DOI: 10.24996/ijs.2025.66.10.33

ISSN: 0067-2904

DISTec: A Detection and Mitigation Technique for DIS Flooding Attacks in the Internet of Things

A. Arul Anitha

Assistant Professor, Department of Computer Applications,J.A. College for Women (Autonomous), Periyakulam, Theni Dt, Tamil Nadu, India,

Received: 12/5/2023 Accepted: 12/10/2024 Published: 30/10/2025

Abstract

The Internet of Things (IoT) is an innovative trend that promotes technological and industrial advancements. As IoT networks expand, RPL (Routing Protocol for Low-Power and Lossy Networks) becomes crucial for efficient routing but is susceptible to a number of security threats and attacks. The DIS (DODAG Information Solicitation) flooding attack is one such attack that compromises network performance. This paper introduces DISTec, a novel technique designed to detect and mitigate DIS flooding attacks in RPL-based IoT networks. DISTec employs an n x m matrix to track DIS message frequencies from nodes, analyzing deviations during the DIS_INTERVAL to identify potential attackers. The Contiki OS Cooja Simulator implemented DISTec on a network consisting of 51 legitimate nodes and one malicious node. DISTec demonstrated a detection accuracy of 98.48%. It effectively isolates attackers by discarding their malicious messages and excluding them from the DODAG, thus enhancing network stability and performance. This research highlights DISTec's effectiveness in maintaining network integrity and reducing the impact of flooding attacks. Future work will explore additional RPL attack scenarios and integrate AI-driven techniques to further improve attack detection and network security.

Keywords: IoT, RPL, DODAG, DIS Attack, DISTec

1. Introduction

The Internet of Things (IoT) is a heterogeneous network of constrained devices that are connected to the global network using IPv6 addressing [1]. RPL is the routing protocol used in the Low-power Lossy Networks (LLNs), which was proposed by the Internet Engineering Task Force (IETF) working group [2]. The resource-constrained nodes and their extensive inclusion heighten the security challenges in the Internet of Things [3]. This makes RPL prone to several security threats and attacks. An attacker has the ability to modify, insert, rerun, and generate data or control messages, which can significantly disrupt the normal operations of the RPL [4]. The authors in [5] and [6] have identified three categories of RPL attacks: attacks that consume network resources, attacks that affect topology, and attacks related to network traffic.

The routing in RPL is performed by constructing a path towards a single destination called the root node. It is called the Destination-Oriented Directed Acyclic Graph (DODAG). The DODAG is constructed using a set of control messages. DODAG Information Solicitation (DIS) is a control message that is sent by a node when it wants to join the existing DODAG topology. Whenever a node receives a DIS message, it has to reply with a DODAG Information Object (DIO) message to invite the sender to participate in the network. In a DIS attack, the

*Email: arulanita@gmail.com

compromised node unicasts or multicasts the bulk of DIS messages to its neighbors. This often leads to the legitimate nodes resuming the Trickle algorithm [7] and sending numerous DIO messages, thereby consuming more energy and other resources. The unavailability of the legitimate nodes can also result in Denial of Service (DoS) attacks on the IoT network [8].

To overcome the negative impacts of DIS flooding attacks, this paper introduces DISTec, a pioneering detection and mitigation method designed specifically for RPL-based IoT networks. Unlike existing solutions, DISTec uniquely employs a dynamic n x m matrix to monitor and analyze the frequency of DIS message transmissions from each node. This approach allows for precise identification of nodes exhibiting abnormal DIS sending patterns indicative of flooding attacks. DISTec's originality lies in its dual strategy: it not only filters out redundant DIS messages from identified attackers but also implements a systematic process to quarantine and isolate these malicious nodes, effectively preventing them from disrupting network performance. This method addresses critical issues associated with DIS flooding attacks, such as packet loss, increased energy consumption, and control overhead, which are often exacerbated by traditional detection techniques. The DISTec technique was rigorously tested using the Contiki OS.

In this paper, Section 2 expounds the research work related to this research. Section 3 describes the control messages for DODAG construction and the DIS attacks in detail. Section 4 explains the proposed DISTec technique. Section 5 explicates the experimental setup and the results obtained by the proposed model. Finally, Section 6 presents the conclusion and suggests some directions for future work.

2. Related Works

The DIS flooding attack consumes more network resources and leads to severe problems in the network topology. This section highlights some of the important works related to the DIS attack and its impacts.

Ruan et al. [9] designed a self-protecting architecture based on the MAPE-K (Monitor, Analyze, Plan, Execute, and Knowledge) control loop. The authors analyzed the impacts of sinkhole, selective forward, black hole, and flooding attacks on the IoT. According to the user-defined security policies, the nodes react to attacks and protect themselves from them.
Evmorfos et al. [10] proposed a neural network architecture for detecting the SYN flooding attacks in the IoT. Random Neural Networks with Deep Learning and Long-Short-Term Memory (LSTM) neural networks were trained with normal and attack network traffic. The TCP SYN Flood attacks were simulated, and the captured packets were used for training and testing neural networks. According to their findings, the random neural network performed well in terms of detection accuracy and false alarm rate.
Thanigaivelan et al. conducted another study [11], presenting a hybrid anomaly-based intrusion detection system. A novel Distress Propagation Object (DPO) was created in this work to report the anomalous activities of the neighboring nodes to their parent node. This system detected packet flooding attacks, selective forwarding attacks, and clone attacks with a lower false positive rate. For large-scale implementation, it requires additional computational resources.
Nguyen et al. [12] conducted a case study on the flooding attacks in low-power lossy networks. The damages caused by this attack were analyzed with reference to delayed energy consumption and packet delivery ratio. This research proposed an overhearing mechanism, which enabled the detection and elimination of the attacks.
Cong Pu [13] investigated the DIS spam attack, which sent large numbers of DIS messages with different identifiers and led to a DoS attack in an RPL-based environment. The research was implemented in the OMNet++ simulator. The author also evaluated the negative impacts

of the DIS flooding attack on the lossy networks in terms of energy consumption and node lifetime. The attack detection phase was not considered in this work.

Verma et al. [14] conducted an experiment on DIS flooding attacks. They analyzed the memory and energy consumption of these attacks and proposed a lightweight technique, Secure-RPL, to mitigate them. They did not report the impact of implementing the proposed technique in their work.

Qureshi et al. [15] proposed a framework to detect flooding attacks, version number attacks, sinkhole attacks, and black hole attacks in the IIoT environment. In order to identify the attacks, the authors introduced threshold values based on the characteristics and functionalities of the control messages. Detection accuracy, throughput, packet delivery rate, and end-to-delay are the important evaluation metrics in this research.

R. Rajasekar and S. Rajkumar [16] did a simulation-based study of how the DIS flooding attack affected RPL-based 6LoWPAN networks. They looked at key metrics such as power consumption, packet delivery rate (PDR), and end-to-end delay (E2ED) in a number of different situations. It provides valuable insights into IoT network vulnerabilities and offers practical recommendations for mitigating these attacks based on attacker node location and quantity.

3. RPL and DIS Attack

3.1. Control Messages in RPL

RPL is a distance vector routing protocol that maintains the network status by constructing DODAGs. Each DODAG is identified by an RPL Instance ID, a DODAG ID, and a DODAG Version Number. The DODAG consists of a root node and a number of child nodes in a tree-like structure [17]. The Code field of the ICMPv6 protocol is used to identify the four types of control messages, which are the core elements for constructing the DODAG. The control messages of the ICMPv6 [18] are listed below:

- DODAG Information Solicitation (DIS)
- DODAG Information Object (DIO)
- DODAG Advertisement Object (DAO)
- DAO-Acknowledgement (DAO-ACK)

The DODAG root issues the DIO message to its neighbors and initiates the DODAG construction process. The DIO message consists of the routing metrics and constraints, like the root node’s ID, rank, and an objective function. While a node gets a DIO message from its neighbor, it adds the corresponding neighbor to its parent listing, computes its own rank via the use of the objective function, and passes on the DIO message with the new rank. When the DIO message reaches the leaf node, the route towards the root node is built through its parent list. To form end-to-end communication from root to other nodes, the leaf node issues a DAO control message to broadcast reverse route information and record the nodes visited along the upward routes. The root node has to respond with a DAO-ACK to the sender of the DAO message [19]. Therefore, the DODAG is constructed and directed towards the root node from other nodes by sending the DIO, DAO, DAO-ACK, and DIS messages.

3.2. DIS Attacks in RPL

When a new node enters the DODAG, it requests the routing details from its neighbor nodes by sending the DIS message. Upon receiving the new node's request, the neighbors are required to respond using the DIO message. Until the new node receives a DIO message, it repeatedly broadcasts the DIS message [20].

In order to send the route information periodically, the trickle algorithm is set for each node. The duration of initiating the trickle algorithm varies based on the stability of the DODAG. When a node gets a DIS message from its adjacent nodes, it ends the current DIO transmission and reruns the Trickle Algorithm, and the time is set as a minimum [21]. The DIS_DELAY is

the delay to initiate the first DIS message by the new node, and the DIS_INTERVAL is the waiting time for the new node to send the second and so on DIS messages. The default value defined in RPL for DIS_DELAY is 5 seconds, and DIS_INTERVAL is 60 seconds.
Let X= $\{x_1, x_2, x_3, .., x_n\}$ is a set of new nodes, where $|X|$=n. The neighbours of each new nodes are represented by using Y= $\{y_1, y_2, y_3, ..., y_m\}$ where $|Y|$=m. Here, X and Y are distinct sets. Hence, X ∩ Y= Φ. The sets X and Y form n x m matrix.
After a small interval Δt (DIS_DELAY), the new nodes send DIS messages to their neighbors to get back DIO messages. The DIS flooding attack will occur during the time t (DIS_INTERVAL). The DIS attacks take I different values, x_i (I = 1, 2, …, I) in the new nodes, and the probability of the value x_i being taken is $P(x_i)$. The set of numbers $P(x_i)$ is said to be a probability mass function. The variable takes one of the values as it is given in Eq. (1).

$$\sum P\ (X_i) = 1 \quad (1)$$

It will take any value in this $0 \leq P_X\ (x_i) \leq 1$ range for all x_i. The probability mass function can be defined in Eq. (2).

$$P\ (x_i) = m_i\ /\ Z \quad (2)$$

where, Z is the total number of DIS attacks and m_i is the node from which the DIS attack is initiated. To represent the number of attacks, n x m matrix M is constructed whenever the new nodes enter the network. If there is a DIS message sent from a node to its neighbors after the DIS_DELAY and before the completion of the DIS_INTERVAL, the corresponding M_{ij} value will be incremented.

If the nodes are normal nodes, they will not send DIS messages during DIS_INTERVAL time. If a DIS attacker is present, it disregards the DIS_INTERVAL or DIS_DELAY time, instead continuously generating and transmitting DIS messages to its neighboring nodes. Thus, matrix M may have no value other than zero in the normal case. But the attacker scenario has some values in the matrix M. The values in the matrix are used to identify whether there is any flooding of DIS messages in the DODAG. The DIS messages initiated in the attacker scenario are discarded. Figure 1 illustrates the DIS flooding attack concept.

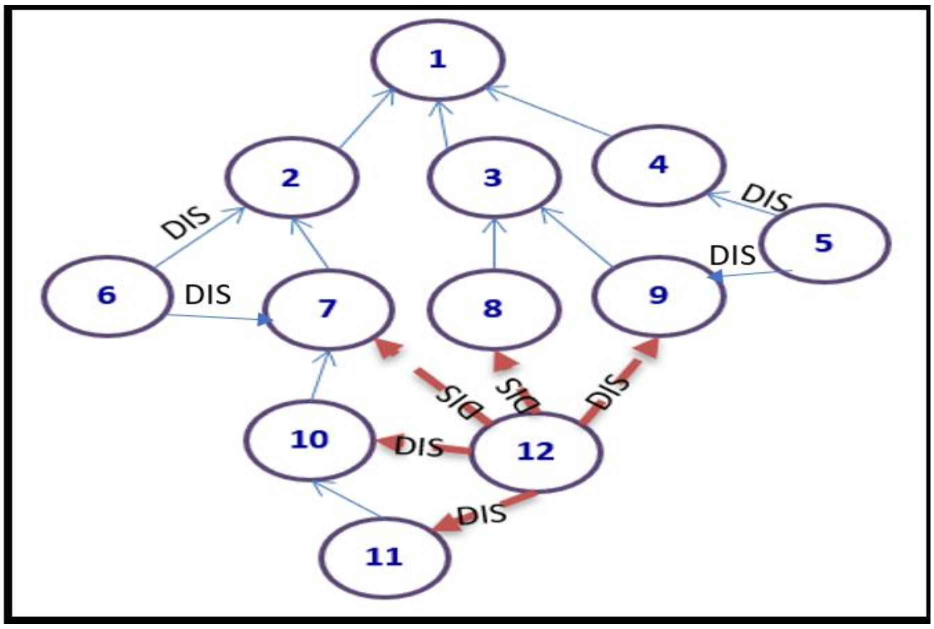


Figure 1: DIS attacker 12 sends multiple DIS messages to its neighbors

As shown in Figure 1, there are 12 nodes in the scenario. Node ‘1’ is the root node, and the nodes ‘5’, '6', and ‘12’ are the new nodes entering the DODAG. Among the three nodes, nodes ‘5’ and ‘6’ are normal nodes, and node ‘12’ is a malicious node that sends multiple DIS messages to its nearby nodes. The node ‘12’ initiates multiple requests to its adjacent nodes ‘7’, ‘8’, ‘9’, '10', and ‘11’ in order to get back the DIO message to join the DODAG. On receiving this request, the adjacent nodes restart the Trickle Timer Algorithm to send the DIO message to the node ‘12’. This unnecessary resetting of the timer increases control traffic and makes the network and communication links unavailable for resource-constrained IoT devices. This malicious node '12' not only increases the control overhead but also consumes the network resources of its neighbors, thereby disrupting the routing process.

4. Proposed DISTEC Methodology

In this section, a technique called DISTec is proposed to address the DIS flooding attacks in RPL-based IoT networks. Keeping in mind the characteristics of the DIS message and the DIS flooding attacker, the DISTec technique has been deployed. This DISTec technique circumvents the unnecessary reset of the trickle timer and the overwhelming generation of DIS messages. The different phases of the DISTec technique to detect the DIS flooding attacks are explained in Figure 2.

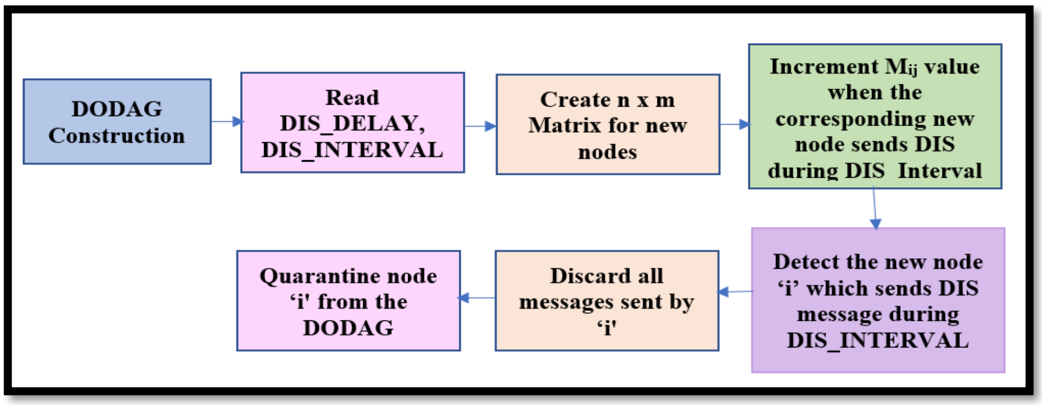

DODAG Construction
Read DIS_DELAY, DIS_INTERVAL
Create n x m Matrix for new nodes
Increment Mij value when the corresponding new node sends DIS during DIS Interval
Detect the new node ‘i’ which sends DIS message during DIS_INTERVAL
Discard all messages sent by ‘i'
Quarantine node ‘i' from the DODAG

Figure 2: Phases of DISTec Technique

As it is given in Figure 2, the DODAG is constructed using the normal procedure. When a new node enters, after the DIS_DELAY, it sends the first DIS message. Meanwhile, an n x m is constructed, where ‘n’ is the number of new nodes and ‘m’ is the corresponding neighbors. If there is a single new node, then the matrix will be like a list or array. The matrix is given in Table 1.

Table 1: n X m Matrix to address the DIS Attack

	1	**2**	**3**	**..**	**m**
1	M_{11}	M_{12}	M_{13}	..	M_{1m}
2	M_{21}	M_{22}	M_{23}	..	M_{2m}
…	..	..	..	..	..
n	M_{n1}	M_{n2}	M_{n3}	..	M_{nm}

In Table 1, the row is denoted by using ‘i’ and the column is represented using the variable ‘j’. Initially, the matrix is initialized with zeros. The matrix will record the number of DIS messages sent from node ‘I’ to node ‘j’ during the DIS_INTERVAL (60 second) time. When a node 'i' sends a DIS message to node 'j', it increments the corresponding value of Mij.
The DIS_DELAY parameter is set to 5 seconds, which matches the default value in the Cooja Simulator. A 5-second delay allows new nodes enough time to stabilize and begin communication without causing unnecessary latency. The DIS_INTERVAL is set to 60 seconds, which is also consistent with the simulator's default value for normal operation. This

ensures that the DISTec technique operates within standard Cooja parameters, allowing for accurate simulation of typical network behavior and attack scenarios.

This interval provides sufficient time for DISTec to capture and analyze message traffic patterns, distinguishing between normal network behavior and potential flooding attacks. The interval length ensures that legitimate nodes are not unnecessarily affected while still allowing for the detection of abnormal behavior indicative of an attack. Since normal nodes do not transmit DIS messages during the DIS_INTERVAL, the values in matrix M are initialized to zero. The M_{ij} value remains unchanged (zero) for normal nodes. For each DIS message detected during the interval, the corresponding node's M_{ij} value is incremented. If any value other than zero is found in the matrix, the corresponding node is identified as an attacker.
By using these standard values, the DISTec method uses simulation parameters that have been used for a long time. This makes sure that the results are reliable and comparable and that attack detection and network performance stay high. The values in the matrix 'M' are analyzed, and the attacker will be detected using the Mij values as given in Eq. (3).

$$M_{ij} = \begin{cases} = 0 & if \;\; 'i' \; is \; a \; normal \; node \\ > 0 & if \;\; 'i' \; is \; anttacker \; node \end{cases} \tag{3}$$

Therefore, if any cell M_{ij} has a value greater than zero, then the particular 'I' node is identified as the attacker node. Any messages generated by node 'I' during the DIS_INTERVAL are discarded, and the node 'i' is declared as an attacker and removed from the DODAG. Subsequently, the DODAG is reconstructed with the remaining nodes. The timer and matrix values are reset after DIS_INTERVAL, and the process continues. This approach is used to develop the DISTec algorithm. The symbols used in the DISTec algorithm are given in Table 1.

Table 2: Symbol and Descriptions used in DISTec

Symbol	Description
D_D	DIS_DELAY; 5 Seconds
D_I	DIS_INTERVAL; 60 Seconds
t	Counter clock (60 to 0)
M	n x m matrix
(Row in M) i	A new node i
(Column in M) j	Neighbours of the new node i

The algorithm used to implement the DISTec technique in this research is explained in Figure 3.

```
Algorithm DISTec
1. Start
2. Read Parameters:
     DIS_DELAY ← 5 seconds
     DIS_INTERVAL ← 60 seconds
3. Initialize:
    $D_D$ ← DIS_DELAY
    $D_I$ ← DIS_INTERVAL
4. Construct the DODAG
Create a Directed Acyclic Graph for routing in the network.
5. For each new node joining the DODAG:
    Wait for $D_D$ (DIS_DELAY).
    Initialize matrix M:
    n ← number of new nodes
    m ← number of corresponding neighbors
    Set M as an n x m matrix where all $M_{ij}$ ← 0.
6. Start the DIS monitoring period:
    Set the counter clock t ← $D_I$ (DIS_INTERVAL).
7. During the DIS monitoring period:
    If node i sends a DIS message:
      For each i from 1 to n:
        For each j from 1 to m:
            Increment $M_{ij}$ by 1 if node i sends a DIS message to node j.
8. Detect Attackers:
    For each node i from 1 to n:
       For each node j from 1 to m:
         If $M_{ij}$ > 0:
            Declare node i as an attacker.
            Discard all DIS messages from node i.
            Isolate node i from the DODAG.
9. Initiate Global Repair:
      Rebuild the DODAG excluding the attacker nodes.
10. Reset Matrix and Timer:
      After analysis, reset matrix M and restart counter clock t.
11. End
```

Figure 3: Algorithm for Detecting the DIS flooding attack

The output of the algorithm is to decide whether there is a DIS flooding attack on the IoT network or not and to disconnect the attacker. If all M_{ij} values of the matrix M contain zero, then there is no attack in the network. Otherwise, there is a DIS attack, and the corresponding i^{th} node is detected as the source of the attack.

4.1. Implementing DIStec Technique

The DIStec technique is applied to detect the DIS flooding attacks in Figure 2. Here, the nodes ‘5’, '6', and ‘12’ are the new nodes. The nodes ‘5’ and ‘6’ are the legitimate nodes, and the node ‘12’ is the attacker. The default DIS_DELAY is 5 seconds, and DIS_INTERVAL is 60 seconds. After the DIS_DELAY time, nodes ‘5’ and ‘6’ send DIS messages to their neighbors to receive a DIO message from them and join the DODAG. After sending the DIS message, the nodes ‘5’ and ‘6’ wait for the DIS_INTERVAL time (60 sec.) to send the next

DIS message. As the node ‘12’ is an attacker, it will not consider the DIS_DELAY or DIS_INTERVAL time. It continuously transmits DIS messages and floods its neighbors. If the node ‘12’ sends one DIS message per second, then the matrix constructed during the DIS_INTERVAL time is given in Table 3.

Table 3: Matrix during DIS_INTERVAL (60 Sec.) for Figure-1

	1	**2**	**3**	**4**	**7**	**8**	**9**	**10**	**11**
5	X	X	X	**0**	X	X	0	X	X
6	X	**0**	X	X	**0**	X	X	X	X
12	X	X	X	X	**60**	**60**	**60**	**60**	**60**

In this matrix, the new nodes are placed in the rows, and the existing nodes in the DODAG are placed in the columns. If the new node in the row doesn’t have a neighbor, then the corresponding cell is marked as ‘x’. Other cells are set as ‘0’ during the initialization. The matrix is analyzed during DIS_INTERVAL time. As the nodes ‘5’ and ‘6’ are the legitimate nodes, they will not send any DIS message during the DIS_INTERVAL, and their corresponding cell value is 0. As the ‘12’ is an attacker and it sends one DIS message per second, the corresponding neighbor cells contain the value 60 after the DIS_ INTERVAL. As it has a value greater than zero, the node ‘12’ is identified as the attacker. Then the matrix is reset, and the node ‘12’ is isolated.

5. Performance Evaluation

5.1. Experimental Setup

This experiment is performed in the Contiki O.S. Cooja Simulator [22] [23]. This experiment uses a random topology to form a 6LoWPAN network with a maximum of 52 nodes. The border router (6BR) acts as the root node. There are 50 normal nodes, one border router, and one malicious node in this scenario. The malicious node frequently sends DIS messages. Table 4 lists the simulation parameters.

Table 4: Simulation Parameters

Normal Nodes	51 (including root)
DIS Attacker	1
Mote type	Tmote Sky
Simulator	Contiki O.S. Cooja Simulator 3.0
Topology	Random
Radio Medium	Unit Disk Graph Medium: Distance Loss
Topology Dimension	200 m X 200 m
Transmission Range	50 m
Inference Range	100 m
Tx Ratio	100%
Rx Ratio	100 %
Simulation Time	30 minutes

Figure 4 provides a screenshot of the simulation with one DIS attacker node.

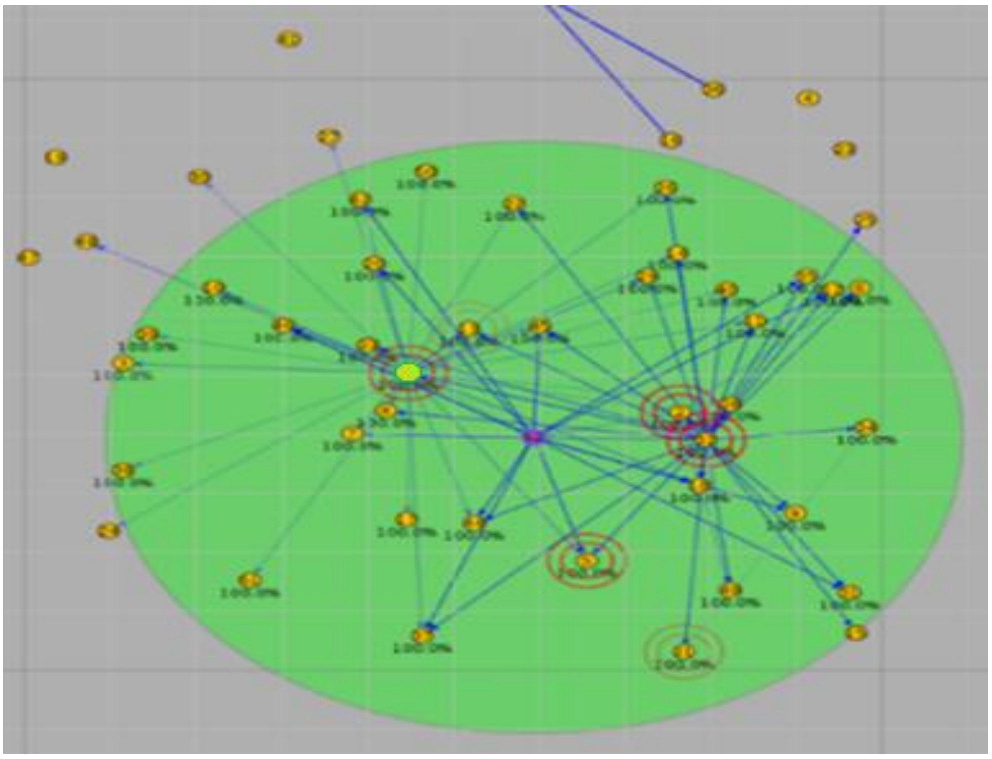
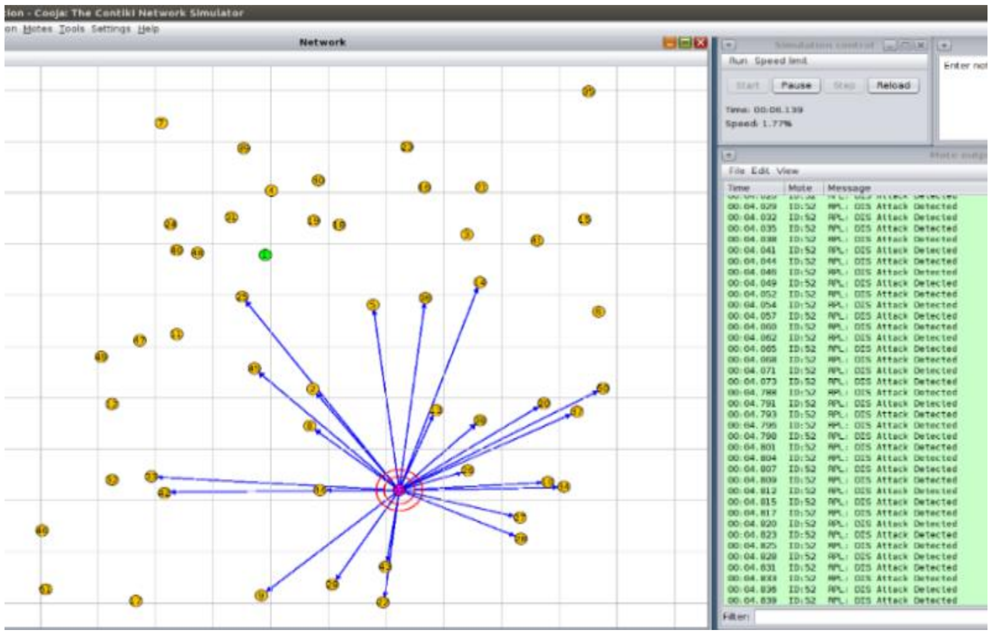


Figure 4: Simulation with DIS Attacker

The node represented by the green color is the root node. The yellow-colored nodes are the normal nodes, and the purple-colored node is the DIS attacker node. The lines between the nodes represent the communication among the nodes.

5.2. DIS attack Detection using DISTec

In the DIS attacker scenario, the proposed DISTec technique was installed in the border router in order to safeguard the IoT networks from DIS flooding attacks. The DISTec technique detects all DIS flooding attacks in RPL-based IoT networks. The screenshot after implementing the DISTec in the root node is given in Figure 5.



```
Time: 00:06.139
Speed: 1.77%

Time        Mote   Message
00:04.029   ID:52  RPL: DIS Attack Detected
00:04.032   ID:52  RPL: DIS Attack Detected
00:04.035   ID:52  RPL: DIS Attack Detected
00:04.038   ID:52  RPL: DIS Attack Detected
00:04.041   ID:52  RPL: DIS Attack Detected
00:04.044   ID:52  RPL: DIS Attack Detected
00:04.046   ID:52  RPL: DIS Attack Detected
00:04.049   ID:52  RPL: DIS Attack Detected
00:04.052   ID:52  RPL: DIS Attack Detected
00:04.054   ID:52  RPL: DIS Attack Detected
00:04.057   ID:52  RPL: DIS Attack Detected
00:04.060   ID:52  RPL: DIS Attack Detected
00:04.062   ID:52  RPL: DIS Attack Detected
00:04.065   ID:52  RPL: DIS Attack Detected
00:04.068   ID:52  RPL: DIS Attack Detected
00:04.071   ID:52  RPL: DIS Attack Detected
00:04.073   ID:52  RPL: DIS Attack Detected
00:04.788   ID:52  RPL: DIS Attack Detected
00:04.791   ID:52  RPL: DIS Attack Detected
00:04.793   ID:52  RPL: DIS Attack Detected
00:04.796   ID:52  RPL: DIS Attack Detected
00:04.798   ID:52  RPL: DIS Attack Detected
00:04.801   ID:52  RPL: DIS Attack Detected
00:04.804   ID:52  RPL: DIS Attack Detected
00:04.807   ID:52  RPL: DIS Attack Detected
00:04.809   ID:52  RPL: DIS Attack Detected
00:04.812   ID:52  RPL: DIS Attack Detected
00:04.815   ID:52  RPL: DIS Attack Detected
00:04.817   ID:52  RPL: DIS Attack Detected
00:04.820   ID:52  RPL: DIS Attack Detected
00:04.823   ID:52  RPL: DIS Attack Detected
00:04.825   ID:52  RPL: DIS Attack Detected
00:04.828   ID:52  RPL: DIS Attack Detected
00:04.831   ID:52  RPL: DIS Attack Detected
00:04.833   ID:52  RPL: DIS Attack Detected
00:04.836   ID:52  RPL: DIS Attack Detected
00:04.839   ID:52  RPL: DIS Attack Detected
```

Figure 5: Screenshot after implementing DISTec

The n x m matrix created by DISTec easily identifies the attacker and the number of attacks that enter the networks from a particular malicious node. The effectiveness of the DISTec technique is assessed using the detection accuracy which measures the proportion of actual DIS flooding attacks correctly identified by the DISTec technique. It is calculated as:

$$\text{Detection Accuracy } (\%) = \left(\frac{\text{Number of Detected Attacks}}{\text{Total Number of Attacks}}\right) \times 100 \quad (4)$$

After implementing the DISTec technique in the attacker scenario, the packets are captured using the 6LowPAN analyzer tool. The number of attacks initiated and the number of attacks detected are listed in Table 5.

Table 5: DIS Attacks detected using DISTec

Simulation Time	30 minutes
No. of Attacks	922
No. of detected Attacks	908
Detection Accuracy in %	98.48 %

The detection accuracy is calculated using Eq. 4, and 98.48% of detection accuracy is obtained. The DISTec approach detects almost all attacks initiated by the DIS flooding attackers.
After the attacker is identified, the attacker details are broadcast by the root node. The DIS messages initiated by the attacker are discarded, and the attacker node is isolated from the DODAG. Then the root node initiates the global repair process in order to safeguard the network.

5.3. Comparative Analysis with Existing Technique

The performance of the DISTec is compared with the hybrid system proposed by [14] based on various aspects and given in Table 6.

Table 6: Comparative Analysis

	DISTec	**Hybrid System [14]**
Detection Accuracy	98.48%	90 -95% detection rates for various attacks, including DIS flooding.
False Positive Rate	Low false positive rate due to targeted analysis of DIS message frequency patterns.	Higher potential for false positives due to the combination of signature and anomaly detection mechanisms.
Methodology	Uses a matrix-based approach to track DIS message frequency and identify attackers. Focused specifically on DIS flooding attacks.	Combines signature-based and anomaly-based detection to cover a wide range of attack types. More complex configuration required.
Scalability and Adaptability	May face scalability challenges in large networks due to matrix size and computational needs.	More adaptable through its hybrid approach, but scalability depends on the efficiency of signature and anomaly detection algorithms.

This table highlights the key differences and similarities between DISTec and the hybrid system, providing a clear comparison across several aspects.

6. Conclusion and Future Direction

This paper proposed the DISTec technique to detect and mitigate DIS flooding attacks in RPL-based IoT environments. The Contiki OS Cooja Simulator successfully implemented the technique, achieving a high detection accuracy of 98.48% with a low false positive rate. By employing a matrix-based approach, DISTec effectively monitors the frequency of DIS messages to identify potential attackers and remove them from the network. While DISTec effectively addresses DIS flooding attacks, further research can explore the detection and mitigation of other RPL-specific attacks, such as rank and version number attacks. Additionally, incorporating AI-driven techniques, such as machine learning or deep learning algorithms, could enhance the adaptability of the system, enabling it to detect evolving attack patterns more effectively. Future work could also focus on improving the scalability of DISTec for larger IoT networks by optimizing the computational requirements of the matrix-based detection mechanism.

References

[1] L. Wallgren, S. Raza, and T. Voigt, "Routing Attacks and Countermeasures in the RPL-Based Internet of Things," *International Journal of Distributed Sensor Networks*, vol. 2013, Article ID 794326, pp. 1–11, 2013. doi: 10.1155/2013/794326.

[2] T. Winter and P. Thubert, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," *RFC Standard 6550*, Mar. 2012.

[3] S. Mangelkar, S. N. Dhage, and A. V. Nimkar, "A comparative study on RPL attacks and security solutions," in *International Conference on Intelligent Computing Control (I2C2-2017),* 2017, pp. 1–6. doi: 10.1109/I2C2.2017.8321851.

[4] A. S. Baghani, S. Rahimpour, and M. Khabbazia, "The DAO Induction Attack Against the RPL-based Internet of Things," *arXiv: 2003.11061v1 [cs.CR]*, 2020.

[5] Mayzaud, R. Badonnel, and I. Chrisment, "A Taxonomy of Attacks in RPL-based Internet of Things," *International Journal Network Security*, vol. 18, no. 3, pp. 459–473, 2016.

[6] T. Le, J. Loo, K. K. Chai, and M. Aiash, "A Specification-based IDS for Detecting Attacks on RPL-based Network Topology," *Information*, vol. 7, no. 2, pp. 25, 2016.

[7] P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko, "Trickle Algorithm," *RFC 6206*, Internet Eng. Task Force, Mar. 2011.

[8] C. Pu, "Sybil Attack in RPL-Based Internet of Things: Analysis and Defenses," *IEEE Internet of Things Journal,* vol. 7, no. 6, pp. 4937–4949, 2020. doi: 10.1109/JIOT.2020.2971463.

[9] R. de A. C. Mello, A. de R. L. Ribeiro, F. M. de Almeida, and E. D. Moreno, "Mitigating Attacks in the Internet of Things with a Self-protecting Architecture," in *The Thirteenth Advanced International Conference on Telecommunication- IARIA 2017*, 2017, ISBN: 978-1-61208-562.

[10] S. Evmorfos, G. Vlachodimitropoulos, N. Bakalos, and E. Gelenbe, "Neural Network Architectures for the detection of SYN flood attacks in IoT systems," in *PETRA '20: The 13th PErvasive Technologies Related to Assistive Environments Conference*, June 2020. doi: 10.1145/3389189.3398000.

[11] N. K. Thanigaivelan, E. Nigussie, S. Virtanen, and J. Isoaho, "Hybrid Internal Anomaly Detection System for IoT: Reactive Nodes with Cross-Layer Operation," *Security and Communication Networks,* vol. 2018, Article ID. 3672698, 2018. doi: 10.1155/2018/3672698.

[12] T. Nguyen, T. Ngo, T. Nguyen, D. Tran, H. A. Tran, and T. Bui, "The Flooding Attack in Low Power and Lossy Networks: A Case Study," in *International Conference Smart Communication Network Technology (SaCoNeT)*, 2018. doi: 10.1109/saconet.2018.8585451.

[13] C. Pu, "Spam DIS Attack against Routing Protocol in the Internet of Things," in *International Conference on Computing, Networking and Communication (ICNC),* IEEE, 2019. doi: 10.1109/iccnc.2019.8685628.

[14] A. Verma and V. Ranga, "Mitigation of DIS flooding attacks in RPL-based 6LoWPAN networks," *Transactions on Emerging Telecommunications Technology*, Wiley, 2019. doi: 10.1002/ett.3802.

[15] K. N. Qureshi, S. S. Rana, A. Ahmed, and G. Jeon, "A Novel and Secure Attacks Detection Framework for Smart Cities Industrial Internet of Things," *Sustainable Cities and Society*, 2020. doi: 10.1016/j.scs.2020.102343.

[16] V. R. Rajasekar and S. Rajkumar, "A Study on Impact of DIS Flooding Attack on RPL-based 6LowPAN Network," *Microprocessors and Microsystems*, vol. 94, p. 104675, 2022. doi: 10.1016/j.micpro.2022.104675.

[17] H. Tian, Z. Qian, X. Wang, and X. Liang, "QoI-Aware DODAG Construction in RPL-Based Event Detection Wireless Sensor Networks," *Journal of Sensors*, vol. 2017, Article ID 1603713, 2017. doi: 10.1155/2017/1603713.

[18] A. Badach, "RPL messages and their Structure," in *Internet of Things- Technologies, Protocols and Applications*, Aug. 2018. Available: https://www.researchgate.net/publication/326960497_RPL_messages_and_their_structure

[19] J. V. V. Sobral, J. J. P. C. Rodrigues, R. A. L. Rabêlo, J. Al-Muhtadi, and V. Korotaev, "Routing Protocols for Low Power and Lossy Networks in Internet of Things Applications," *Sensors*, vol. 19, p. 2144, 2019. doi: 10.3390/s19092144.

[20] D. Sourailidis, R. Koutsiamanis, G. Papadopoulos, D. Barthel, and N. Montavont, "RFC 6550: On Minimizing the Control Plane Traffic of RPL-based Industrial Networks," in *2020 IEEE 21st International Symposium on A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, Aug. 2020, Cork, Ireland, pp. 439-444. doi:10.1109/WoWMoM49955.2020.00080.

[21] B. Farzaneh, M. A. Montazeri, and S. Jamali, "An Anomaly-Based IDS for Detecting Attacks in RPL-Based Internet of Things," in 5th *International Conference on Web Research (ICWR),* 2019. doi: 10.1109/icwr.2019.8765272.

[22] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki—A Lightweight and Flexible Operating System for Tiny Networked Sensors," in *Proceedings of the 1st IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pp. 655–662, 2004. doi:10.1109/MASS.2004.1389206.

[23] A. Dunkels, and B. Gronvall, "Cooja: A Network Simulator for Contiki," in *Proceedings of the 1st IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pp. 662–670, 2004. doi:10.1109/MASS.2004.1389207.