node F joins DODAG as a legitimate node like other nodes, but it becomes spiteful after the DODAG becomes stable. Now, node F sends DIO (dotted arrows) messages to its neighboring nodes, i.e., Node C, E, H, and G (assuming in the transmission range). This DIO message consists of the modified version number. Node C rejects the received DIO message, as it is from the child node. Nodes E, H, and G also receive the DIO message with the updated version number. Upon receiving DIO, Nodes E, H and G update their existing version number and propagate it in the DODAG. It will result in new DODAG tree formation and breaks the single tree into two DODAG's as shown in Fig. 2b. The new DODAG formation is one of the implications of DODAG version number attack, where Node F becomes root node (as shown in Fig. 2b) from the spiteful node (as seen in Fig. 2a).

### 3 Related work

Section 3.1 describes the study of the detection strategies for the DODAG version number attack. As already discussed, for stealthy attacks, e.g., the DODAG Version Number attack, DES-based techniques are implemented. Section 3.2 comprises the background of DES-based IDS techniques and their drawbacks.
3.1 DODAG version number RPL attack detection techniques

The technique presented in SVELTE [30] consists of three modules for detecting routing attacks in a 6LoWPAN network. The first module collects data from the IPv6 at the edge router, and the second module determines the invasion in the traffic. The third module prevents unwanted traffic from entering the 6LoWPAN network. The main issue of the scheme is that it shows high false positive rates.

Authors in [15] present a solution for attacks on VeRA-version number and rank authentication in RPL. This cryptography-based solution secures rank and version number authentication. However, the authors have neither evaluated network performance parameters nor discussed the paper’s implementation.

Ahmet Arõsü et al. [8] have proposed mitigation techniques for the Version Number Attack. The technique eliminates the Version Number coming from the child nodes and allows the node to change its Version Number only when most neighbors with better ranks claim a Version Number update. However, the multiple Version Number attack condition has not been considered in this work.

Firoz Ahmed et al. [5] have proposed a technique for the detection of the version number attack using a cooperative verification technique. In this scheme, the node exchanges verification messages among the two-hop neighbors and collects the version number information from the two-hop neighbors to identify the attacker. The issue with this scheme is that every node should know the address of every two-hop neighboring node. Also, this scheme itself can become the source of the DoS attack by repeatedly sending the verification request messages to misuse the neighbors’ resources.

The detection, as mentioned above, techniques are either the cryptographic schemes (involving resource overheads) or need protocol modification, software updation, etc.

Some attacks will require no change in protocol or header format and neither lead to any significant statistical deviation in the network parameters but can still exploit the vulnerabilities. These attacks are known as stealthy attacks, and DODAG Version Number attack falls in this category. For stealthy attacks, the DES-based IDS technique has been proposed in the literature that has shown impressive results in non-requirement of protocol modification, low resource overheads, low false alarm rate, etc.

In the following subsection, we discuss the DES-based IDS schemes and issues therein.

3.2 DES based IDS techniques

N. Hubballi et al. [20] proposed a LAN attack detection technique using Discrete Event Systems. ARP spoofing is a stealthy attack where signature and anomaly-based IDS have a high false alarm rate. In this work, the IDS detector is designed to detect ARP request and response spoofing using the probing scheme. The scheme has salient features like no protocol change, low false alarm rate, etc. This technique only detects ARP attacks based on the IP-MAC pair conflicts.

In [33], the authors have designed the DES-based IDS detector for the de-authentication attack. The victim’s access point gets disconnected from the network due to the de-authentication frame in the network. As for the IDS of ARP, in this case, also the DES-based IDS for de-authentication requires neither protocol change nor firmware upgradation.

Authors in [21] proposed an event-based detection approach to recognize impersonated IP packets. This approach uses proactive authenticity testing of the obtained packets. The active probing scheme uses discrepancies in the Time-to-Live (TTL) values of the received packets to verify whether the earlier packet was impersonating or not. This method helps to identify impersonated IP packets with the help of the DES-based detector.

The critical problem with the DES-based system is that it begins with designing the model from the English language specification of the protocol under normal and attack conditions [27]. The state-transition model is built manually, which may lead to an inaccurate design. It is observed that inaccurate design can have a significant impact on the entire IDS.

In [22], Jiang and Kumar discussed the failure diagnosis problem for DES with LTL specifications. In the temporal logic setting, the diagnosability of DES is established. The issue of diagnostic testing is reduced to that of verification of the model.

The LDES framework has the facility of verification that is missing in the automata-based DES scheme. So, the LDES scheme is adopted for the detection of the Version Number attack in this paper.

The benefits of the LDES framework over automata-based DES are as follows.

- LTL scheme offers a representation of the user-friendly common language specifications as compared to the automata-based DES [29].
- The proof for the correctness of the IDS can be done easily because the automated method, i.e., model checking, can be used for the verification of the LTL specification as compared to the detector of the state-based DES.

The main contribution of the proposed technique for version number attack detection is described below:

- To detect DODAG Version Number Attack, LTL-based DES framework IDS is proposed.
The IDS designed for detecting DODAG Version Number attack does not require protocol modification can work with low resource overhead etc.

- The LTL-based DES IDS can be verified to ensure the correctness.

## 4 Overview of detector design using LTL-based scheme

This section presents the steps to construct the detector with model variables using the LTL-based scheme.

The tuples of the system model $M_d$ for fault diagnosis (or attack detection) are defined as:

$$M_d = (S_i, S_{i0}, V, \Sigma, \tau, R, AP, L)$$

where

- $S_i$ is finite set of states.
- $S_{i0}$ is set of initial states, $S_{i0} \subseteq S_i$
- $V$ is the set of model variables.
- $\Sigma$ is the set of events. Events can be observable or unobservable.
- $\tau$ is the set of transitions.
- $R \subseteq S_i \times \tau \times S_i$ is the transition relation.
- $AP$ is the finite set of atomic propositions.
- $L$ is a marking function $S_i \rightarrow 2^{AP}$ that marks each state with the set of atomic propositions which are valid at each state.

### 4.1 Steps for diagnosability test and detector design

The steps for the design of detector for $M_d$ are as follows.

1. The first step consist of construction of FSA known as Buchi Automata, which shows the normal behaviour of the system obtained automatically through the LTL specification. The Buchi Automata accepts the input pattern if there exists a self-loop to atleast one of the final states. Buchi Automata consists of 5 tuples are defined as:

$$B_f = (C_f, \Sigma_{AP}, R_f, q_0^f, F_f)$$

where,

- $C_f$ is the set of states.
- $\Sigma_{AP}$ is the set of events
- $R_f$ is the transition relation, where $R_f \subseteq C_f \times \Sigma_{AP} \times C_f$.
- $q_0^f$ is the initial state.

- $F_f \subseteq 2^{C_f}$ is the generalized Buchi automata.

2. This step tests the pre-diagnosability of the system model $M_d$. The pre-diagnosability test is performed to verify whether the fault occurrence can be observed within a finite time after its occurrence. The pre-diagnosability property will automatically hold if the LTL formula satisfies a safety property; it means either failure events are never triggered, or failure states are never visited. With this property, the infinite failure trace can be detected through the observations of the state traces. As a result, no detector can be constructed if a system cannot pass the pre-diagnosability test. The proposition synchronization of $B_f$ and $M_d$ constructs the automata $T_1$. $T_1$ is defined as follows:

$$T_1 = (Q_1, \Sigma, \exists, R_1, Q^1_0, AP \cup F_1, L_1)$$

where,

- $Q_1 = C_f \times S_i$ is the set of states.
- $\Sigma$ is set of events.
- $\exists$ is the set of transitions.
- $R_1 \subseteq Q_1 \times \exists \times Q_1$ is the transition relation.
- $Q^1_0 = \{(q, s) \in Q_1 \mid (q_0^f, L(s), c_f) \in R_f, s \in S_{i0}\}$ is the set of initial states.
- $AP \cup F_1$ is the new set of proposition.
- $L_1$ is the marking function.

When every failure state has its own indicator, then the system model is called pre-diagnosable to a specification. So, if the states are visited infinitely often for every infinite proposition generated in $T_1$, then the system model is considered pre-diagnosable. The model $T_11$ should assure the LTL specification $GF \ f_11$ where $f_11 \in F_1$.

3. The diagnosability of $M_d$ is tested in this step. The failure diagnosis approach captures both safety and liveness failures. The system model tests the diagnosability to the specification formula after completing the pre-diagnosability test. The fault diagnosis shows the detection and identification of the faulty states in the model. The diagnosis means either the fault has already occurred or predicts it will occur (liveness property). A detector can be constructed only after the system $M_d$ is diagnosable. The construction of model for $T_2$ consist of 5-tuples as follows:

$$T_2 = (Q_2, \Delta, S_m, R_2, Q^2_0)$$

where,

- $Q_2 = Q^1_0 \cup \{ q \in Q_1 \mid (q', \tau, q) \in R_1 \}$ is the set of states.
- $\Delta$ is the set of observable events.
• \( \mathcal{M} \) denotes the set of observable equivalent transitions of \( \mathcal{D} \).
• \( R_2 \subseteq Q_2 \times \mathcal{M} \times Q_2 \) is the transition relation.
• \( Q_0^2 \) is the set of initial states.

Now design,

\[
T'_2 = (Q_2, \Sigma, \mathcal{M}, R'_2, Q_0^2), \text{ where }
\]

\[
M^{-1}(lang(T_2)) = M^{-1}M(lang(T_1)) \text{ are equivalent.}
\]

Therefore, language generated through \( T'_2 \) is all unobservable events are added with observable events equivalent to the language generated by \( T_1 \).

Now from \( T'_2 \), design \( T_3 \) as the event synchronization of \( T'_2 \) and \( M_d \).

\[
T_3 = (Q_3, \Sigma, \mathcal{M}, R_3, Q_0^3, AP, L_3) \text{ where,}
\]

• \( Q_3 = Q_2 \times S_i \) is the set of states.
• \( \Sigma \) is the set of events.
• \( \mathcal{M} \) is the set of transitions.
• \( R_3 \subseteq Q_3 \times \mathcal{M} \times Q_3 \) is the transition relation.
• \( Q_0^3 = Q_0^2 \times S_i \) is the set of initial states.
• \( AP \) is the set of atomic proposition.
• \( L_3 = Q_3 \rightarrow 2^AP \) is the marking function.

The language accepted by \( T_3 = lang(T'_2) \cap lang(M_d) \). The diagnosability can be checked using LTL model checking.

4. The construction of the detector \( D = (T_2, M_{\mathcal{D}}) \) is the last step. If any transition is not generated in the language of model \( T_2 \) then the detector gives output as “fault”.

5 Proposed technique: LTL based DES IDS for version number attack

This section provides an outline of the proposed technique to detect version number attacks as discussed in Sect. 2. The network under consideration comprises devices with limited power, energy, and processing. The network is assumed to be dense, and the root node is never compromised. After the formation of the DODAG tree, the spiteful nodes show unexpected (malicious) behavior, that is the attack.

The assumptions made for hosts connected to the DODAG network that is being monitored for version number modification attack are as follows:

1. Each host connected to the DODAG network is allocated a Node ID.
2. A Spiteful node will never be able to stop a genuine node.

Fig. 3 Block diagram of the system components

3. IDS and root are trustworthy nodes (assumed never gets compromised).
4. All genuine nodes connected in the network are assumed to reply with a request packet within a defined interval of time.

5.1 System component and its goal

The block diagram for the IDS used for the detection of the version number modification attack is shown in Fig. 3. The different components are described as follows:

1. **Tables and probe:** This section discusses how an IDS utilises probe and tables to detect version number attack. It implies that the technique requires extra probe messages, i.e., RQP and RSP packets to verify the genuineness of the DIO message. Additionally, tables are used to maintain the data pertaining to the DIO packet information verified from the host for verification. These tables are helpful for separating the spiteful node from the DODAG to save the probes in future.

   - **Tables**
     - **Monitoring table:** This table is used to have a list of all nodes that joined the DODAG network. An IDS performs verification to identify whether the DIO message sender node is from internal DODAG or outside DODAG. The monitoring table is formulated during the DODAG construction. This table maintains information about the nodes like node ID and node address.
     - **Blacklist table:** This table maintains the list of spiteful nodes performing malicious behavior, i.e., version number modification attack. When the IDS decides that sender of modified node is spiteful node, the details of spiteful node gets added into blacklist table. After addition of this information, IDS sends alert to all nodes of DODAG network.
     - **Whitelist table:** This table maintains the list of legitimate nodes when the node is verified genuine. When the IDS decides that sender of node is legitimate node, the details of legitimate nodes are added into whitelist table.

2. **Monitoring table:** This table is used to have a list of all nodes that joined the DODAG network. An IDS performs verification to identify whether the DIO message sender node is from internal DODAG or outside DODAG. The monitoring table is formulated during the DODAG construction. This table maintains information about the nodes like node ID and node address.

   - This table is used to have a list of all nodes that joined the DODAG network. An IDS performs verification to identify whether the DIO message sender node is from internal DODAG or outside DODAG. The monitoring table is formulated during the DODAG construction. This table maintains information about the nodes like node ID and node address.

   - **Blacklist table:** This table maintains the list of spiteful nodes performing malicious behavior, i.e., version number modification attack. When the IDS decides that sender of modified node is spiteful node, the details of spiteful node gets added into blacklist table. After addition of this information, IDS sends alert to all nodes of DODAG network.

   - **Whitelist table:** This table maintains the list of legitimate nodes when the node is verified genuine. When the IDS decides that sender of node is legitimate node, the details of legitimate nodes are added into whitelist table.

\[ \exists \] Springer
node gets added into whitelist table. The table maintains the information like node ID, current version number.

- **Probes**
  - **Request (REQ) packet:** This probe is sent from the IDS to verify only when the node receives DIO message having version number greater than its current version number. The REQ packet consists of following details: *(destination node address (DSNA), IDS address, AskVersionNumber)*.
  - **Response (RSP) packet:** This probe is sent in response of the REQ packet to the IDS. The RSP packet consists of following details: *(IDS address, Target address, Node’s latest Version Number)*.

2. **Version number diagnoser:** The diagnoser is constructed from the DES model under normal and attack conditions. It is the main component of the detection methodology. The construction of the diagnoser are described in subsection 5.3.

3. **DIO message handler:** The DIO Message Handler is responsible for extracting vital information like version number from the DIO frame format. It is responsible for generating the events like RQP, RSP, FIN and u. It also makes use of the value of the DODAG version number (VNN), existing version number (VNO), DSNA version number (PRVN) and Root node’s version number (RVNI). The handler forwards these events and attributes to the diagnoser which determines whether the version number modification attack is present or not.

5.2 Working principle of the IDS

Initially, during new DODAG formation, information of all nodes joining the DODAG will be stored in the monitoring table. In the proposed technique, whenever a node accepts the DIO frame from adjoining nodes, a comparison of the DODAG version number (VNN) in the DIO message with the existing version number (VNO) of the accepting node is performed, as illustrated in Fig. 4. The node also checks whether the VNN is greater than VNO or not. If the VNN is greater than VNO, the receiving node (RN) sends the VNN...
and DIO Sender Node Address (DIO message sender abbreviates as DSNA) to the IDS node (located near root node); otherwise, RN ignores the received DIO message. Now the IDS is invoked to detect and verify the identity of neighboring node DSNA to determine whether the node is spiteful or not. For verification, the IDS node obtains the current DODAG version number with the help of Request (REQ) and Response (RSP) packets.

After joining the network, even if a spiteful node transmits DIO’s without modifying the version number or other spiteful activity, the network functions cautiously. All nodes can observe the transmission behavior of the neighboring nodes, which ensures the neighboring node’s honesty. Each node notices the neighboring node’s transmission behavior by examining the DODAG version number sent by its neighbors in the DIO control message. The honesty of a node depends on whether a node has sent the correct version number in DIO or not to its neighboring nodes.

After receiving DIO messages from its neighbors, a node starts examining the DODAG version number field of the DIO control frame. If DIO has high Version Number (VNN) value than receiving node’s existing Version Number (VNO), the receiving node (RN) sends VNN and DSNA to the IDS. The IDS is located near the root node. Upon receipt of this information, the IDS searches for the DSNA and RN node IDs in the monitoring table to confirm the presence of nodes in the internal network of the DODAG. If the search is successful, the identification procedure in IDS will be initiated to identify whether the neighboring node (DSNA) is malicious or not. For the confirmation of the neighboring node, whether it is malicious or not, the IDS sends REQ = (destination node address, IDS address, AskVersionNumber) message to know the version number. The REQ message is sent to the root node and DSNA asking for the current version number. Both nodes send responses to the IDS on receipt of the REQ message. The RSP = (IDS address, Target address, Node’s latest Version Number) contains version number. Now, the IDS may receive version number from both the node’s response messages, i.e., Root Node’s version number (RVNI) and DSNA version number (PRVN). The IDS verifies whether the RSP message is received from DSNA or not. If the RSP from DSNA is received, IDS checks whether VNN is equivalent to PRVN or not. If the result is false, node (DSNA) is considered as a Spiteful node; otherwise, it compares VNN with RVNI. Even if the RSP message is not received at the IDS, it compares VNN with RVNI. If RVNI is equivalent to VNN, the node is considered as Normal node, otherwise as a Spiteful node. When DSNA is found as a legitimate node, the IDS adds the entry of DSNA node in the Whitelist table. The confirmation node is validated by verifying the temporarily stored version number on the IDS with the version number received from the response messages. If the IDS receives two or more distinct version number values, it then considers the version number sent by the root node. We assumed that the root node is never compromised and always responds. After DSNA is found as a Spiteful node, the IDS adds the entry of DSNA to the Blacklist table. Now, the IDS alerts all the nodes of the DODAG network about the DSNA. For Intrusion Prevention (IP), a spiteful node will be detach from the DODAG network and will not be allowed to communicate with any node in that particular DODAG.

Now, we demonstrate the working of the verification procedure through an example. Initially when the DODAG is constructed current version number of the DODAG is “2” and the details of nodes present in the DODAG network are stored in monitoring table, depicted in Table 1. Considering Fig. 5, node 7 receives the DIO control message with an increased version number, i.e., “3” from node 6. Now, node 7, instead of changing its existing version number with the new DIO version number (VNN), compares them to check if the new version is larger than the old version number. Node 7 will send the new version number and node’s 6 address to the IDS. After receiving the information sent by node 7, IDS searches for the Node IDs of node 7 and node 6 in the monitoring table to ensure that both nodes are present in the internal DODAG network. If the node ID 7 is found in the monitoring table but node ID 6 is not found, IDS searches for node ID 6 in the blacklist table. If the node 6 ID is found in the blacklist table then IDS alerts node 7 in the DODAG that node 6 is a spiteful node, so, another parent node is adjoined and detection process exits; else IDS sends the REQ1 packet to the root node and REQ2 to node 6, asking for the version number. After a while, IDS receives RSP1 and RSP2 from...
the root node and node 6, respectively. Finally, IDS compares the received version number in RSP1 and RSP2 with the VNN and decides whether node 6 is spiteful or normal. If the node is found as spiteful, the IDS adds the information of node 6 to the blacklist table, shown in Table 3 and alerts all nodes of the DODAG about node 6 to disconnect themselves from node 6 and adjoin another parent node.

Consider the scenario that the legitimate node, after registering itself in the whitelist table, sends modified version number value (VNN) to launch the threat. In that scenario the node id can be found but the version number value will not match. For example, consider the scenario where the node 6 is legitimate node and has an entry in whitelist table as shown in Table 2. Now the node 6 sends the modified version number as 3 to neighboring node, i.e., node 7. Now, the IDS searches for the node 6 and 7 ID in the monitoring table. If Node IDs of node 6 and node 7 are found in the monitoring table, IDS searches for node 6 id in the whitelist table. If the node 6 ID is found in whitelist table, IDS compares the version number value (VNO), i.e., already stored in the whitelist table with the modified new version number (VNN) value sent by node 6. If the values compared are same in the whitelist table then IDS sends no REQ message. If the value differs, IDS sends REQ message to verify the version number. The whitelist table helps in restricting the number of REQ messages to be sent or not. The IDS verifies the correct version number. If the verified version number is VNO i.e., stored in the whitelist table, the entry remains same as shown in the Table 2. If the verified version number is VNN, IDS deletes an entry for node id 6 from whitelist and monitoring table as shown in Tables 5 and 4. The IDS adds an entry of node 6 id and VNN value in the blacklist table as shown in Table 3.

Even if any spiteful node drops the REQ message or declines to forward the message due to communication breakdown, the scheme will work. It is so because other intermediate nodes would have received the response messages other than the spiteful node. If any node does not send any response message, that node can be called spiteful. The response message may not be transmitted due to link failure; in this condition, the node needs to send an error message to IDS.

5.2.1 Challenges

The main challenge for IDS is to detect attacks masked by Blacklist technique. The ability of the IDS would be determined through Blacklisting spiteful node performing the version number attack. The main challenge in the design of the IDS framework is to have an approach that can assure the integrity of DIO message field using low weight approach. There are various ways to secure version number DIO field using trust-based approach, cryptography-based approaches, hardware upgradation etc. These traditional
approaches either require change in syntax of a network packet or generates high amount of traffic in the DODAG network. In the proposed technique, the IDS uses extra messages, i.e., RQP and RSP to detect a spiteful node in a DODAG. So, the low weight IDS framework is proposed which does not require change in hardware or syntax of the network packet except two extra messages to detect version number modification attack. It may be noted that the extra probes are as per the RPL protocol and hence do not require change in protocol. The proposed approach also generates limited traffic in the DODAG.

Another challenge is to verify the correctness of an IDS. This paper mainly focuses on the LTL based DES (LDES) to construct the IDS that detects DODAG version number attack. The LDES is used to render the IDS framework less vulnerable to errors. LDES also performs model checking at every stage of IDS development to verify the correctness, i.e., from modeling to detector design.

The basic idea used in the active IDS for version number attack involves sending RSP and REQ packets to nodes in the DODAG for observations (like changes in version number). It may be noted that development of IDS require a number of manual procedures namely, modeling the attacks from English language specifications of the protocols and vulnerabilities, IDS design etc. Although the protocols and vulnerabilities for IDSs are complex but existing IDSs in the literature lack in formal modeling and verification paradigm. Similarly, the proposed scheme has not been formally verified. So, from the above issue it may be stated that an efficient IDS for detecting version number attack should be verifiable for correctness, dynamic, scalable etc. Therefore, we use formal verification techniques to ensure there are no design flaws in the IDS before its implementation. Most of these parameters are achievable through LTL-based DES framework. The LTL specification describes safe property and use model checking for system verification.

In the next subsection, the DES-based system model is designed for version number attack. Further, this system model helps in designing the detector for version number attack. The LTL-based DES framework helps in modeling verification to ensure the correctness of the IDS.

5.3 DES modelling for version number attack and its detector

This section presents the DES model for version number attacks in normal and attack conditions. The LTL specification for normal behavior with its Buchi model and NuSMV model checking for the LTL formula are presented. The steps as discussed in Sect. 4, i.e., the IDS construction, are introduced.

-Model variables

The Model variables used for DES modeling of Version Number attack are [VNO, VNN, DIOVN, DIOPVN, PRVN, RVNI]. The description of all model variables are as follows:

- \( VNN \) → New Version Number
- \( VNO \) → Old version Number
- \( DIOVN \) → Version Number obtained in current DIO frame
- \( DIOPVN \) → Version Number obtained in previous DIO frame
- \( RVNI \) → Version Number received in RSP1
- \( PRVN \) → Version Number received in RSP2

-Event set

Based on changes in the network, the events are listed as follows.

- \( DIO \): DODAG information Object Control Frame
- \( RQP \): Request frame from IDS
- \( RSP1 \): Response frame from Root node
- \( RSP2 \): Response frame from Parent Node
- \( FIN \): The IDS initiates a clock, after a RQP is sent. It maintains track of the \( T_{req} \) interval in which the responses should arrive. After time finishes, FIN event is triggered.
- \( u \): unobservable event to model the occurrence of attack.

-Transition

A transition is a tuple with three values i.e., \( \sigma, \text{check}(v), \text{assign}(v) \).

For example, \( \langle DIO, VNN > VNO, VNN \leftarrow DIOVN && VNO \leftarrow DIOPVN \rangle \) is a transition, where, DIO event is triggered, with the assignment of DIOVN to VNN, and DIOPVN to VNO. There is a checking condition also that checks whether VNN is greater than VNO. \( \langle RSP1, VNN==RVNI, - \rangle \) is another transition where event RSP1 triggers after checking whether VNN is the same as RVNI or not. No assignment operation is performed in the third field of the transition.

-Proposition set

The set of propositions are listed as follows.

- \( P1 \) → Source state for the detector.
- \( P2 \) → DIO frame is received. After any transition of DIO event occurs, proposition P2 is true.
- \( P3 \) → QRP has been sent. After any transition of QRP event occurs, proposition P3 is true.
- \( P4 \) → RSP2 frame is received having VNN \( == \) PRVN. After any transition of RSP2 event occurs, proposition P4 is true.
- \( P5 \) → RSP2 frame is received having VNN \( != \) PRVN. After any transition of RSP2 event occurs, proposition P5 is true.
- P6 $\rightarrow$ RSP1 frame is received having VNN $==$ RVNI. After any transition of RSP1 event occurs, proposition P6 is true.
- P7 $\rightarrow$ RSP1 frame is received having VNN $!=$ RVNI. After any transition of RSP2 event occurs, proposition P7 is true.
- P8 $\rightarrow$ $T_{req}$ expires. After any transition of FIN event occurs, proposition P8 is true.

-Mask function

This function considers all the events viz. DIO, RQP, RSP1 and RSP2 are measurable. The event FIN is also measurable but event $u$ is unmeasurable. All defined model variables are also observable. Therefore, Mask Function is as follows:

$$M_D = DIO, M(RQP) = RQP, M(RSP1) = RSP1, M(RSP2) = RSP2, M(FIN) = FIN, M(u) = \epsilon.$$ 

The network model for version number attack under normal and faulty conditions are shown in Fig. 6. The model is represented as $M_d$ and its description as follows:

$$M_d = (S_i, S_{i0}, \tau, AP, L, \Sigma, R, V)$$

where,

- $S_i = \{ S1, S2, S3, S4, S5, S6, S7, S8, S9, S10 \}$
- $S_{i0} = \{ S1 \}$
- $\tau = \{ t1, t2, t3, t4, t5, t6, t7, t8, t9, t10, t11 \}$
- $AP = \{ P1, P2, P3, P4, P5, P6, P7, P8 \}$
- $L$ is the marking function given by $L(S_i) \rightarrow 2^AP$ that can be seen from Fig. 6.
- $\Sigma = \{ DIO, RQP, RSP1, RSP2, FIN, u \}$
- R shows the transition relation depicted in Fig. 6.
- $V = \{ VNO, VNN, DIOVN, DIOPVN, PRVN, RVNI \}$

In system’s DES modeling, each transition is divided into three value tuples as depicted in Fig. 6. The transitions are represented in the form of $t_i$, instead of three tuples, to avoid complications in the figures.

For the both ‘normal and failure (attack)’ scenarios, the DES model is described, as depicted in Fig. 6 is given below.

Normal scenario:
Initially, the system is normal and is in state S1. When there is a DIO frame, a transition from S1 to S2 occurs. In this transition, the model variables $VNN$ and $VNO$ are initialized by $DIOVN$ and $DIOPVN$, respectively. Along with the initialization, it checks whether the $VNN$ is greater than $VNO$ or not. If the comparison is true, transition $(t1)$ moves to the S2 state; otherwise, it remains in the initial state. The transition $(t2)$ moves from S2 to S3 when the request $RQP$ frame is received, asking for the version number. This QRP frame is received by the root node and Parent node (node sent the DIO frame). After receiving $RQP$, the root node and Parent node send the RSP1 and RSP2, respectively. The root node
is assumed never to be compromised and reliable. When the response frame \( RSP2 \) is received from the parent node, it verifies the received version number \( PRVN \) with the \( VNN \). In the checking, if \( PRVN \) is equal to \( VNN \), the transition moves from state S3 to S4. This transition is taken only when the frame is received within the required frame response time. Afterward, in this transition, the version number \( RVNI \) received from the response frame \( RSP1 \) is compared with the \( VNN \). If the comparison is true, then this condition is considered normal. The change of state from S4 to S5 occurs due to transition t4. \( RSP2 \) cannot be received due to link failure or any other reason. In that case, the only \( RVNI \) version number is verified with \( VNN \). If \( RVNI \) is same as \( VNN \), the situation is considered as normal. So, transition(t4) moves from S3 to S6 state. Transition (t7) occurs from S6 to state S10 after the \( T_{req} \) time has passed and event \( FIN \) is fired. In the normal situation, the \( VNN \) should be same as \( PRVN \) and \( RVNI \). The \( RSP2 \) may not be received; in that case, if \( RVNI \) is the same as \( VNN \) the situation is considered as ‘normal’.

**Attack scenario:**

When there is a DIO frame, transition (t1) from the initial state S1 to S2 occurs. This transition initializes \( VNN \) and \( VNO \) model variables, respectively, with \( DIOVN \) and \( DIOPVN \). The transition also checks the equality of \( VNN \) with \( VNO \). If the check is found true, transition moves to S2 state, otherwise remains at initial state S1. When the \( RQP \) event occurs, transition (t2) moves to S3. A \( RQP \) frame is sent to root node and Parent node asking for version number. The \( RSP1 \) and \( RSP2 \) events are triggered by root node and parent node, respectively. The \( RSP1 \) and \( RSP2 \) frames contain the version number \( RVNI \) and \( PRVN \), respectively. Further, these responses are to arrive within \( T_{req} \) time after \( RQP \) is sent. A model moves from S3 to S4 on observation of the event \( RSP2 \) by transition t3. After transition t2, responses from the attacker arrives (i) transition (t3) having \( PRVN \) the same as \( VNN \) (ii) transition (t6) not having \( PRVN \) same as \( VNN \), where system moves to state S8. It may be noted that \( RSP2 \) event for transition (t6) is considered malicious. But \( RSP2 \) event for transition (t3) proceeds with response \( RSP1 \). Enabling of transition (t5) is only dependent on \( RSP1 \). The model variable equality is ensured by checking \( RVNI \) with \( VNN \). In case \( RSP1 \) arrives from attacker, the \( RVNI \) is not equal to \( VNN \) in transition (t5). The transition (t5) fires on receipt of single response \( RSP1 \) within \( T_{req} \). It checks \( RVNI \) with the \( VNN \) and t5 moves to state S9. The transitions (t7) is enabled when event \( FIN \) occurs and system moves to state S10.

**5.4 LTL specification**

The LTL specification for the version number attack in the normal scenario is formulated. In terms of events, the specification of the system’s non-failure scenario (no version number attack) is described as:

Either any response with unmatched \( VNN-PRVN \) and \( VNN-RVNI \) should not be encountered, or any response (\( RSP1 \) and \( RSP2 \)) frame should not be detected after the \( RQP \) request frame is sent before the system terminates.

Therefore, according to the transition set, the specification is considered as: Either t5 and t6 (caused by \( RSP1 \) and \( RSP2 \)) should not be detected, or t7 (caused by \( FIN \)) does not occur just after transition t2 before the system terminates.

It can be observed from the proposition set that P5 and P7 are the propositions that are true when transition t6 and t5 occur; respectively, P3 corresponds to transition t2 and P8 is true when t7 exists, and the system terminates.

The specification is described in the form of LTL formula \( f_1 \) as:

\[ ((¬P5 \land ¬P7) \land (P3 \rightarrow X¬P8)) \lor P8 \]

Using LTL, the system’s non-failure scenario is stated logically and unambiguously. The conversion of the LTL formula into a Buchi Automata is done automatically. Due to this, the LTL-based framework provides a proper and suitable way to state specifications compared to the state-based scheme.

**5.5 LTL-based DES detector for version number attack**

The construction steps for the detector are as follows:

1. **Construction of buchi automata** The Buchi Automata \( B_f \) is obtained from the given LTL specification as follows.

\[ B_f = (C_f, \Sigma_{AP}, R_f, q^0_f, F_f) \] is shown in Fig. 7 where,

\[ - C_f = \{ B0, B1, B2, B3, B4, B5, B6, B7, B8, B9, B10 \} \text{ is the set of states.} \]

\[ - \Sigma_{AP} = \{ P1, P2, P3, P4, P6, P8 \} \text{ is the set of propositions defining the model corresponds to the event.} \]

The set of propositions are represented as \( A_i \) where, \( i \geq 1 \) (e.g., \( \{ P1, P2, P3, P4 \} \) is written as \( A1 \)).

\[ - R_f \text{ is the transition relation, where } R_f \subseteq C_f \times \Sigma_{AP} \times C_f. \]

\[ - q^0_f = B0 \text{ is the source state.} \]

\[ - F_f = \{ B5, B10 \} \text{ are the final states that satisfies the specification expressed through } F_1. \]

As shown in Fig. 7, \( B_f \) shows five transitions emerging from the source state B0. All the transitions satisfy \((¬P5 \land
The state $Y_{1.2}$ is reachable by the transition $t_1$ from the initial state $Y_{1.1}$ i.e., $(B_1, S_1)$, $t_1(B_1, S_2) \in R_1$ as $L(S_2) = P_2$, $(B_1, P_2, B_1) \in R_{fi}$ and $(S_1, t_1, S_2) \in R$. We can also say, $(Y_{1.1}, t_1, Y_{1.2}) \in R_1$. $Y_{1.1}$ is labeled as $P_1$, i.e., $P_1 \in L_1(Y_{1.1})$ as $P_1 \in L_1(S_1)$. $Y_{1.2}$ is marked as $P_2$, i.e., $P_2 \in L_1(Y_{1.2})$ as $P_2 \in L_1(S_2)$. According to this, $T_1$ is constructed and all equivalent states are merged.

As mentioned in Sect. 4.2, the system model is pre-diagnosable only if $T_1$ visits the $F_i$ states infinitely for every proposition trace generated by $T_1$. In terms of model checking, $T_1$ must satisfy the LTL specification $\forall F_i$. This model checking is performed by NuSMV tool for pre-diagnosibility (is shown in Sect. 6).

2. After completing the pre-diagnosability test, the system model proceeds to test the diagnosability to specification formula. Synthesis of Masked $T_1$: With the use of mask function $M$, $T_2$ is constructed as shown in Fig. 9.

$$T_2 = (Q_2, \Delta, \mathcal{I}_m, R_2, Q_0^2),$$

where

- $Q_2 = \{Y_{1.1}, Y_{1.2}, Y_{3.3}, Y_{6.4}, Y_{7.5}, Y_{7.6}, Y_{5.10}\}$
- $\Delta = \{\text{DIO}, \text{RSP}, \text{RQP1}, \text{RQP2}, \text{FIN}\}$
- $\mathcal{I}_m = \{t_{M1}, t_{M2}, t_{M3}, t_{M4}, t_{M5}\}$. $t_{Mi}$ denotes the observable event equivalent of transition $t_i$ (of $M_d$).
- $R_2$ is the transition relation, as shown in Fig. 9.
- $Q_0^2 = \{Y_{1.1}\}$

The set $Q_2$ contains all those states of $T_1$ where the transition leading into that state is triggered having an observable event. Hence, $Y_{10.10}$ state (in $T_1$) gets eliminated as $t_8$ is the only transition that leads to $Y_{10.10}$ state whose observed event is $\epsilon$.

Unmasking $T_1$: Now from $T_2$, the construction of $T'_2$ is done. $T'_2$ accepts $M^{-1}(\text{lang}(T_2))$. The $T'_2$ is represented using 5-tuples shown in Fig. 10 as follows:

$$T'_2 = (Q_2, \Delta, \mathcal{I}_m, R'_2, Q_0^2),$$

where

- $Q_2 = \{Y_{1.1}, Y_{1.2}, Y_{3.3}, Y_{6.4}, Y_{7.5}, Y_{7.6}, Y_{5.10}\}$

The construction of pre-diagnosibility model $T_1$ is shown in Fig. 8. $Q_0^1 = \{(c_{fi}, S) \in Q_1 \mid (q_0^{fi}, L(S), c_{fi}) \in R_{fi}, S \in S_0\}$. Therefore $Q_0^1 = \{(B_1S_1), (B_2S_1), (B_3S_1), (B_4S_1)\}$ as $L_1(S_1) = P_1$. $B_0, P_1, B_1 \in R_{fi}$, $(B_0, P_1, B_2) \in R_{fi}$, $(B_0, P_1, B_3) \in R_{fi}$, $(B_0, P_1, B_4) \in R_{fi}$ and $S_1 \in S_0$. So, the initial states are $\{Y_{1.1}, Y_{2.1}, Y_{3.1}, Y_{4.1}\}$. The initial states i.e., second $Y_{2.1}$, third $Y_{3.1}$ and fourth $Y_{4.1}$, get merged with initial state $Y_{1.1}$.

Fig. 8 Model $T_1$ for $f_1$

Fig. 9 Model $T_2$ for $f_1$
The detection technique uses the Contiki 2.7 operating system and its Cooja simulator for performance evaluation. Contiki is an open-source and multi-tasking operating system for wireless sensor networks. The Contiki 2.7 update

1. **Final detector** The system model $M_d$ is diagnosable, so its detector can be designed. It can be seen in Fig. 9 that $T_2$ accompany $M_d$ to construct the final detector where,

$$M_{2^+} : \Delta^* \rightarrow \{\text{failure}\}$$

is a partial function where $\forall s \in \Delta^*, M_{2^+}(s) = \text{failure}$ if $s$ is not produced by $T_2$.

Using mask $M$, the detector notices the traces of transitions that are generated by $M_d$. If same trace is not generated by $T_2$, the detector shows that attack has occurred as output. For example, consider $t_1, t_2, t_3, t_5, t_11$ as the transition trace generated through $M_d$, then its equivalent transition trace $t_{M1}, t_{M2}, t_{M3}, t_{M5}, t_{M11}$ is not generated through $T_2$. The transition trace $t_{M1}, t_{M2}, t_{M3}, t_{M4}, t_{M7}$ that is generated by $T_2$. The trace $t_{M1}, t_{M2}, t_{M3}, t_{M4}, t_{M7}$ corresponds to $P1, P2, P3, P4, P7$ which violates $((-P5 \land -P7) \land (P3 \rightarrow X-P8))$. Thus the detector shows the attack as a trace as an output. On the other side, for transitions $t_1, t_2, t_3, t_4, t_7$ the equivalent transition trace is $t_{M1}, t_{M2}, t_{M3}, t_{M4}, t_{M7}$ that is generated by $T_2$. The trace $t_{M1}, t_{M2}, t_{M3}, t_{M4}, t_{M7}$ corresponds to $P1, P2, P3, P4, P6$ which coheres to $((-P5 \land -P7) \land (P3 \rightarrow X-P8)) \cup P8$. Thus, the detector shows normal for this case as output. Any transition that does not satisfy the property i.e., any transition is not generated in the language of the model $T_2$ then detector shows output as attack.

6 **Performance evaluation**

The performance of the Version Number detection technique has been evaluated through simulation and the NuSMV model checker.

In this work, we considered the random topology with the increased percentage of the spiteful node to analyze the performance changes in the metrics. The simulation topology consists of 20 nodes, i.e., root, IDS, and remaining sensor nodes, as shown in Fig. 12. All nodes are placed in a 100x100m area. In random topology, all nodes are placed randomly except the IDS node. The IDS node lies near the root node. Here, node 1 and node 2 are considered as the root node and IDS node, respectively, as shown in Fig. 12a. In other scenarios, the simulation setup has nodes 19 and 20 as spiteful nodes, depicted in yellow. Figure 12b has only node 20 as a spiteful node, and the scenario shown in Fig. 12c has nodes 19 and 20 as spiteful nodes. In such a way, we have considered there scenarios wherein the number of spiteful nodes increases by 5%.

**Simulation environment**

The detection technique uses the Contiki 2.7 operating system and its Cooja simulator for performance evaluation. Contiki is an open-source and multi-tasking operating system for wireless sensor networks. The Contiki 2.7 update
contains the Contiki RPL protocol [5]. Cooja is a simulator for the simulation of the networks of sensors running over the Contiki operating system.

On a plane square, the Tmote Sky motes are introduced. All motes, i.e., motionless motes, are called static motes. For all motes, the communication and interference range is 50 and 100 meters, respectively. The simulation is verified 20 times for each scenario, and the average values for all parameters are determined. In Table 6, the simulation parameters and their values are given.

By increasing the number of spiteful nodes in random topology, the performance of the detection scheme has been assessed. For an attack or normal case, each simulation runs for 50 minutes. It is assumed that 50 minutes are sufficient to determine the network’s performance and the network’s DODAG becomes stable within three minutes.

**Performance parameters:**

The following performance parameters are considered:

- **Resource requirements:** Average power consumption, and Memory requirements. Average power consumption is calculated based on the average of the total power consumption of nodes except root, IDS, and attacker nodes. Memory requirement shows memory (RAM and ROM) needed to implement the DVN detection technique, i.e., the IDS.

- **Network parameters:** Packet Delivery Rate (PDR) and Control Message Overhead. The ratio between the number of packets delivered (D) to the total number of packets sent (S) from source node to the destination is known as Packet Delivery Rate (PDR). PDR is computed according to the Eq. 1.

\[
PDR = \frac{D}{S}
\]  

Control Message Overhead refers to the total number of control packets sent for the path establishment of the node as well as the verification of the spiteful node.

- **Accuracy:** True Positives Rate (TPR), False Positives Rate (FPR). True Positive Rate (TPR) is calculated as per Eq. 2. In Eq. 2, TP is the number of true positives and P refers to the total number of positive cases. A true positive TP is a case where our proposed solution detects the legitimate node correctly as legitimate.

\[
TPR = \frac{TP}{P}
\]  

False Positive Rate (FPR) is computed as per Eq. 3. In Eq. 3, FP is the number of false positives while N refers to the total number of negative cases. FP is a case in which our proposed mechanism detects legitimate node incorrectly as spiteful node.

\[
FPR = \frac{FP}{N}
\]  

| Parameter                  | Value                      |
|----------------------------|----------------------------|
| Operating system           | Contiki 2.7                |
| Simulator                  | Cooja                      |
| Radio medium               | UGDM                       |
| Simulation area            | 100m x 100m                |
| Node type                  | Tmote Sky                  |
| MAC layer                  | ContikiMAC/6LoWPAN         |
| Number of nodes            | 20                         |
| Number of malicious nodes  | Up to 25%                  |
| Physical layer             | IEEE 802.15.4              |
Table 7  Accuracy metrics values

| % of spiteful node | TPR   | FPR  |
|-------------------|-------|------|
| 5                 | 99.7  | 0.55 |
| 10                | 98.8  | 1.02 |
| 15                | 97.7  | 1.9  |
| 20                | 96.5  | 2.1  |
| 25                | 95.4  | 2.35 |

Analysis on simulation results:
The performance of the simulation parameters in terms of accuracy metrics, i.e., True Positive Rate (TPR) and False Positive Rate (FPR) with the distinct percentage of spiteful nodes are shown in Table 7, and resource requirements like average power consumption and memory utilization are shown in Fig. 13 and Table 8, respectively.

As seen from Table 7, the true positive rate drops when the percentage of spiteful nodes rises. It happens because an increased percentage of spiteful nodes may collide, and therefore, IDS cannot detect all spiteful nodes. It is showed that the proposed scheme functions well with near about 99.7 and 95.5 true positive rate when the percentage of the spiteful node is 5 and 25, respectively. Table 7, it is also shown that false-positive rates rise with the increased percentage of the spiteful nodes. It happens when there is no path to send the version number value to the IDS node to verify the version number.

The simulation uses the emulated sky motes [1] which utilizes the MSP430 controller to determine the memory usage. The controller consists of 10 KB RAM and 48 KB ROM. The verification technique requires the extra overhead of sending the probing packets and verifying the node, which requires extra RAM and ROM usage. In this case, an additional 327 bytes and 743 bytes of RAM and ROM are required, as shown in Table 8. It may be noted that this memory requirement is reasonable. The IDS stores the DIO message VNN and DIO sender’s address to verify the spiteful node through the REQ and RSP message.

The average power consumption results are shown in Fig. 13, which correspond to the RPL+Spitful node and RPL+IDS+Spitful node environments. It can be seen that the power consumption increases with the increase in the percentage of spiteful nodes. It happens since packets are dropped due to a global repair mechanism. This causes the nodes to waste energy due to the repeated transmissions of the packets. Therefore, it leads to high power consumption. When the efficiency of the IDS is checked, we can see that power consumption has been reduced compared to the RPL+Spitful node environment.

The performance of the network parameters, i.e., Packet Delivery Rate and Control Message Overhead against the distinct percentage of spiteful nodes, are shown in Figs. 14 and 15, respectively. The PDR performs better when there is no spiteful node in the network, i.e., higher than 99. Nevertheless, when 5% of the total nodes are spiteful in the network, the PDR value drops drastically to nearly 40 percent from 99 percent. The PDR value deprecates more with the increase of spiteful nodes. However, our proposed detection technique works efficiently, as it reduces the effect of the Version Number Attack. Figure 14 shows that the PDR of the proposed technique is better in the RPL+IDS+Spitful scenario than the RPL+Spitful scenario. It happens because our proposed technique detects the spiteful node more accurately after the verification of the suspect node. The verification node does not update the new Version Number sent from the suspect node until the verification procedure gets complete and the suspect node gets verified as normal. The packet is sent to the
Fig. 15  Control message overhead versus percentage of spiteful nodes

The performance of the Control Message Overhead (CMO) can be seen in Fig. 15. The CMO of the RPL+Spiteful environment increases drastically even when 5% spiteful nodes of total nodes are present. The CMO increases with the increase of the spiteful nodes in the RPL+Spiteful environment. It happens because of the version number attack behavior. However, the proposed solution’s control overhead is not proportional to the percentage of spiteful nodes, as the proposed scheme prevents DIO’s from the broadcasting process, which broadcasts for the reconstruction of the DODAG. As a result, the CMO is much lower in the RPL+IDS+Spiteful environment than the RPL+Spiteful environment, as shown in Fig. 15.

Presence of tables:
The performance of the parameters, i.e., power consumption, control message overhead and memory usage in the presence and absence of the blacklist and whitelist tables in the IDS, are measured.

– Both tables are present: When both blacklist and whitelist tables are present in the IDS, and the IDS receives a DIO message, the IDS searches for the node id and version number in the whitelist table. If the IDS finds a node id and version number in the whitelist table, no REQ message is sent from the IDS and declares “normal condition”. But, if the node id and version number are not found in the whitelist table, the IDS searches for the same in the blacklist table. No REQ message is sent for verification if the node id and version number is found in the blacklist table and declares “attack condition”. The REQ message is sent for verification only if the node id and version number are not found in both tables. In the presence of both the tables, the IDS maintains the history of normal and attacker nodes. Therefore, only a limited number of REQ packets will be sent when the DIO message is from a new node id or version number.

– Only blacklist table is present: When the blacklist table is present in the IDS, only the history of adversary nodes is maintained. So, REQ messages will be sent from the IDS to verify even for those DIO messages that were previously determined as genuine. In the presence of only the blacklist table, more REQ messages will be sent from the IDS compared to the presence of both tables.

– Only whitelist table is present: When the whitelist table is present in the IDS, only the history of genuine nodes is maintained. So, REQ messages will be sent from the IDS to verify even for those DIO messages that were previously determined as spiteful. In the presence of only a whitelist table, more REQ messages will be sent from the IDS compared to the presence of both tables.

Figure 16 represents the effect over average power consumption in the presence of only blacklist or whitelist or both tables. The power consumption of the IDS is proportional to the number of REQ messages sent from the IDS. It can be seen in Fig. 16 that the power consumption is lowest when both tables are present and increases as the table are reduced.

The performance of the control message overhead in the presence of only blacklist or whitelist, or both tables can be seen in Fig. 17. As in the case of power consumption, the control message overhead is proportional to the number of REQ messages sent from the IDS. The graph of Fig. 17 shows similar trends as power consumption.
The LTL specification used for the normal condition is as follows:

\[(\neg P5 \land \neg P7) \land (P3 \rightarrow X\neg P8) \cup P8\]

**Pre-diagnosibility Model Checking:**

The pre-diagnosibility testing of the system model \(M_d\) (as depicted in Fig. 6) is done through NuSMV model checking. Figure 18a depicts the code snap for the pre-diagnosibility of \(M_d\). The variables \(status\) and \(ev\) are of enumeration type. Both variables \(status\) and \(ev\) are initialized with proposition value (pone) and event (DIO) that can be seen in Fig. 18a. The variable \(ev\) is assigned the value among DIO, RQP, RONE, RTWO, EXP, UNOBV of different events. Proposition values like pone, ptwo, three, pfour, psix, peight are referred through the variable \(status\) as depicted in Fig. 18a. The definition of a transition in NuSMV is not supported, so the code snap consists of the triggering events of transitions (for example, DIO denotes \(t1\)). Furthermore, the propositions are presented in words (e.g., P1 is represented through Pone; similarly, other propositions are represented). The cases are evaluated where P8 is true for the pre-diagnosibility test. The result is true for the specification \((\mathcal{G}\mathcal{F}(P8))\) that can be seen in Fig. 18b. So, it can be said that the system is pre-diagnosable, as model \(T_1\) satisfies the specification.

**Model checking Test for Diagnosability:**

For testing the diagnosibility, the event synchronization of the system model and \(T_2\) model together constructs the model \(T_3\) as depicted in Fig. 11. The code snap for diagnosibility testing and output corresponding to \(T_3\) are shown in Fig. 19 with the help of NuSMV model checker. The system model is said to be diagnosable with respect to \(T_3\), as, model satisfy the specification \((\neg P5 \land \neg P7) \land (P3 \rightarrow X\neg P8)) \cup P8\).

As \(T_3\) is formally verified, the diagosner can be constructed (as per Section 4.5), and the resulting IDS can be considered free of errors.

7 **Validation by LTL**

In this section, we show how the LTL model checking helps in validating the model developed from the protocol of the network under normal and attack conditions. The construction of the DES model is done manually and since the protocol and attacks are complex, manual modeling is prone to errors leading to IDS having flaws. Here, we present an example where the model has a manual error and how LTL based framework detects the same.

The DES model with error is shown in Fig. 20. The DES model is designed manually based on the protocol for both normal condition and attack on the DODAG network for version number.
Table 10  Comparison of proposed approach with related work

| Mechanism                              | PDR (in %) | Overhead (Pkt/sec) | Topology                  |
|----------------------------------------|------------|--------------------|---------------------------|
| Lightweight mitigation [8]             | 96.24      | 5045               | Random, grid-centre       |
| Distributed and cooperative verification [5] | 97        | 1500               | Random                    |
| VeRa [15]                              | –          | –                  | Random                    |
| Proposed approach                      | 98.7       | 1100               | Random                    |

Fig. 18  Pre-diagnosibility testing of the model

Fig. 19  Diagnosibility testing of the model

7.1 DES model with Manual Error

The DES model with manual error is depicted in Fig. 20. The DES model without error is shown in Fig. 6. Figure 20 in comparison with Fig. 6 shows that the transition (t6) is missed that should have been from state S3 to S8 (in the model with error). Transition (t6) is the case when RSP2 event occurs and the value of VNN is not same with PRVN. The DES model with error is represented as $M_{d_error}$ as follows:

$$M_{d_error} = (S_t, S_{i0}, \tau, A_P, L, \Sigma, R, V)$$

where,
Fig. 20  DES model ($M_{d_{error}}$) with transition (t6) missing

Fig. 21  Buchi automata ($B_{f_{error}}$) from error DES model and LTL specification

Fig. 22  Model $T_{L_{error}}$ with error for $f_1$

Fig. 23  Diagnoser error model $T_3$

- $S_i = \{ S1, S2, S3, S4, S5, S6, S7, S9, S10 \}$
- $S_{io} = \{ S1 \}$
- $\tau = \{ t_1, t_2, t_3, t_4, t_5, t_7, t_8 \}$
7.2 Verification by LTL framework

The specification considered for the verification of the model with error is shown in Fig. 20 is same as considered in Sect. 5.4 for DES model without error. The specification is $\neg P_5 \land \neg P_7 \land (P_3 \rightarrow X \neg P_8) \cup P_8$. The LTL specification is represented as $f_1$.

1. Construction of buchi automata for erroneous DES model

The Buchi Automata $B_{f_{\text{error}}}$ is obtained from the given model with error ($M_{d_{\text{error}}}$) and LTL specification is shown in Fig. 21.

$$B_{f_{\text{error}}} = (C_f, \Sigma_{AP}, R_f, q_0^f, F_i)$$

where,

- $C_f = \{B_0, B_1, B_2, B_3, B_4, B_5, B_6, B_7, B_8, B_9, B_{10}\}$ is the set of states.
- $\Sigma_{AP} = \{P_1, P_2, P_3, P_4, P_6, P_8\}$ is the set of propositions defining the model corresponding to the event. The set of propositions is represented as $A_i$ where, $i \geq 1$ (e.g., $\{P_1, P_2, P_3, P_4\}$ is written as $A_1$).
- $R_f$ is the transition relation, where $R_f \subseteq C_f \times \Sigma_{AP} \times C_f$. 
– $q_0^f = B_0$ is the source state.
– $F_1 = \{ B_5, B_{10} \}$ are the final states that satisfying the specification expressed through $F_1$.

The Buchi automata for DES model without error is shown in Fig. 7. In comparison of Fig. 21 with Fig. 7, we can see an extra transition $A_{10}$ from $B_2$ to $B_{10}$ in Fig. 21. The transition $A_{10}$ takes place when no response message is received against the request message sent from the IDS. According to LTL specification, there must be no direct transition from proposition $P_3$ to $P_8$.

2. **Test for pre-diagnosability for erroneous DES model**

The model with error for pre-diagnosability is shown in Fig. 22 that is constructed using synchronization of Buchi automata ($B_f_{error}$) and DES model with error ($M_d_{error}$) of Figs. 20 and 21, respectively. The proposition synchronization is represented as model $T_{1_{error}}$ for $f_1$. $T_{1_{error}}$ consists of 7-tuples as:

$T_{1_{error}} = (Q_1, \Sigma, \delta, R_1, Q_0^1, AP \cup F_1, L_1)$

– $Q_1 = \{Y_{1,1}, Y_{1,2}, Y_{3,3}, Y_{6,4}, Y_{7,5}, Y_{7,6}, Y_{5,10}, Y_{10,10}\}$
– $\Sigma = \text{set of events } \{ \text{DIO}, \text{RQP}, \text{RSP1}, \text{RSP2}, \text{FIN}, u \}$
– $\delta = \{t_1, t_2, t_3, t_4, t_7, t_8 \}$ is the set of transitions.
– $R_1$ is the transition relation as shown in Fig. 22.
– $Q_0^1 = \{Y_{1,1}\}$
– $AP \cup F_1$ represents the set of proposition.
– $L_1$ represents marking function, which is shown in Fig. 22.

The pre-diagnosability model for DES model without error is shown in Fig. 8. In Fig. 22, we can see an additional transition $t_7$ from $Y_{3,3}$ to $Y_{5,10}$ in comparison with Fig. 8. The transition $t_7$ occurs in pre-diagnosability model with error ($T_{1_{error}}$) when no RSP message is received at the IDS for the REQ message sent from IDS. The model checking based validation for $T_{1_{error}}$ performed by NuSMV tool for testing pre-diagnosability is shown in Fig. 24. In terms of model checking, $T_{1_{error}}$ must satisfy the LTL specification $GF F_1$.

3. **Test for Diagnosability for erroneous DES model**

After passing the test of pre-diagnosability, the system proceeds to test the diagnosability for LTL formula ($f_1$).

$T_{3_{error}} = (Q_3, \Sigma, \delta, R_3, Q_0^3, AP, L_3)$ where,

– $Q_3 = \{Y_{(1,1)}, Y_{(1,2)}, Y_{(3,3)}, Y_{(6,4)}, Y_{(7,5)}, Y_{(7,6)}, Y_{(5,10)}, Y_{(10,10)}\}$
– $\Sigma = \text{set of events } \{ \text{DIO}, \text{RQP}, \text{RSP1}, \text{RSP2}, \text{FIN}, u \}$
– $\delta = \{t_1, t_2, t_3, t_4, t_7, t_8 \}$ is the set of transitions.
– $R_3$ is the transition relation as shown in Fig. 23.
– $AP$ is the set of proposition, like $M_d$.

– $L_3 = Q_3 \rightarrow 2^{AP}$ is the marking function, which is shown in Fig. 23.

Figure 23 represents the final diagnoser with error ($T_{3_{error}}$) that is constructed from the DES model with error ($M_d_{error}$). The final diagnoser model $T_{3_{error}}$ without error is shown in Fig. 11. In Fig. 23, the extra transition $t_7$ occurs from state $Y_{(3,3)}$ to $Y_{(5,10)}$ that is not present in Fig. 11. In error model, no response frames RSP1 and RSP2 are received from root node and parent node, respectively, after receipt of RQP frame at state $Y_{(3,3)}$. This shows that for REQ probe message no RSP message is received. In the error DES model, the case of non-receipt of response for REQ probe is also considered as attack. This is a false positive. The non-receipt of response for REQ also contradicts the assumption considered that for every REQ message there must be at least one response message.

The model $T_{3_{error}}$ shown in Fig. 23 is constructed to test the diagnosability. However, the designer of IDS constructs incorrect model $M_d_{error}$ depicted in Fig. 20. The model checking test for diagnosability is shown in Fig. 25. The code snap for diagnosability testing and output corresponding to $T_{3_{error}}$ are shown in Fig. 25 with the help of NuSMV model checker. The output is shown as false and gives the path of error as counterexample. In the counterexample it is shown that model $T_{3_{error}}$ is having path from $P_3$ to $P_8$ that does not satisfy the formula $f_1$. It contradicts the assumption, where at least one response (RSP) message should be received for the request (REQ) message. The LTL-based scheme is important as it helps in verifying the correctness of the diagnoser.

8 Conclusion

The LLN devices are constrained in terms of low resources, high packet loss, etc. The RPL protocol uses DODAG version number to form DODAG (acyclic) topology. However, there is no technique to authenticate the DODAG version number. So, RPL has become vulnerable to DODAG version number attacks. DVN requires neither any change in packet format nor the sequence of packets and hence falls into the category of stealthy attacks. DES-based IDS has been applied for stealthy attacks that achieve low false alarm rates, low overhead, etc. The construction of DES-based IDS for the network protocol is done manually, which may lead to errors. So the designed IDS does not guarantee its correctness. Therefore, the LTL-based DES scheme has been used to design and formally verify the proposed DES-based DODAG version number detection scheme that deals with its correctness. The IDS detects spiteful node and alerts all nodes of the DODAG network. Now, the legitimate nodes as
prevention not allow a spiteful node to communicate within the DODAG.

The proposed detection scheme has been simulated in the Contiki cooja simulator. The simulation results show the accuracy of 99.7% and 95.4% when the percentage of spiteful nodes are 5% and 25%, respectively. The true positive rate decreases with the increase of spiteful nodes, whereas the false positive rate increases with spiteful nodes. It happens since there is no path to send the version number value to the IDS node for its verification. In sending the packets and verifying them, the nodes require extra 327 bytes and 743 bytes of RAM and ROM, respectively, which are minimal when compared to the memory available in IoT nodes.

We plan to advance the proposed research work using a test-bed environment to validate the results in a real environment. Future research could be to analyze the version number attack collaborated with other attacks. Various other DES diagnoser concepts like stochastic approach, logical approach, petri net models etc. can be explored to develop IDS frameworks for different classes of attacks. As IoT networks are also getting popular, applicability of the proposed IDS framework may be explored for IoT frameworks.

Data Availability Statement Data cannot be made available for reasons.

Declarations

Conflict of interest All author declares that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. http://www.eecs.harvard.edu/konrad/projects/shimmer/references/tmote-sky-datasheet.pdf. MoteivCorporation, Tmote Sky: Datasheet
2. Nusmv[online]. http://www.nusmv.fbk.eu
3. Almusaylim, A.Z., Jhanjhi, N., Alhumam, A.: Detection and mitigation of RPL rank and version number attacks in the Internet of Things: Srpl-rp. Sensors (2020). https://doi.org/10.3390/s20215997
4. Agarwal, M., Biswas, S., Nandi, S.: Discrete event system framework for fault diagnosis with measurement inconsistency: case study of rogue DHCP attack. IEEE/CAA J. Automatica Sinica 6(3), 789–806 (2019). https://doi.org/10.1109/JAS.2017.7510379
5. Ahmed, F., Ko, Y.: A distributed and cooperative verification mechanism to defend against dagor version number attack in RPL. In: Proceedings of the 6th international joint conference on pervasive and embedded computing and communication systems—PEC, (PECCS 2016), pp. 55–62 (2016). https://doi.org/10.5220/000509300005050062
6. Alexander, R., Brandt, A., Vasseur, J., Hui, J., Pister, K., Thubert, P., Levis, P., Struik, R., Kelsey, R., Winter, T.: RPL: IPv6 routing protocol for low-power and lossy networks. RFC 6550 (2012). DOI: https://doi.org/10.17487/RFC6550 https://www.rfc-editor.org/info/rfc6550
7. Ammar, M., Russello, G., Crispo, B.: Internet of things: a survey on the security of IoT frameworks. J. Inf. Secur. Appl. 38, 8–27 (2018). https://doi.org/10.1016/j.isact.2017.11.002
8. Arş, A., örs Yalç, S.B., Oktuğ, S.F.: New lightweight mitigation techniques for RPL version number attacks. Ad Hoc Netw 85: 81–91 (2019). DOI: https://doi.org/10.1016/jadhoc.2018.10.022.
9. Aufner, P.: The IoT security gap: a look down into the valley between threat models and their implementation. Int. J. Inf. Secur. 19, 3–14 (2020). https://doi.org/10.1007/s10201-019-00445-y
10. Babun, L., Denney, K., Celik, Z.B., McDaniel, P., Uluagac, A.S.: A survey on IoT platforms: communication, security, and privacy perspectives. Comput. Netw. 192, 108040 (2021). https://doi.org/10.1016/j.comnet.2021.108040
11. Chugh, K., Lasebae, A., Loo, J.: Case study of a black hole attack on 6lowpan-rpl. In: SECURE 2012, Sixth international conference emerging secure information system technology, pp. 157–162 (2012)
12. de Souza, C.A., Westphall, C.B., Machado, R.B., Sobral, J.B.M., dos Santos Vieira, G.: Hybrid approach to intrusion detection in fog-based IoT environments. Comput. Netw. 180, 107417 (2020). https://doi.org/10.1016/j.comnet.2020.107417
13. Deng, H., Li, W., Agrawal, D.: Routing security in wireless Ad hoc networks. IEEE Commun. Mag. 40(10), 70–75 (2002). https://doi.org/10.1109/MCOM.2002.1039859
14. Djejegj, N., Tandjaoui, D., Medjek, F., Romdhani, I.: Trust-aware and cooperative routing protocol for IoT security. J. Inf. Secur. Appl. 52, 102467 (2020). https://doi.org/10.1016/j.jisa.2020.102467
15. Dvir, A., Holczer, T., Buttyan, L.: Vera—version number and rank authentication in RPL. In: 2011 IEEE Eighth international conference on mobile Ad-Hoc and sensor systems, pp. 709–714 (2011)
16. Edmund, J., Clarke, M., Grumberg, O., Kroening, D., Peled, D., Veith, H.: Model Checking. MIT Press Cambridge, MA, USA (1999)
17. El Ksimi, A., Leghris, C.: A new ipv6 security approach for a local network. In: Khoubki, F., Bahaj, M., Ezziyyani, M. (eds.) Smart Data and Computational Intelligence, pp. 17–26. Springer, Cham (2019)
18. Ferraris, D., Fernandez-Gago, C.: Trustapis: a trust requirements elicitation method for IoT. Int. J. Inf. Secur. 19, 111–127 (2020). https://doi.org/10.1007/s10201-019-00438-x
19. Hofer-Schmitz, K., Stojanovic, B.: Towards formal verification of IoT protocols: a review. Comput. Netw. 174, 107233 (2020). https://doi.org/10.1016/j.comnet.2020.107233
20. Hubballi, N., Biswas, S., Roopa, S., Ratti, R., Nandi, S.: Lan attack detection using discrete event systems. ISA Trans. 50(1), 119–130 (2011). https://doi.org/10.1016/j.isatra.2010.08.003
21. Hubballi, N., Tripathi, N.: An event based technique for detecting spoofed ip packets. J. Inf. Secur. Appl. 35, 32–43 (2017). https://doi.org/10.1016/j.jisa.2017.04.001
22. Jiang, S., Kumar, R.: Failure diagnosis of discrete-event systems with linear-time temporal logic specifications. IEEE Trans. Automat. Control 49(6), 934–945 (2004). https://doi.org/10.1109/TAC.2004.829616
23. Kilinc, I.F., Ertam, F., Sengur, A.: Machine learning methods for cyber security intrusion detection: datasets and comparative study. Comput. Netw. 188, 107840 (2021). https://doi.org/10.1016/j.comnet.2021.107840
24. Mayzaud, A., Badonnel, R., Chrisment, I.: A taxonomy of attacks in RPL-based Internet of Things. Int. J. Netw. Secur. 18(3), 459–473 (2016). https://doi.org/10.6633/IJNS.201605.18(3).07
25. Mayzaud, A., Sehgal, A., Badonnel, R., Chrisment, I., Schönwälder, J.: Mitigation of topological inconsistency attacks in RPL.
based low power lossy networks. Int. J. Netw. Manag. (2015). https://doi.org/10.1002/nem.1898
26. Medjek, F., Tandjaoui, D., Djedjig, N., Romdhani, I.: Multicast dis	ack attack mitigation in RPL-based IoT-LLNS. J. Inf. Secur. Appl. 61, 102939 (2021). https://doi.org/10.1016/j.isja.2021.102939
27. Mitra, M., Banerjee, P., Barbhuiya, F.A., Biswas, S., Nandi, S.: Ids for ARP spoofing using ltl based discrete event system framework. Netw. Sci. 2, 114–134 (2013). https://doi.org/10.1007/s13119-013-0019-1
28. Montenegro, G., Hui, J., Culler, D., Kalalnagar, N.: Trans-
mision of IPv6 packets over IEEE 802.15.4 networks. RFC 4944 (2007). https://doi.org/10.17487/RFC4944 https://www.rfc-editor.org/info/rfc4944
29. Pnueli, A.: The temporal logic of programs. In: 18th annual sym-
pouin on foundations of computer science (sfcs 1977), pp. 46–57 (1977)
30. Raza, S., Wallgren, L., Voigt, T.: Svelte: real-time intrusion detec-
tion in the Internet of Things. Ad Hoc Netw. 11(8), 2661–2674 (2013). https://doi.org/10.1016/j.adhoc.2013.04.014
31. Sampath, M., Sengupta, R., Lafortune, S., Simamohideen, K.,
Teneketzis, D.: Diagnosability of discrete-event systems. IEEE Trans. Automat. Control 40(9), 1555–1575 (1995). https://doi.org/
10.1109/9.142626
32. Seth, A.D., Biswas, S., Dhar, A.K.: De-authentication attack detec-
tion using discrete event systems in 802.11 wi-fi networks. In: 2019 IEEE international conference on advanced networks and telecommu-

lications systems (ANTS), pp. 1–6 (2019). https://doi.org/10. 
1109/ANTS47819.2019.9118100
33. Simoglou, G., Violetas, G., Petridou, S., Mamatas, L.: Intrusion detection systems for RPL security: a comparative analysis. Comput. Secur. 104, 102219 (2021). https://doi.org/10.1016/j.cose. 
2021.102219
35. Tsao, T., Alexander, R., Dohler, M., Daza, V., Lozano, A., Richard-
son, M.: A security threat analysis for the routing protocol for low-power and lossy networks (RPLs). RFC 7416 (2015). https:// 
doi.org/10.17487/RFC7416
36. Ziegler, S., Crettaz, C., Ladid, L., Krco, S., Pokric, B., Skarmeta, 
A.F., Jara, A., Kastner, W., Jung, M.: Io6 - moving to an ipv6-based future IoT. Future Internet LNCS 7858, 161–172 (2013)

Publisher’s Note Springer Nature remains neutral with regard to juris-
dictional claims in published maps and institutional affiliations.