Data traffic interruption detection and mitigation framework for RPL-based WSNs

Daniel Francis Soriano[1],**Wilson Vicente Ruggiero**[1]

[1]Escola Politécnica – Universidade de São Paulo (USP) – São Paulo, SP - Brazil

`daniel.soriano@usp.br/daniel.francis.soriano@gmail.com, wilson@larc.usp.br`

Abstract. *Wireless Sensor Networks (WSN) monitor various aspects of the environment in which they are deployed. Sensor nodes collect and send environment data by neighboring nodes until it reaches the gateway node (root node). WSNs are susceptible to several attacks that totally or partially interrupt the data flow, such as sinkhole, blackhole, gray hole/selective forwarding. We here present an application layer framework for RPL-based WSNs that detects and mitigates the attacks that interrupt data traffic while informing the root node (and the network maintainers) of the suspect nodes by alternative routes, thus preventing interception by attackers. The attacker nodes are then relegated to leaf-nodes by the local reaction of legitimate nodes, as they are prevented from acting as routers, effectively mitigating attacks. By concentrating monitoring functions on the sink, network managers can identify compromised nodes, which can lead to their physical removal.*
The experiments show that our framework kept the package loss rate at about 2% or lower in most of those tested. Besides managing to maintain a low control message overhead (i.e., 4.67% in a no attack scenario and 9.74% under attack) it was able to provide the network managers with information about the location of the attackers.

1. Introduction

The commercial use of IoT solutions is growing daily, and is commonly used for monitoring the health and performance of industrial systems [Bal 2014], or environmental characteristics of large agricultural and forested areas [Prathibha et al. 2017]. Wireless sensor networks (WSN) can be implemented in these scenarios. In this type of network, the sensing nodes also assume the role of routers, forwarding packets from their neighbors to the destination, which is generally a base station (sink).

The cost of these sensor devices is affected by many factors, but low cost is generally achieved by limiting their processing and memory capacity. That restricts the ability to process complex calculations, such as asymmetric encryption algorithms that are much more compute intensive than symmetric encryption [Kumaran et al. 2016][Oludele et al. 2018]. Thus, in a market that seeks increasingly lower production costs, the choice is eventually the adoption of unsafe solutions, particularly when the perception of security of environmental data is not that of confidential information.

Traditional routing protocols are not appropriate for wireless sensor networks; in this scenario, RPL (Routing Protocol for Low-Power and Lossy Networks) has emerged as the de facto standard for routing in WSN [Iova et al. 2015].

1.1. Motivation

Insecure wireless sensor network solutions tend to be cheaper than more secure solutions, and this cost-benefit ratio can induce the purchase of a larger quantity of insecure sensors for the same price as secure solutions with fewer sensors. This can force small companies and developing countries to choose insecure solutions.

The perceived value of secure solutions is further reduced by the notion that environmental monitoring data, for example, does not require confidentiality. Therefore, they can be susceptible to existing attacks, and many of them may cause interruption to data collection. For companies, using WSN is a means of monitoring the production operation, and the lack of this information can stop production lines and increase costs [Bal 2014].

This work presents a method for identifying and reacting to attacks that disrupt traffic from nodes to the sink, here applied to the context of selective forwarding attacks. The proposed method combines the individual reaction of nodes with centralized monitoring by the sink. These two approaches make the network react autonomously and immediately to attacks while also allowing network managers to investigate suspicious nodes and remove them, if possible.

2. RPL protocol

RPL is a distance vector-based routing protocol for LLN networks (Low Power and Lossy Networks) that supports IPv6. Created by the ROLL working group (Routing Over Low-power and Lossy networks) RPL aims to achieve reliable data transmission in networks with variable losses. It is responsible for forwarding packets and deciding which route to use [Parasuram et al. 2016].

The RPL protocol works by building a tree in which the root is the gateway and traffic flows predominantly from the leaves to the root. This tree structure, called DODAG (Destination Oriented Direct Acyclic Graph), is formed when each node selects a neighbor to be its parent, a process that occurs according to a determined metric (lowest rank among the neighbors), which is typically the distance in hops to the base. Once the routes to the base have been defined, they are not changed, except when a node identifies a neighbor with a better rank than the current parent [Winter et al. 2012].

2.1. RPL Security Modes

The RPL protocol specification in its current form, as described in RFC6550, considers three security modes: “unsecured”, “preinstalled” (pre-installed keys) and “authenticated”.

The ”unsecured” mode considers that RPL control messages are sent without additional security mechanisms (independently of possible security mechanisms in use by the link layer) [Winter et al. 2012].

The ”preinstalled” mode requires nodes to have a preinstalled shared key (e.g., the same key for all nodes and root), which is used to add a MAC, or signature, to RPL control messages, that can optionally also be encrypted for confidentiality [Winter et al. 2012].

The ”authenticated” mode also requires a pre-installed key, but routing nodes, besides a new key from an authenticating authority. The ”authenticated” mode was included in the specification only as a future possibility since, according to the RFC, the authenticated mode cannot be implemented with symmetric keys and the current RPL specification only supports symmetric encryption. Also, ”preinstalled” and ”authenticated” modes add protection against replay attacks by using Consistency Check Messages [Winter et al. 2012].

2.2. Limitations of RPL

It is natural for RPL security modes to focus on the routing dimension; consequently, its protection only affects control messages. Therefore, the main function of these networks, data collection, requires additional protections. Selective Forwarding and Packet Drop attacks, which act only during the transmission of non-control messages, are not detected or mitigated by the RPL security modes.

Another characteristic of the RPL security modes provided is the use of a single key shared among all nodes. This means that the compromise of a single node, which has the shared key extracted, makes the entire network susceptible to attacks.

With the key extracted from one of the nodes, the attacker is able to generate valid control messages, which basically opens the network for the attacker. RPL networks do not usually allow network managers to identify an attack situation, thus removing any time restrictions for invasion attempts, and eliminating the possibility of reaction by network managers.

2.3. Contribution

This work presents an application layer framework for RPL based wireless sensor networks, with the ability to identify and repel attacks induced both externally and internally, which cause loss of data messages (measurements made by the sensors). The proposed solution presents superior results to the approaches currently in the technical and scientific literature in this area in terms of packet delivery rate.

Selective Forwarding and Packet Dropping attacks are able to evade the native security protections of the RPL protocol by maintaining normal control message traffic during attacks. By monitoring the receipt of useful messages, nodes are able to detect these attacks and initiate mitigation measures. The proposed solution aims to align the focus of network protection with its main function, which is data collection.

The RPL protocol, when in its preinstalled security mode, makes use of a symmetric key shared between all nodes, which leaves the entire network vulnerable if the key is extracted from any of the nodes. The proposed solution assumes that any node can be compromised by an attack and, in this way, makes the nodes trust only the root node. This feature leads to the use of symmetric keys shared only between each node and the sink, effectively eliminating the vulnerability by removing the possibility of a compromised node affecting the security of the entire network. The solution presented in this work was designed to work independently of RPL security modes.

The proposed solution also adds mechanisms for consolidating attack information and connectivity tests. These additions allow the sink to infer possible suspects from the information collected, thus giving visibility of attacks to network managers.

This work also innovates by proposing the stratification of results by the number of affected nodes, which enables direct comparison between results from multiple experiments (with different simulation seeds), given the dynamic characteristic of DODAG construction.

3. Problem definition

The network layer of sensor networks is vulnerable to different types of attacks, such as: Spoofed rounting; Selective Forwarding/PacketDrop/Gray Hole; Sybil Attack; Sinkhole/Blackhole; Hello Flood; Wormhole; Acknowledge spoofing; and Sniffing attack [Chowdhury and Kader 2013].

Among the attacks above, many have the effect of partially or completely interrupting data traffic, such as: Selective forwarding/Packet Drop; Sinkhole/Blackhole; Acknowledgment spoofing and HELLO flooding. Other attacks do not necessarily purpose to interrupt traffic, but may cause this indirectly, such as Spoofed Routing; Sybil Attack and Wormhole. Finally, there are attacks aimed solely at collecting data, such as Sniffing attack.

Traffic disruption attacks can be externally or internally induced. Externally induced attacks are carried out using a jammer, equipment that interferes with the transmission range used by networks to the point of preventing communication between nodes [Chen et al. 2018]. Internally induced attacks are carried out by compromising one of the network nodes or by introducing an external node that pretends to be a legitimate one.

The low computing capacity of wireless sensor networks poses a delicate balance between the networks self-protection capacity and their longevity, limiting the use of complex cryptographic solutions, therefore making it very difficult to guarantee confidentiality and integrity.

4. Literature Analysis

The literature search sought to find works that aimed to detect traffic interruption attacks in wireless sensor networks with RPL. The authors' analysis of the revised material is shown below.

As pointed out by [Maidamwar and Chavhan 2018], there are situations in which human intervention is necessary, a need not addressed by most works. The works described in [Singh et al. 2016], [Heurtefeux et al. 2015], [Chen et al. 2018], [Stephen and Arockiam 2018], [Airehrour et al. 2017], [Sudhakar and Abhinaya 2019] and [Alansari et al. 2023] do not provide mechanisms to inform network managers of alleged attacks. RPLAD3 by [Alansari et al. 2023] does not mitigate externally induced (jammer) attacks, either.

Several studies—such as those by [Violettas et al. 2021], [Chen et al. 2018], and [Airehrour et al. 2017]—show objectives that align with ours. In addition to detecting attacks, they also recover the network and mitigate future attacks. Another similarity of these works is the potential for communicating attacks to network managers which, despite not declared as an objective, appears to have potential to be added later.

[Amish and Vaghela 2016] broadcast a message informing the identification of suspicious nodes, information that could be used to monitor attacks if they were processed in the sink, but this is not described by the authors. When control messages, intended to protect the network, are not protected by some type of encryption or signature, the security mechanisms themselves become vulnerable to control packet forgery. This means that security mechanisms can be used to mark legitimate nodes as attackers. This type of characteristic is found in [Violettas et al. 2021], and also in other works, such as: [Muzammal et al. 2022], [Patel et al. 2023], [K et al. 2022],

Among the works analyzed, the use of techniques such as: using alternative routes [Heurtefeux et al. 2015], analysis of behavioral changes [Stephen and Arockiam 2018] [Amish and Vaghela 2016] is common. Most of the works analyzed use individual node reactivity to detect and/or mitigate attacks [Singh et al. 2016], [Chen et al. 2018], [Stephen and Arockiam 2018], [Airehrour et al. 2017], [Sudhakar and Abhinaya 2019]; however, few works ([Violettas et al. 2021]) combine the individual reactivity of nodes with the centralized processing of suspects for monitoring purposes.

[Chen et al. 2018] applies to only a specific type of externally induced attack, using a jammer, not reacting to internally induced attacks, after a valid node is compromised, for example. The work presented here aims to overcome this limitation by detecting attacks induced both externally and internally

[Sharma et al. 2022] checks the absence of communication with the parent as a result of traffic interruption, a similar characteristic to our work, but their method of verifying suspects also requires neighboring nodes not to be attackers.

Many works use eavesdropping, such as [Airehrour et al. 2017] and [Alansari et al. 2023]. This technique requires operating the radio for an additional period of time to confirm the forwarding of packets by their parents. Furthermore, this mechanism can only identify traffic interruption if it occurs at the immediately superior node (parent), and not at other nodes on the route.

Another difference between this work and most of those found in the literature is resilience in the event of attacks that attempt to exploit the characteristics of the proposed framework itself. This occurs due to the use of HMAC with symmetric keys and the assumption that the network is naturally insecure; therefore, the nodes trust only the sink, whose communication can be validated (integrity and authenticity) by cryptography.

5. Proposed solution

The proposed solution is an application layer framework for RPL networks called DMAIT (acronym for Detection and Mitigation of Traffic Interruption Attacks in Portuguese), and it suggests a way to enable nodes to autonomously identify a suspected attack situation, and to react independently, preventing attacking nodes from becoming parents, while informing the sink about the attack.

To complement RPL security solutions, that focus on the routing layer (control messages), the proposed solution focuses on data collection and, therefore, attack detection is based on the recurrence of failure to receive the ACK of data packets. The recurring absence of ACK indicates that there is an interruption in communication with the root, and then DMAIT initiates its' reaction.

To ensure that the sink receives the data without modifications, we use symmetric keys; each node has its individual key that is shared only with the sink. The keys are used to generate HMACs (Hash-based Message Authentication Code) of data packets and their

ACKs, besides some new control messages, which adds the ability to verify integrity and authenticity, since only the sink has the keys to all nodes.

The use of HMACs practically eliminates the possibility of modifying data and forging ACK packets, as this would require obtaining the symmetric key of the target node. If the key of one of the nodes is obtained, its use is limited to just one node, since the other nodes trust only the sink, and they have individual keys for this communication.

5.1. Key management

The keys must be installed on the nodes prior to network deployment, and can be installed on the sink when the network is activated, thus allowing new nodes to be added to the network later, simply by installing their keys in the sink. We understand that this mechanism brings two advantages:

1. It avoids complex mechanisms for generating/derivating keys, which can consume battery, in addition to the transmission of keys (or part of it) over the network, which can be intercepted and exploited externally by more powerfull machines.
2. It does not generate complexity in long-term network maintenance, since each damaged/compromised node can have its key removed from the sink, and each new node only needs to have its key inoculated in it and into the sink for it to be added to the network.

To enable simulation experimentation, instead of generating dozens of different binaries as node firmware, each with its inoculated key, the PoC implementation dynamically calculates the node keys based on their identifier (node id), so that the sink can also generate the keys based on the node id received in the control messages, thus emulating the proposed mechanism.

5.2. Connecting to the network and route validation

As soon as nodes connect to the network, they need to know all the nodes on their ascending route (route to the root), because if ACKs are not received, the affected node assumes that the route has been compromised and needs to report its members as suspects to the sink.

Without a route validation mechanism, nodes would not be able to name suspects, or would be susceptible to being tricked by their neighbors (which could spoof their identity), and could eventually report an invalid suspect list. Route validation assures nodes that the sink has verified the key of all nodes and the order in which they appear on the route. The steps in this process are shown below:

1. As soon as a node connects to the network, it sends a route validation message to its parent containing a block with its identity followed by that of its immediate parent, in addition to an HMAC of the block (Figure 1-A).
2. The next node (parent) receives the message and then composes a new message bringing all the blocks of the received message, plus a new block containing the following information: identity of the previous node, its own identifier and the identity of its immediate parent, followed by an HMAC of its block, generated from its key (Figure 1-A).

3. The following nodes repeat step 2 which, at the end of the process, generates a message containing all blocks, with the HMACs generated by each node with their respective keys (Figure 1-A).
4. By having the keys of all nodes in the network, the sink is able to verify the received message and all its blocks. The sink is then able to validate the HMAC of all blocks and to reconstruct the route taken by the message and to validate the identity of all nodes, in addition to checking the continuity of the route (Figure 1-B).
5. After validating the route and the identity of all the nodes composing it, the sink generates a response with the identity of all the nodes on the route, adding an HMAC with the key of the original node. The message is then sent along the same route that has been validated, but in the opposite direction (Figure 1-B).
6. The original node receives the message and checks the HMAC confirming the identity of the sink and the list of nodes on the original route. The node then stores it's, now validated, route to root, and from this point on the node is sure of all the nodes of its route to the sink (Figure 1-B).

If the sink does not validate the route no response is sent, which makes the route be invalidated by the node. When marking the route as invalid, the node places its immediate parent on the negative list (no longer considering it as a viable parent) and disconnects it from the network. The node then tries again to connect to the network, but now using another node as a parent and repeats the route validation.

Route validation generates additional message traffic on the network, and at first it may seem excessive. This feature is partially caused by the decision to create a framework on top of RPL rather than reusing RPL's own control messages, but we understand that the benefits outweigh the costs. In our tests we noticed that the network converged quickly, and after that the traffic of these messages ceased. This limits the overhead to only the initial moments of the network, and in long-lasting networks this overhead tends to be negligible.

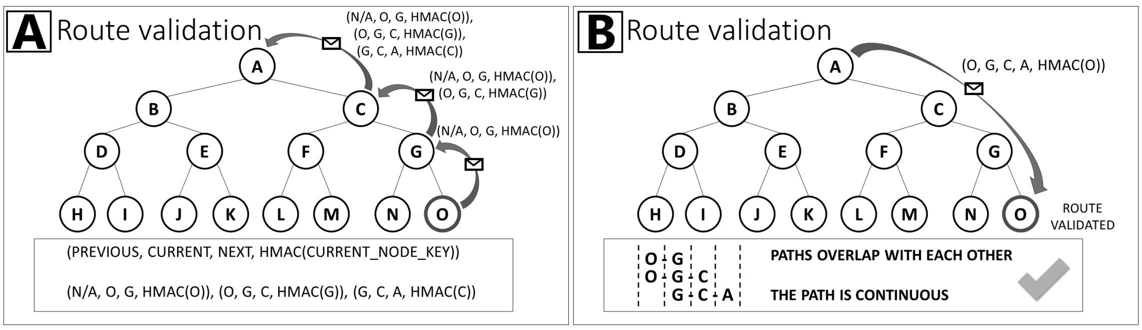
A Route validation
(N/A, O, G, HMAC(O)), (O, G, C, HMAC(G)), (G, C, A, HMAC(C))
(N/A, O, G, HMAC(O)), (O, G, C, HMAC(G))
(N/A, O, G, HMAC(O))
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
(PREVIOUS, CURRENT, NEXT, HMAC(CURRENT_NODE_KEY))
(N/A, O, G, HMAC(O)), (O, G, C, HMAC(G)), (G, C, A, HMAC(C))
B Route validation
(O, G, C, A, HMAC(O))
ROUTE VALIDATED
O G
O G C
G C A
PATHS OVERLAP WITH EACH OTHER
THE PATH IS CONTINUOUS

Figure 1. Route Validation: Chain of blocks - Each node adds a block with the three pieces of information (previous, current, next) plus the HMAC with its key (A). Continuity check - With the HMACs and route continuity validated, the sink sends the complete route plus the HMAC generated with the original node key to the original node (B).

The DAO and DAO-ACK messages, native to the RPL, could not be used by the route validation process as, even in the "preinstalled" mode of RPL which uses cryptographic keys, there is no authenticity of the origin of the messages, since the cryptographic key is shared by all nodes. The "authenticated" mode could be an option for validating routes when it comes into existence; however the DAO-ACK message would need to be modified to carry the entire route, authenticated by the sink.

5.3. Data transmission and attack detection

After route validation, the node can start transmitting data to the sink. The transmission of data packets between the nodes and the sink is supplemented with an HMAC code, which provides the sink with the ability to verify the integrity and authenticity of the message sent by each node, and vice versa. ACKs also have their own HMAC, but they also include the HMAC of the original data packet, so that the node can verify whether the data packet arrived at the sink intact, and that the ACK has not been altered. The node therefore only needs to store the HMACs of the latest packets sent for integrity checking when receiving ACKs.

Attack detection occurs when the node stops receiving ACKs from the sink, exceeding the predefined time limits and amount of loss, or when the HMACs of the ACK are not validated. Any change to the packets by an attacking node would invalidate the HMACs. This mechanism directly addresses Acknowledgement spoofing attacks.

When an attack is detected, the node initiates the reaction, which comprises two distinct actions: local reaction and attack communication.

In the local reaction, suspicious nodes are added to a negative list in each affected node, so that they are prevented from becoming parents of the affected nodes. Over time, this mechanism prevents suspects from positioning themselves as routers and from disrupting traffic to affected nodes, while also allowing suspects to send messages through affected nodes. This heuristic causes attacking nodes to be relegated to the leaves of the tree, where they can no longer interrupt transmissions.

For communicating the attack, the node tries to establish an alternative route to the sink, avoiding nodes on its traditional route, now considered suspicious. The purpose of the alternative route is to inform the sink of all suspicious nodes, while reducing the possibility of interception by one of the compromised nodes.

5.4. Testing and consolidation of suspects

Once a suspected route is received, the sink then begins testing suspicious nodes by sending a message to the affected node (which notified the attack), going through the suspicious route. The message sent by the sink requires an individual response with an HMAC from each member of the suspect route. The node responsible for the interruption (and its descendants) is expected not to be able to respond, which determines the location of the network failure point and increases the quality of detection.

Having a privileged position in the network, the sink can combine suspects sent from different nodes into a single list that considers the frequency of suspicions of each node. This list provides managers with an overview of the network and a more accurate identification of nodes compromised by the attack. In the current implementation of DMAIT, the sink program runs inside a node, which does not allow using advanced visualization and analysis tools. However, the information displayed in the console by the sink in the current version is sufficient to generate heatmaps.

6. Experimental setup

To show the effectiveness of the solution, the DMAIT framework was implemented as an application on the Contiki-NG operating system with its native RPL implementation. The experiments were built in the COOJA simulator.

The application was designed to simulate an environmental monitoring network, with data sent by nodes every 1 minute (with random jitter of up to plus or minus 1 second), and was built on Contiki-NG release v4.5-209.

Two applications with these characteristics were built, one with the complete implementation of the algorithm, called DMAIT, and the other as a control called BASELINE, which only sent messages using the RPL, with or without ACKs depending on the scenario to be tested. In the BASELINE application, ACKs, when used, were only intended to generate traffic and were not processed or validated by the nodes. In both scenarios (with or without ACKs), the BASELINE application did not have any of the other implementations of the DMAIT algorithm.

Selective Forwarding (also called Gray Hole and Packet Drop) was selected to represent the traffic interruption attack, as this type of attack can be made invisible to the RPL; therefore, its detection and mitigation are completely dependent on the mechanisms added by our method. The attack was implemented using the framework by F. Algahtani, T. Tryfonas and G. Oikonomou [Algahtani et al. 2021], which provides a reference implementation for various attacks on RPL networks, compatible with the script automation of the COOJA simulator. The semantics of the Selective Forwarding attack is the same as defined by the authors, in which, when the attack is initiated, all packets are discarded by the node except ICMPv6 packets with code 155, which is the code used by RPL packets, thus keeping them invisible to this attack.

Both applications were then subjected to experiments to qualitatively measure the reaction (recovery) to attacks and the ability to inform network maintainers about the attack. The results of experiments with the DMAIT framework were compared with the control application (BASELINE). We also compared the DMAIT framework with other methods for detecting these attacks found in the literature.

6.1. Methodology

To evaluate the proposed method, three scenarios were designed, each with 16 nodes in total, one of them being the sink (node 1), and 2 of them the attackers (nodes 2 and 8). The differences between the scenarios are due to the positioning of the nodes and the distance (transmission radius) between them. All the scenarios are presented in Figure 2.

5TREE Scenario: Configured as 5 different trees, 3 levels deep each, with the same sink, organized in parallel and positioned in a circular fashion (the first tree is next to the last). The intermediate level is within reach of only its children and the sink, and is not within the communication radius of its neighbors at the same level. Each of the leaves has a neighbor node that belongs to another tree and, therefore, a different path to the sink. This configuration was designed with 2 objectives: to allow the existence of alternative routes to the sink and DODAG stability among multiple simulations (to generate comparable results from different simulation seeds).

SPARSE GRID Scenario: Configured as a 4 x 4 grid, where each node is located within the radius limit of the 4 nodes around it (above, below, left and right), if any. In this scenario, there is a large supply of neighbors and multiple possible routes.

GRID DENSE Scenario: Configured as a 4 x 4 grid, similar to the GRID SPARSE scenario, but with the nodes closer together. Each node can reach two nodes in each horizontal and vertical directions, and one additional node in each of the diagonals. This scenario further expands the number of possible routes compared to the previous one.

To allow comparison between the different executions in the GRID SPARSE and GRID DENSE scenarios, the results of 15 experiments were stratified by the number of affected nodes (identified by the DODAG generated before the attacks began), while in the 5TREE scenario, the average of the results of 10 executions was used.

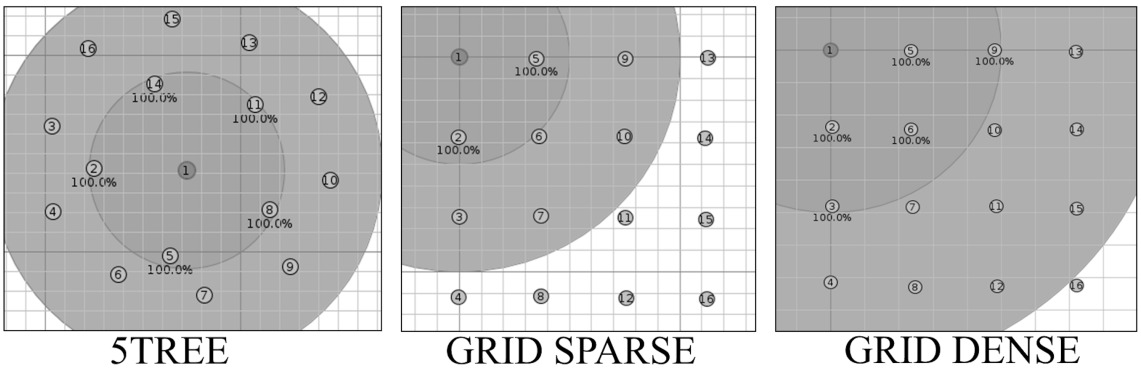

Figure 2. Experiment Scenarios - 5TREE Scenario (left), GRID SPARSE Scenario (middle) and GRID DENSE Scenario (right)

7. Results and discussion

7.1. Overhead analysis

The experiment considered situations with and without attack, and the BASELINE application was measured with and without ACKs also, to evaluate the individual cost of using ACKs and DMAIT control messages. The attack scenarios considered the 2 attackers (nodes 2 and 8) with a total of 4 nodes affected, and the values obtained are presented in Table 1.

Control overhead is the ratio of control messages to the received data packets [Surendar and Umamakeswari 2016]; therefore, the number of control messages added by the DMAIT method was, on average, 4.89% in the attack scenario and 2.34% in the non-attack scenario. The overhead values are shown in chart A of Figure 3.

7.2. Delay analysis

Chart B of Figure 3 shows the message delay, disregarding the ACK transmission and reception time, and it is thus possible to include the 5TREE BASELINE - NO ACK scenario in the comparison. The maximum delay when using DMAIT increased by 25.43% when compared to the 5TREE BASELINE - NO ACK scenario, and 9.15% compared to the BASELINE scenario with ACK. The average delay for the 5TREE scenario increased 24% (from 39.86ms to 49.42ms) when using DMAIT, an acceptable amount considering

Table 1. Average number of messages per scenario

SCENARIO	DATA MESSAGES	DATA ACKS	CONTROL MESSAGES	AFFECTED NODES	PACKET LOSS
5TREE BASELINE	892.7	672.6	0	4	24.66%
5TREE BASELINE - NO ACK	892.6	0	0	4	24.64%
5TREE DMAIT	847.3	839.3	82.5	4	0.94%
5TREE BASELINE - NO ATTACK	893.1	893.1	0	0	0.00%
5TREE BASELINE - NO ACKS & NO ATTACK	892.7	0	0	0	0.00%
5TREE DMAIT - NO ATTACK	856.3	856.3	40	0	0.00%

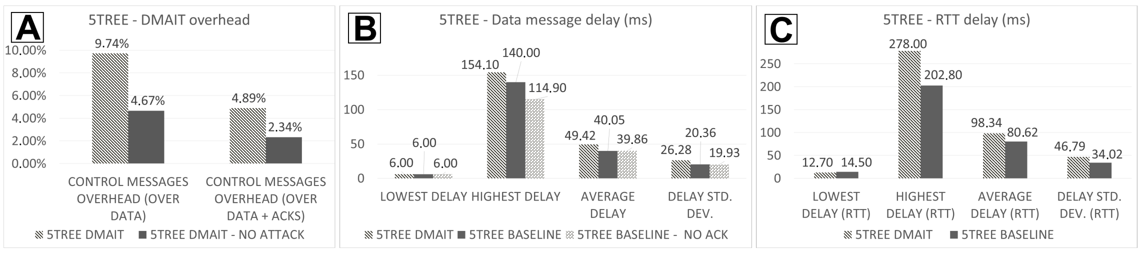

Figure 3. 5TREE scenario data - Control message overhead (A). Delays (until the data message is received) in the 5TREE scenario (B). Comparison of RTT delays (until ACK is received) in the 5TREE scenario (C).

the drop of more than 26 times (from 24.66% to 0.94%) in packet loss rate when using DMAIT.

Among the 5TREE scenarios with ACK, there is an increase in packet delivery delays of around 22% in the average RTT delay (Round Trip Time - from sending the data message to receiving its ACK) and up to 38% in the highest delay, both shown in chart C of Figure 3.

In both GRID-SPARSE and GRID-DENSE scenarios, the packet delivery delay was practically constant with DMAIT, while in BASELINE it was reduced as the number of affected nodes increased. This was noticed in both data message delivery and RTT. With DMAIT, the constancy of values shows the reaction to the attack and the maintenance of packet delivery rates. This behavior can be seen in all charts of Figure 4.

7.3. Packet loss analysis

In the 5TREE scenario, packet loss in the BASELINE scenario was 24.66%, while in the DMAIT scenario, packet loss was reduced to 0.94%, as observed in Table 1.

Packet loss is proportional to the number of affected nodes, and the GRID-SPARSE and GRID-DENSE scenarios are not as stable as the 5TREE scenario; therefore, at each experiment, a different seed generated a new DODAG, changing the number of affected nodes. For this reason, the results were grouped by the number of affected nodes.

The loss rate curve per number of affected nodes in the GRID-SPARSE and GRID-DENSE scenarios are depicted in Figure 5. In both scenarios, packet loss was reduced by the DMAIT algorithm to around 2% and 4% in all ranges of affected nodes, with a downward trend as the number of affected nodes increases.

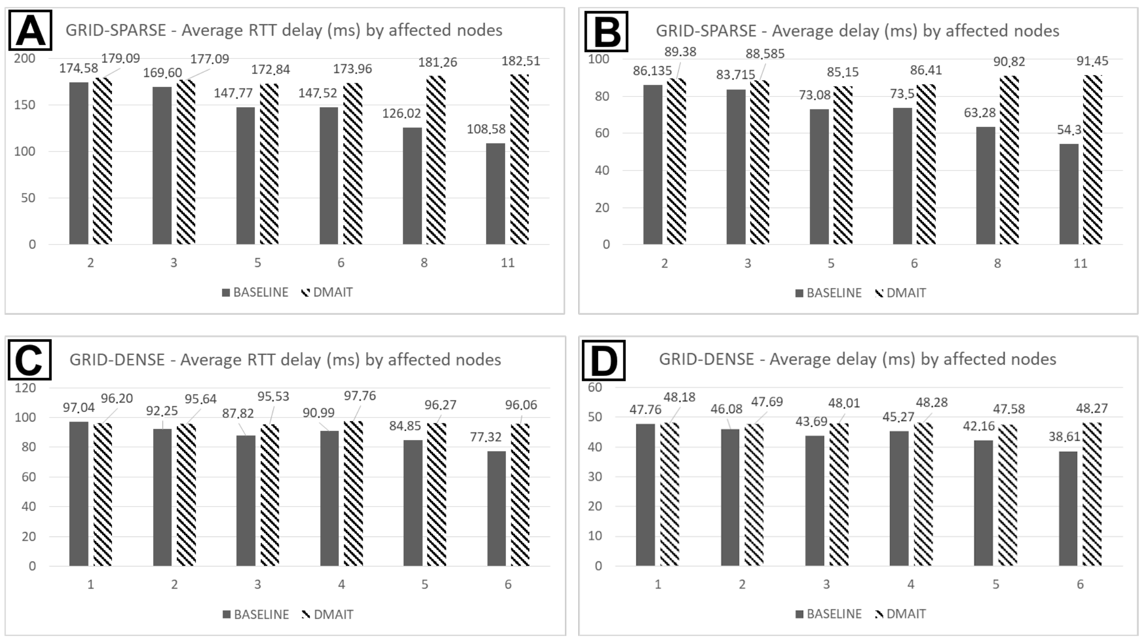

Figure 4. Comparison of delays in the GRID-SPARSE and GRID-DENSE scenarios - Average RTT delay by affected nodes in GRID-SPARSE (A). Average delay by affected nodes in GRID-SPARSE (B). Average RTT delay by affected nodes in GRID-DENSE (C). Average delay by affected nodes in GRID-DENSE (C).

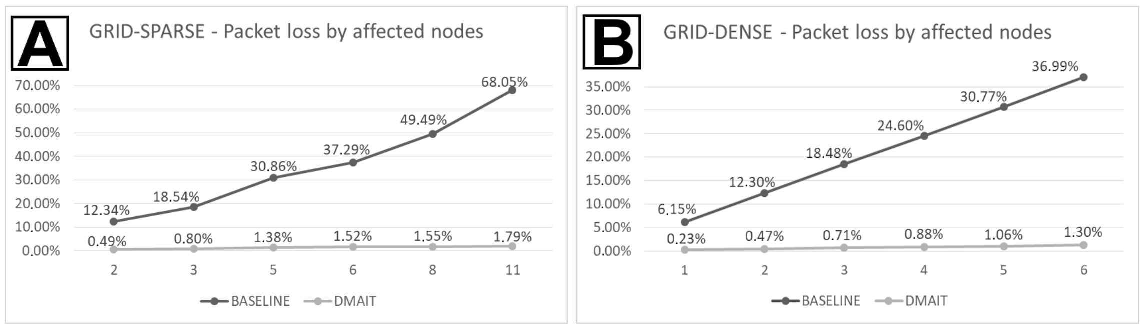

Figure 5. Packet loss analysisss - GRID-SPARSE packet loss curve (A). GRID-DENSE packet loss curve (B).

7.4. Attack monitoring

In the 5TREE scenario, which was designed to always generate the same DODAG, all experiments showed the same list of suspects, as shown below. Note that nodes 8 and 2 were identified as suspects, based on the addresses, as shown as follows.

```
**CURRENT SUSPECTS**
0208:0008:0008:0008 - NEGATIVE: 2, POSITIVE: 2, DIFFERENCE: 0,
   RATIO: 1.00
0202:0002:0002:0002 - NEGATIVE: 2, POSITIVE: 2, DIFFERENCE: 0,
   RATIO: 1.00
****
```

The GRID-DENSE scenario (Figure 6-B) shows a concentration of negative scores in nodes 2 and 8, which were in fact the attackers in this scenario. The average accuracy of the GRID-DENSE scenario experiments was 88.47%. The guide line represents half of the highest score obtained.

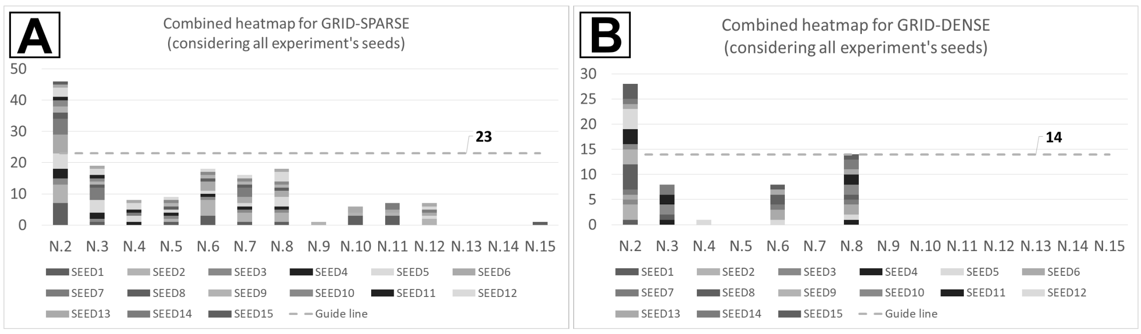

Figure 6. Heatmaps - Combined heatmap of the GRID-SPARSE scenario, showing the concentration of suspicion in node 2 (A). Combined heatmap of the GRID-DENSE scenario, showing the concentration of suspicion in nodes 2 and 8, which are the attackers (B).

In the GRID-SPARSE scenario, the scores resulted in the combined heatmap shown in Figure 6-A, which shows a concentration of suspicions in node 2, while attacker 8 shares the third position with node 6. This behavior can be explained by the position of node 8 not being favorable to acquiring children, since in this scenario, the radio range is only 1 neighbor (in each of the 4 directions), which makes node 12 its only candidate child, likely to become a child of node 11, nullifying the impact of node 8.

Note that, in both the GRID-SPARSE and GRID-DENSE scenarios, the position of node 8 limited its impact, so that in 5 of the 15 experiments it had no children, thus having zero impact. Even so, it is prominently displayed as a suspect in both scenarios.

Nodes 3 and 6 also appear as highlights in both scenarios because they are both affected by the attack of node 2, as they are used by most of the other nodes as routes to the sink.

7.5. Comparison with previous studies

In this section, experiments are carried out for comparing the performance of the proposed solution with similar works from the literature. It was not possible to obtain simulation data from any of the works listed; hence, only the experiments that had sufficient data in their publications were considered, and then replicated to the best of our abilities.

[Jamil et al. 2021] proposed a cross-layer variation of RPL, called CLS-RPL, which uses information from the link and network layers to identify potential attackers by listening to parent relays. The experiment scenario was reconstructed from the data available in the publication and the topology is shown in Figure 7-A. Node 1 represents the sink while node 13 is the attacker.

In the original work, CLS-RPL achieved an 88% packet delivery rate in the attack scenario, compared to the 36% rate of traditional RPL. During the reproduction of the experiments from the original work, it was not possible to reproduce such a large impact, of only 36% delivery rate in the control scenario (pure RPL), since the configuration of the children changed with each simulation seed and, consequently, also the number of nodes affected by the attack. In our tests, the maximum loss we obtained under these

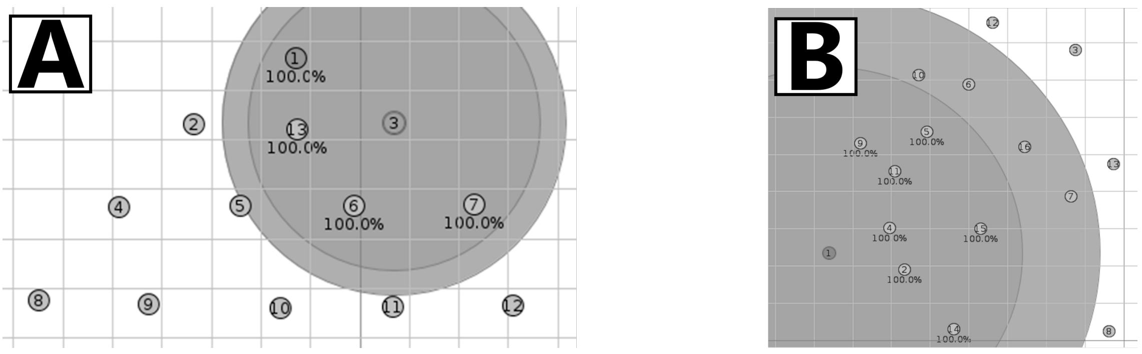

Figure 7. Reconstructed scenarios - [Jamil et al. 2021] scenario (A). [Sharma et al. 2022] scenario with 16 nodes (B)

conditions was when node 13 (attacker) had 5 descendants and, in these simulations, we obtained a packet delivery rate of 60.36%, still much higher than the 36% of the original control scenario. Chart A in Figure 8 shows the comparison with scenario 1.

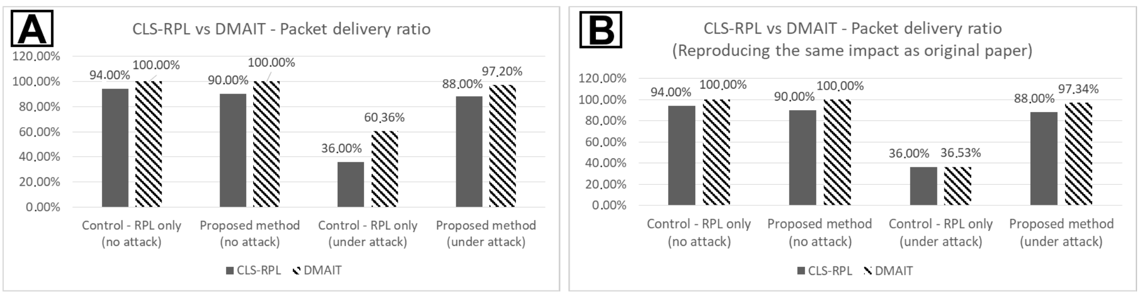

Figure 8. Packet delivery rate comparison between CLS-RPL and DMAIT - Scenario 1 (A) and scenario 2 (B).

The result shows that DMAIT achieved a higher delivery rate in scenario 1 (chart A of Figure 8). However, to verify the behavior of DMAIT with an equivalent impact, scenario 2 was generated, which swaps 1 attacker (node 13) by 2 (nodes 2 and 3), and thus increases the number of affected nodes from 5 to 9 nodes. It was also necessary to double the message transmission rate from 1 per second to 2 per second. These changes allowed achieving a package delivery rate in the control experiment of 36.53%, similar to the 36% in the original work. The results of scenario 2 are shown in chart B of Figure 8.

The [Sharma et al. 2022] 16-node scenario was reproduced, to the best of our ability, based only on the information available in the article, considering the lossless scenario. The topology comparison is shown in Figure 7-B.

The original experiment considers nodes 15 and 16 as attackers, and they are here likewise reproduced. The original work shows that 6 of the remaining nodes were severely impacted (packet delivery rate reduced by more than 10%). The original work does not mention whether the consolidated values from the experiments are combinations of results from different DODAGs. We thus tried to reproduce the experiment in 10 simulations with different seeds, all with 6 affected nodes.

The largest number of affected nodes we were able to reproduce was 6 nodes, and one of them was always node 16, which was also one of the attackers. In all the experiments with 6 affected nodes, node 16 positioned itself as a child of node 15, another attacker. The exact packet loss values are not shown in the original work, and were thus estimated from the charts. In the original work, node 3 was affected, while in our experiments node 16 (also an attacker) was affected. The other affected nodes were the same: 6, 7, 8, 10 and 13.

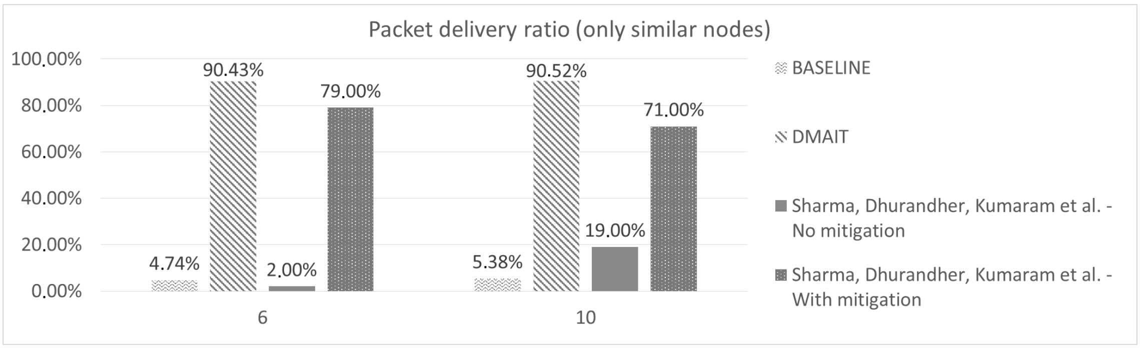

Figure 9. Packet delivery rate comparison with [Sharma et al. 2022] scenario - Comparison of the packet delivery rates of DMAIT and BASELINE (control), of nodes with similar loss and recovery profiles.

Figure 9 shows the comparison between the two nodes (6 and 10) with the most similar impact and recovery profiles, between the original work and our reproduction of the experiment. The results also show the recovery of the delivery rate in the DMAIT experiments to the level of 90%, in all affected nodes. The original work does not show the same consistency, with values between 62% and 90%.

7.6. Challenging issues and future work

The number of behaviors and mechanisms added with the implementation of the DMAIT algorithm on Contiki greatly increased the size of the generated ROM, higher than the available at known models of physical MOTES, which prevented us from obtaining data regarding the energy consumption of the nodes. Removing parts of the algorithm would negatively impact the results and we therefore chose to have the full implementation of the algorithm. Future research can aim to seek an optimization of the code that allows simulation on known devices, and to obtain energy consumption data.

8. Conclusion

The experiments show that the recovery of the packet delivery rate promoted by the DMAIT framework is consistent and superior to the control scenarios and compared works, succeeding in mitigating traffic interruption attacks. The delivery delay was also constant with the application of DMAIT regardless of the number of nodes affected. The suspects were consolidated by the sink and it was possible to identify the attackers among the suspects in most scenarios.

Another contribution of this work was the results comparison method that considers the number of affected nodes. Stratifying the results by number of affected nodes allows a direct comparison among different methods, which is not possible if data from scenarios with different numbers of affected nodes are consolidated.

Finally, the security mechanisms added with DMAIT promise resilience to attacks launched directly against the framework using HMAC and symmetric keys.

The results of the experiments, in both raw and processed form, and the source code for the DMAIT framework PoC and BASELINE application are available at:

Github: `https://github.com/danielfsoriano/DMAIT/tree/0.0.1`

Google Drive: `https://drive.google.com/drive/folders/1b-GhbZBkaD_B0xJPaNTwxcVSfd0yxCbC?usp=drive_link`

Also as a DOI: `https://doi.org/10.5281/zenodo.10699174`

References

Airehrour, D., Gutierrez, J., and Ray, S. (2017). A trust-aware rpl routing protocol to detect blackhole and selective forwarding attacks. *Australian Journal of Telecommunications and the Digital Economy*, 5.

Alansari, Z., Anuar, N. B., Kamsin, A., and Belgaum, M. R. (2023). Rplad3: anomaly detection of blackhole, grayhole, and selective forwarding attacks in wireless sensor network-based internet of things. *PeerJ Computer Science*, 9.

Algahtani, F., Tryfonas, T., and Oikonomou, G. (2021). A reference implemenation for rpl attacks using contiki-ng and cooja. In *2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 280–286.

Amish, P. and Vaghela, V. (2016). Detection and prevention of wormhole attack in wireless sensor network using aomdv protocol. *Procedia Computer Science*, 79:700 – 707. Proceedings of International Conference on Communication, Computing and Virtualization (ICCCV) 2016.

Bal, M. (2014). An industrial wireless sensor networks framework for production monitoring. In *2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE)*, pages 1442–1447.

Chen, B., Li, Y., and Mashima, D. (2018). Analysis and enhancement of rpl under packet drop attacks. In *2018 10th International Conference on Communication Systems Networks (COMSNETS)*, pages 167–174.

Chowdhury, M. and Kader, M. F. (2013). Security issues in wireless sensor networks: A survey. *IFGN*, 6:97–116.

Heurtefeux, K., Erdene-Ochir, O., Mohsin, N., and Menouar, H. (2015). Enhancing rpl resilience against routing layer insider attacks. In *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*, pages 802–807.

Iova, O., Theoleyre, F., and Noel, T. (2015). Exploiting multiple parents in rpl to improve both the network lifetime and its stability. In *2015 IEEE International Conference on Communications (ICC)*, pages 610–616.

Jamil, A., Ali, M., and Tharwat, M. (2021). Sinkhole attack detection and avoidance mechanism for rpl in wireless sensor networks. *Annals of Emerging Technologies in Computing*, 5:94–101.

K, M., Jebadurai, I., Jeba, G., and Jebadurai, J. (2022). Mitigating sinkhole attack in rpl based internet of things environment using optimized k means clustering technique. pages 502–507.

Kumaran, S., Kailasanathan, N., and Mohan, S. (2016). Review of asymmetric key cryptography in wireless sensor networks. 8:859–862.

Maidamwar, P. and Chavhan, N. (2018). A Survey on Security Issues to Detect Wormhole Attack in Wireless Sensor Network.

Muzammal, S. M., Murugesan, R., Jhanjhi, N., Humayun, M., Osman, A., and Abdelmaboud, A. (2022). A trust-based model for secure routing against rpl attacks in internet of things. *Sensors*, 22:7052.

Oludele, A., Okesola, O., Okokpujie, K., Fowora, D., and Adebiyi, A. (2018). Cryptography and the improvement of security in wireless sensor networks.

Parasuram, A., Culler, D., and Katz, R. (2016). An analysis of the rpl routing standard for low power and lossy networks.

Patel, B., Vasa, J., and Shah, P. (2023). Forwarding neighbor based sink reputed trust based intrusion detection system to mitigate selective forwarding attack in rpl for iot networks. *SN Computer Science*, 4.

Prathibha, S. R., Hongal, A., and Jyothi, M. P. (2017). Iot based monitoring system in smart agriculture. In *2017 International Conference on Recent Advances in Electronics and Communication Technology (ICRAECT)*, pages 81–84.

Sharma, D., Dhurandher, S., Kumaram, S., Gupta, K., and Sharma, P. (2022). Mitigation of black hole attacks in 6lowpan rpl-based wireless sensor network for cyber physical systems. *Computer Communications*, 189.

Singh, R., Singh, J., and Singh, R. (2016). Wrht: A hybrid technique for detection of wormhole attack in wireless sensor networks. *Mobile Information Systems*, 2016:8354930.

Stephen, R. K. and Arockiam, L. (2018). An enhanced technique to detect sinkhole attack in internet of things. *International journal of engineering research and technology*, 5.

Sudhakar, B. and Abhinaya, E. (2019). An efficient network discovery storage based resilient packet-forwarding scheme for the mitigation of black hole and wormhole attacks in 6lowpan sensor networks. *International Journal of Recent Technology and Engineering*, 8(2):1543–1547.

Surendar, M. and Umamakeswari, A. (2016). Indres: An intrusion detection and response system for internet of things with 6lowpan. pages 1903–1908.

Violettas, G., Simoglou, G., Petridou, S., and Mamatas, L. (2021). A softwarized intrusion detection system for the rpl-based internet of things networks. *Future Generation Computer Systems*, 125:698–714.

Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, J., and Alexander, R. (2012). Rpl: Ipv6 routing protocol for low-power and lossy networks. RFC 6550, RFC Editor. `http://www.rfc-editor.org/rfc/rfc6550.txt`.