IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 28 May 2023, accepted 21 June 2023, date of publication 18 July 2023, date of current version 28 July 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3296558

RESEARCH ARTICLE

A Novel Energy-Efficient Scheme for RPL Attacker Identification in IoT Networks Using Discrete Event Modeling

DIPOJJWAL RAY[1], **PRADEEPKUMAR BHALE**[1], **(Student Member, IEEE), SANTOSH BISWAS**[2], **(Senior Member, IEEE), PINAKI MITRA**[1], **(Member, IEEE), AND SUKUMAR NANDI**[1], **(Senior Member, IEEE)**

[1]Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, Guwahati, Assam 781039, India
[2]Department of Computer Science and Engineering, Indian Institute of Technology Bhilai, Raipur, Chhattisgarh 492015, India

Corresponding author: Dipojjwal Ray (dipojjwal@iitg.ac.in)

This work was supported in part by the ICPS, Department of Science and Technology (DST), Government of India, for the project titled "Formal Methods for Modeling and Verification of Intrusion Detection System in Wireless Networks."

ABSTRACT The Internet of Things (IoT) paradigm facilitates communication for a multitude of connected smart objects and provisions essential and mission-critical services across diverse sectors. To route packets, IoT networks use Routing Protocol for Low-Power and Lossy Networks (RPL) by default. However, RPL lacks security features by design, making IoT-RPL prone to low-overhead internal attacks such as the rank and version attacks. The attack and normal traffic are found to be identical, making detection challenging for signature-based and anomaly-based Intrusion Detection Systems (IDS). Moreover, a formal proof of correctness of IDS schemes is lacking. In this paper, we propose a novel rank and version attack detection and rank attacker location identification mechanism that utilizes active probing and Discrete Event System (DES) based IDS. Our proposed IDS scheme is centralized with inputs from sensing at the leaf levels. IDS uses as an intelligent probing technique that helps distinguish normal and attack behaviour. Further, DES is used to model the normal and attack specifications. A DES diagnoser, constructed from the DES models, generates an alert when a malicious node is identified. We also prove the correctness and completeness of our scheme. The DES framework is implemented only at root node, therefore using our IDS does not require any heavy deployment, protocol modifications, or training. Proposed method is implemented in simulation and testbed, with a sufficiently large number of IoT devices. We compare our scheme to state-of-the-art approaches. Our performance is found to be energy-efficient, having minimal false positives and achieving more than 99% accuracy in detecting intrusions and identifying the malicious nodes.

INDEX TERMS RPL, the Internet of Things (IoT), network security, intrusion detection system (IDS), discrete event systems (DES), rank attack, version number attack.

I. INTRODUCTION

The Internet of Things (IoT) system is witnessing a rapid evolution, due to the ever increasing number of connected smart and pervasive devices [1]. Consisting of a multitude of connected heterogeneous objects, which we rather call as *things*, the IP-connected IoT is spread over diverse domains like smart cities, autonomous vehicles, industrial cyber-physical systems, smart homes, e-health sector, etc [2], [3]. IoT networks are typically Low power and Lossy Networks (LLN), comprising mostly of embedded sensors and actuators. Not only do such networks require to uniquely address billions of these connected devices, but also support embedded technologies for sensing and gathering data from the environment. With the mighty responsibilities in hand, IoT-connected resource constrained devices suffer from major operational challenges like constrained processing capabilities, inadequate memory and limited power. Hence,

The associate editor coordinating the review of this manuscript and approving it for publication was Peng-Yong Kong.

IoT remains vulnerable to a wide array of attacks because of insecure LLNs, device limitations, varying technologies, etc.

To enable efficient and reliable communication, IETF has standardized IPv6 Routing protocol for Low Power and Lossy Networks (RPL) [4]. The design of RPL is tailored for low-power IoT devices. RPL maintains loop-free Destination Oriented Directed Acyclic Graphs (DODAG). A DODAG is created and maintained using control messages, primarily, DODAG Information Object (DIO) for upward paths, DODAG Advertisement Object (DAO) for downward paths and DODAG Information Solicitation (DIS) for node joining. RPL ensures cost-optimized topologies by ordering participating nodes on the basis of an integer cost function, *rank*. Individual rank of a node determines its position in the DODAG, relative to a 6BR sink node (root). Also, a single *version* number prevalent in the DODAG is maintained in DIO for consistency. Though RPL provisions various mechanisms and is secure enough from external attackers, yet the resource constrained nature of IoT devices, the typical characteristics of IoT networks such as lossy links, lacunas in infrastructure, dynamic topology, etc., can render IoT-RPL susceptible to internal attacks [5], [6], [7], [8].

Various internal attacks have been shown in the literature, of late, that make illicit use of RPL. Puppet attacks [9], advanced vampire attacks [10] make use of forged source routes, while attacks like energy depletion attack [11] and vampire attacks [12] drain resources by repeatedly sending useless data packets. Sybil attacks [13] and spam DIS attacks [14] have been shown to make use of DIS messages with counterfeit identities, essentially causing denial of service. DIO suppression attacks eavesdrop DIO messages for replaying it repeatedly in fixed intervals [15]. Out of the various DIO-specific attacks explored, proposed rank and version attacks continue to be of paramount importance since they are of low-overhead and are realizable using DIO only. To launch such attacks, the rank and version number fields of a DIO message are fabricated causing formation of loops, sub-optimal routes, traffic redirection and network partitioning. Significant path delay is incurred since a large number of control messages are exchanged in the DODAG, resulting in energy depletion of the constrained nodes and disruption of network services. Moreover, rank attacks may be combined with other cross-layer attacks like selective forwarding attacks to alleviate the damage caused.

Proposed methods for securing IoT networks against RPL rank and version attacks have their own typical limitations [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26]. Cryptography-based mitigation schemes are resource exhaustive and computationally heavy, especially in a network of resource constrained devices. Machine learning-based approaches require investment of extensive training time, as per the system under consideration. Protocol-based approaches require modifying the protocol policies. IDS based approaches do not suffer from these above limitations, but the implementation of these schemes for rank attack detection is challenging since attack behaviour resembles and normal behaviour. Hence, use of signature-based IDS and anomaly-based IDS schemes in the context of IoT attacks generate a large number of false positives. Furthermore, there exists many variants of rank attacks which present complex characteristics to evade detection capabilities of IDS. Formal verification of IDS schemes are also lacking.

This paper presents an intelligent probing based scheme for the detection of rank and version attacks that also identifies location of malicious nodes. The probing mechanism helps differentiate the normal and attack behaviour. Our scheme incorporates Discrete Event System (DES) based IDS [27], [28], [29] and a set of agents with event monitoring enabled, that make use of probe packets [30], [31], judiciously. System failures and network attacks involve analogous behavioural deviations from the normal system functioning, which motivates the use of DES based IDS. Deploying our IDS does not require a change in protocol policies, encryption, extensive training time or any need for proprietary hardware support. Using our IDS also helps ensure a formally verifiable proof of correctness of our approach. The major contributions in this paper are enumerated as follows:

- We propose a novel rank attacker identification scheme that also detects version attacks in IoT-RPL. Our scheme makes use of an intelligent active probing technique that helps create a deviation of attack traffic and normal traffic [30]. Our proposed scheme is centralized and uses a DES based IDS.
- We extend the power of traditional DES based IDS with attack type modeling for attacker identification.
- We prove the correctness and completeness of our approach by enumerating all the attack cases.
- The performance of our scheme is tested through simulations and real testbed. The experimental results highlight the applicability of our approach. Comparison of our scheme to state-of-the-art countermeasures shows our approach is energy-efficient with less packet overhead. The proposed solution is scalable, has minimum false positives and achieves more than 99% accuracy in identifying the malicious nodes.

The rest of this paper is organized as follows: We discuss the related works and motivation in Section II. Section III is background. The design of our proposed scheme using a DES based IDS is presented in Section IV. Experimental results are summarised in Section V, highlighting the performance of our scheme, and we finally conclude with Section VI.

II. RELATED WORK

We here discuss the various schemes proposed in the literature. The existing methods either employ mitigation techniques [16], [17], [32] using cryptographic solutions [18], [19], [33], acknowledgement based schemes [34], trust based methods [20], [21], [22], recent machine learning approaches [23], [24], [25], [26], or IDS based approaches [35], [36] using specifications and mathematical (statistical) methods to make DODAG secure. One of the primary works, VeRA [33], suggested the use of one-way hash functions generated by RPL root, where each of the nodes authenticate neighbours' rank by repeated usage of the function. TRAIL [34] improved upon VeRA by abstaining from a fully cryptographic technique. Newer attack vectors are also identified. Their proposed approach detects and mitigates topological inconsistencies in the network by checking for upward routes. They make use of encryption chain authentication as opposed to MAC authentication, thus ensuring backward secrecy. Their scheme lacks in scalability and requires maintaining state information. Nikravan et al. [18] utilise an identity based offline-online signature. Their solution is scalable compared to VeRA and TRAIL, requiring the size of signature to be independent of the network size. The above approaches to mitigate rank attacks however are resource exhaustive or computationally heavy.

Trust based methods have also been largely used in this direction [21], [22], [37]. They mostly resort to reputation score calculations and trust values for attack detection. SecTrust-RPL [20], a time-based trust-aware routing protocol, used a trust based principle that computes reliability, gained from message exchanges. They also validate their approach, however, it required each node to be run in promiscuous mode for sniffing packets. Later, a dynamic hierarchical trust model is proposed in DCTM-RPL [38]. Secure communication is shown to have been achieved by building up trust above a threshold value in their approach. Among the various protocols proposed [17], [39], a secure protocol, SRPL-RP [40], mitigates rank and version number attacks. It uses a timestamp threshold to validate a legitimate sender node. Though their approach improves upon overhead and average energy consumption, energy is wasted in the absence of any attacker. Furthermore it may be noted that protocol based approaches modify the protocol policies. There has also been significant contributions, of late, that use machine learning based methods. Specifically, deep learning based [25], [41] and artificial neural network based approaches [26] have been applied to detect rank and other routing attacks. However, it is worth mentioning that such approaches require investment of extensive training time and further improvements in their accuracy can be achieved by better dataset.

Usage of IDS has received considerable attention over the years in the security research community. Network based Intrusion Detection Systems (NIDS) have been largely employed to secure the IoT network against attacks [35], [36], [42], [43]. NIDS for IoT are mostly signature-based (or, knowledge-based), anomaly-based, specification-based or hybrid [44], [45], [46]. SVELTE [36], one of the notably important proposal has shown the use of real-time intrusion detection in IoT. A specification-based IDS with hybrid placement that detects blackhole, sinkhole and selective forwarding attack, SVELTE used a mix of both signature based and anomaly based methods. While an IDS module runs on the root node, the firewall and response model runs on every node, which is again resource intensive. Some of the other limitations of the scheme are false detection and the lack of DIO synchronisation. Recently, FORCE [45], a specification based IDS that exploits the parent-child relationship is proposed and performs better than SVELTE in terms of detection rates and energy consumption. A version attack detection scheme using temporal logic based IDS [47] is shown, but a comparison of their scheme is lacking. A few works [32], [48], [49] improve upon SVELTE in terms of false positives. Version attack is mitigated and attacker is identified using trust-based distributed IDS [50] and also by distributed monitoring mechanisms [51]. A sink-based IDS is proposed in [46], but the scheme suffers from high computational overhead and average power consumption.

Few approaches in the literature have performed malicious node identification and isolation [32], [50], [51], [52]. In IoT networks, control packets are exchanged in the RPL for maintenance and a rank update legitimacy cannot be directly verified, since they are not differentiable across normal and spoofed conditions. An increased rank may be advertised due to various genuine reasons like a node gone off or not running, node services interrupted, etc. Moreover, variations of rank attacks lack direct anomalies or known signatures. In this regard, signature-based and anomaly-based IDS approaches in turn result in an increased number of false positives when generating relevant signatures or statistics. We overcome the discussed shortcomings by developing an energy-efficient and formally verifiable probing based scheme. Probing helps differentiate the attack characteristics from the normal network characteristics. Analyzing the topological changes due to rank attacks aid our development of probing techniques for malicious node identification. We not only detect but also identify the location of the malicious node with enhanced precision, lower false positives and lower detection time. Our scheme is centralized and uses a DES based IDS, correctness of which can be formally verified. DES based IDS are accurate and generate minimal false positives [27], [28], [29], [53]. Moreover, using DES based IDS do not require a change in protocol policies, extensive training time, encryption or a need for proprietary hardware support.

III. BACKGROUND

In this section, we discuss the preliminaries of RPL protocol, DODAG creation and RPL attacks, namely, increased rank and version attacks, in particular.

A. RPL PROTOCOL

RPL is inspired from distance-vector routing protocol, source routing protocol and DAG. It is the de-facto routing protocol that operates on top of IEEE 802.15.4 MAC layer while supporting multipoint-to-point traffic using upward routes, point-to-multipoint traffic using downward routes and a combination of the above routes to facilitate multipoint-to-multipoint traffic. Independent downward routes and upward routes are established in DODAG. Depending on the mode of operation, downward routes may be optionally supported. RPL supports three node types, namely, (i) Low Power and

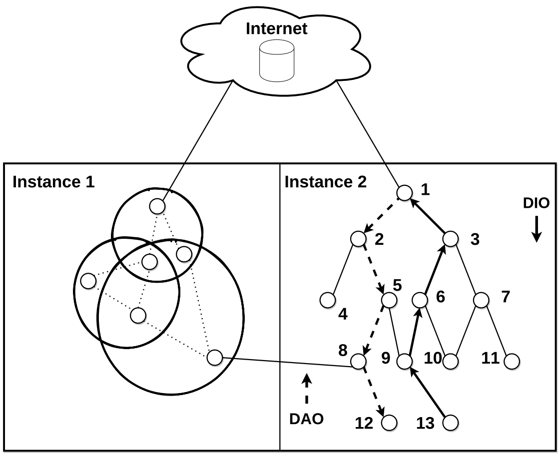


FIGURE 1. RPL DODAG.

Lossy Border Routers (LBRs) which acts rather as gateway between LLNs and the Internet, (ii) Routers which can forward as well as generate traffic and (iii) Hosts that can generate but not forward traffic. Nodes are organized in the form of DODAG tree with a provision for parallel execution of multiple RPL instances, as shown in Figure 1. An RPL instance is uniquely characterized using RPL Instance ID and a DODAG using DODAG ID. The DODAG root is a special kind of node that acts as an LBR or a destination sink. The root determines and maintains the DODAG configuration parameters and starts disseminating DIOs [54].

In RPL, an ICMPv6 control message can be any one of these following types: (i) DODAG Information Solicitation (DIS) (ii) DODAG Information Object (DIO) (iii) DODAG Advertisement Object (DAO) (iv) DODAG Advertisement Object Acknowledgement (DAO-ACK).

Rank is an integer value assigned to each node in the DODAG. All the nodes conforming to the inclusion policy in the DODAG instance are ordered on the basis of these values as per an instance defined metric. They are a measure of the position of the node relative to the sink node. A higher rank value pertaining to a node means it is more distant from the sink compared to another node with a lower rank value. Objective Functions (OF) are used for topology optimization depending on a set of goals that need to be met, such as link quality, hop count, energy consumption, etc. OF is used by the RPL to select the best routing path. Instances use OFs to determine the rank. The OF determines metrics that are included in the DIO messages. OF is realized using Objective Code Point included in the DIO configuration options.

1) DODAG CREATION AND MAINTENANCE

Creation and maintenance of an RPL DODAG is done using the DODAG control messages. When a DODAG is built, the root link local multicasts DIO messages for building upward paths. The rank value, objective code point and node ID are included in the DIO messages [55]. DIO messages are periodically disseminated downwards, where the period is decided by the Trickle algorithm [56]. From the received DIO messages from neighbours, each node has the decision on selection of its parent set among its neighbours. Among its parent set, it selects a preferred parent from the best advertised rank value. Thus, when a node forwards a message to the DODAG root, the preferred parent is chosen by default. The received DIO message is then updated at the node and forwarded to its neighbours. On completion of DIO message exchanges till the leaf node, the upward route is created upto the DODAG root, consisting of preferred parents from each node. A node uses DIS broadcast messages to join a DODAG. DAO messages are used by the nodes for building downward paths.

B. RANK AND VERSION NUMBER ATTACK

Alteration/Spoofing attacks in RPL have been widely investigated. Rank and version attacks in RPL are identified as misappropriation or alteration attacks where the ranking scheme is exploited, indirectly, making false advertisements using DODAG control packets [7], [16].

1) VERSION NUMBER ATTACK

RPL incorporates versioning in DODAG to prevent loop formation and to ensure updated topologies. A malicious node makes use of the version number field to attract descendant nodes. False version number updation in the DIO advertisements practically actuate a DODAG tree rebuilding operation affecting the network performance, indirectly. As a result, energy exhaustion, loop formation, increased overheads ensue. Moreover, it provides avenues for launching more serious combined forms of attack.

2) INCREASED RANK ATTACK

One or more node(s) may misbehave in the network by increasing the rank values. We here restrict ourselves to the case where the network has a single misbehaving node. The malfunctioning node suddenly multicasts a DODAG Information Object (DIO) message to its neighbor nodes with an incremented rank value. The neighbor nodes, then, does the same, recursively, till the network upward routes are updated. Hence, there is a huge burst in control packet traffic in the network. The nodes being resource constrained illicitly face exhaustion of their battery. As a result of this type of attack, the network may even include loops that may not be mitigated using local repair mechanisms in RPL. Otherwise, the node simply joins at a lower rank in the network (i.e., more distant from the DODAG root) and such behavior may be primarily intended to starve a targeted node by disrupting communication.

C. INCREASED RANK ATTACK TIMELINE

The increased rank attack timeline is shown in Figure 2. The time-slots $T1$ through $T4$ are briefly explained. [$T1$:] R is the root of the RPL DODAG while other nodes are numbered $\{A, N1, \ldots, N6\}$. Node, A is rendered vulnerable. [$T2$:] The vulnerable node probes rank values of the

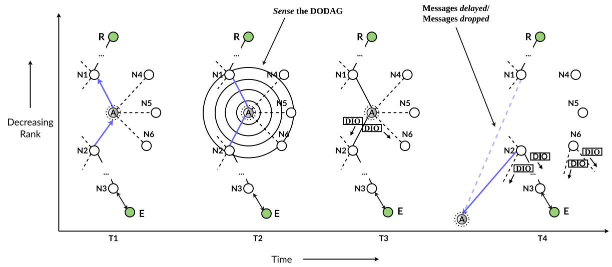


FIGURE 2. **Rank Attack Timeline.**

neighbouring nodes. [$T3$:] On having chosen a rank value, A now multicasts DIO messages with its updated rank value. [$T4$:] DIO messages are exchanged till leaf nodes update the upward route. The DODAG topology is modified at A.

D. INTRUSION DETECTION SYSTEMS

Intrusion Detection Systems (IDS) are identified as one of the basic tools that are employed to protect networks and data. An alert is raised to the system administrator if any suspicious activity is detected by the IDS. An IDS can be software or hardware that are built to monitor and analyze the network packets that are sniffed or the events that occur in the host machine. Designing an IDS requires considering the processing ability and memory capacity of the nodes where they may be deployed. The primary components of an IDS are sensors that collect data, and an IDS engine that analyzes the collected data and reports to a network administrator for suitable actions. IDS are classified in the literature depending on the source of the data being monitored, depending upon the strategy it takes, and also depending on the monitoring techniques. Source of monitoring the data classifies IDSs into NIDS, HIDS and Hybrid. Based upon the strategy of detection, IDSs are classified as signature-based, anomaly-based, specification-based, and hybrid. Depending on the monitoring technique, IDS are classified into active and passive monitoring which are further subdivided into centralized, decentralized and hybrid monitoring techniques.

IV. PROPOSED RANK ATTACKER IDENTIFICATION SCHEME

Here, we present the different aspects of our proposed scheme for RPL attack detection and identification. We introduce DES based IDSs followed by an overview of the detection methodology using our proposed IDS. We then discuss the employed techniques and algorithms to identify the attacker. The construction of normal, attack models and DES diagnoser that are indispensable for attacker identification are described next. Proof of correctness and completeness is presented subsequently. We assume that an attacker is unable to differentiate probe packets from normal packets and hence responses to them.

A. DES BASED IDS

Classical DES theory has been largely adopted in systems for Fault Detection and Diagnosis (FDD) [57], [58], [59]. Motivated from fault diagnosis, DES based IDSs have been successfully used in network attack detection [27], [29]. The characteristic similarities of network attacks and faults in DES literature is what motivates its usage. The basic idea is to develop a model for the normal functioning of the network and another for attack (fault) behavior. Additionally, multiple fault types in DES literature are diagnosed by developing exclusive fault DES models corresponding to each fault type. Each fault type leads to unique deviations from the normal behavior. Analogously, we augment traditional DES based IDS with attack types in our work here. It may be noted that an attack type corresponds to behavior of the network under the influence of a particular attacker. Attack type DES models corresponding to the location of the attacker are modeled. In DES based IDS a DES diagnoser is used as our IDS engine. It is a state estimator automaton which is constructed from the knowledge of normal and attack type DES models. The diagnoser observes system event traces and gives a decision on the system condition being normal or under attack by generating alerts. To summarize, by using DES based IDS, and given all possible attack instances, it can be ascertained if an attack can always be exclusively identified, correctly and completely.

B. OVERVIEW OF PROPOSED ATTACKER IDENTIFICATION PROCEDURE

The primary research challenges in detection of rank attacks are as follows: **(i)** Nodes with rank values lower than the malfunctioning node, including the 6BR root, remain unaware of the inconsistency created in any subtree **(ii)** Normal scenario cannot be differentiated from the attack scenario by monitoring network traffic or topological changes. Sensing of network events at the leaf level using agents helps overcome the first challenge, while an intelligent probing technique helps overcome the second challenge discussed above. Active probe packets generate distinguishable packet sequences between normal and attack scenario. The system we consider consists of an IoT network of resource constrained devices using RPL. We use a centralised IDS, functioning at the network layer, working in a distributed manner with the help of agent nodes. An example of a DODAG with our IDS and agents deployed is demonstrated using Figure 3. The 6BR root (n_R) is software controlled and IDS handles communication for this node. The set of agents, $\mathcal{T} = \{n_1, n_2, \ldots, n_t\}$, with event monitoring enabled are deployed at the leaves. Henceforth, the IDS node is designated as n_R. The notations used are listed in Table 1.

Components in the IDS: The block diagram of our proposed IDS with the basic components is shown in Figure 4 and are discussed here as follows:

- **Packet Sniffer:** It captures control and data packets in the network while working in promiscuous mode. Relevant packets are sniffed and others are dropped. It then forwards the sniffed packets to the "RQST_RSP_HANDLER()" component.

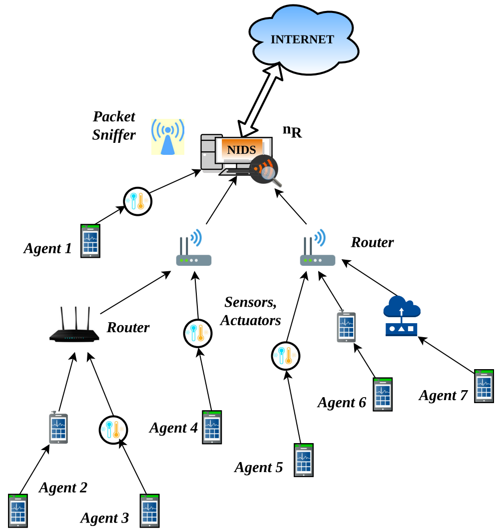


FIGURE 3. IoT network DODAG representation with IDS and agents deployed.

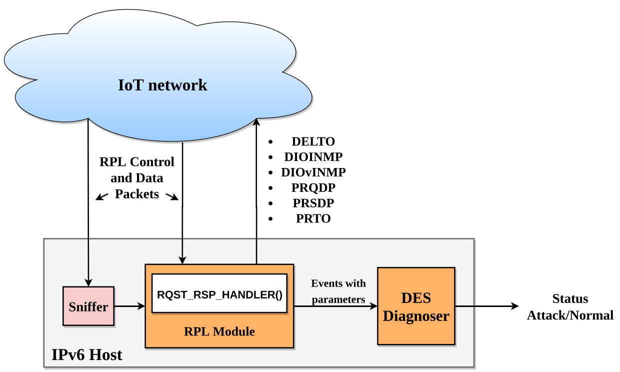


FIGURE 4. Architecture of proposed IDS.

TABLE 1. Notations.

Notation	Meaning
$DIORQP$	Rank Update Packet
$DIOINMP$	DIO Rank Update Intimation Packet
$DIOvINMP$	DIO Version Update Intimation Packet
$PRQDP$	Probe Request Data Packet
$PRSDP$	Probe Response Data Packet
$PRSDP^*$	Delayed Probe Response Data Packet
PR_TO	Probe Timeout Event
$URDES$	Unreachable Destination Message

- **RQST_RSP_HANDLER():** Its prime responsibility is to extract vital information from the control or data packets like source client's IP address, MAC address, Transaction identifier, etc. It also makes note of rank and version value attributes and generates the events *DIOINMP, DIOvINMP, URDES, PRQDP, PRSDP, PR_TO, PRSDP**. The generated events are passed to the DES diagnoser. The working procedure of this handler is described in Section IV-E.
- **DES Diagnoser:** This component diganoses the attacker node and is implemented as a software module. Given the knowledge of the DES model specifications pertaining to normal and attack type conditions, the diagnoser can be constructed. RQST_RSP_HANDLER() passes information regarding network events to the diagnoser. Based on the event parameters that are shared, the diagnoser generates an alert on attack detection or identification of malicious nodes. The usage and construction of the diagnoser is described in Section IV-F3.

Attack detection and identification is sequentially carried out in phases. Version attack detection phases are **setup**, **intimation** and **diagnosis**, whereas, rank attack detection consists of **setup**, **intimation**, **active probing** and **diagnosis**. The working principle of our proposed scheme is demonstrated next. The flow of our scheme is shown using Figure 5. Prior to attack, network traffic is monitored and data is logged to setup the IDS as shown in the initial module. This forms the setup phase. IDS performs all the normal functionalities besides gathering and analysing the sniffed data in this phase. Considering there are t agents deployed, t tables (TPATH) are maintained and updated during this phase. Each table consists of round-trip time (RTT) values and information of the intermediate nodes between n_R and an agent. The table elements are ordered on rank values. After the IDS is setup, suppose an irregular DIO is received by an agent, n_j, where $n_j \in \mathcal{T}$ and $1 \leq j \leq t$. It then intimates this information as obfuscated application data to n_R after a random delay. This is the intimation phase. In case a version inconsistency is intimated, the diagnoser (IDS engine) validates the report and declares the status to be normal or a version attack, which is the diagnosis phase. On the other hand, on receipt of an irregular rank update intimation from an agent n_j, a j^{th} table is chosen. Subsequently, the RQST_RSP_HANDLER() on behalf of n_R sends ICMPv6 request packets to the nodes in this table, one by one, to probe for topological inconsistencies in the DODAG. This forms the active probing phase. An acknowledgement (ACK) response is generated for a probe request packet when received at a destination node. Now, a probe ACK response may not be received at all at n_R, genuinely, if any node has gone off, or if a link is broken, or a loop is present, and falsely if an attacker is present. So a missing ACK probe response cannot be directly marked as a suspicious activity. We hence characterise the received responses based on RTT values. RTT for a destination that is probed is computed and compared with RTT computed before intimation. Depending upon the learnt characteristics of RTT values from the sequence of probe packets sent, further probing is continued or a decision is taken by the diagnoser. The latter validates the probe responses against the DES model specifications provided at the start corresponding to normal as well as attacker specific behavior. Our normal and attack modeling capture the characteristic differences. The RTT values computed using the probing technique for a parent and child pair pose unique characteristics that help differentiate a normal and attack scenario. Moreover, the RTT characteristics for the sequence of nodes probed in the chosen table, i.e., j^{th} here, are differentiable in case of a specific

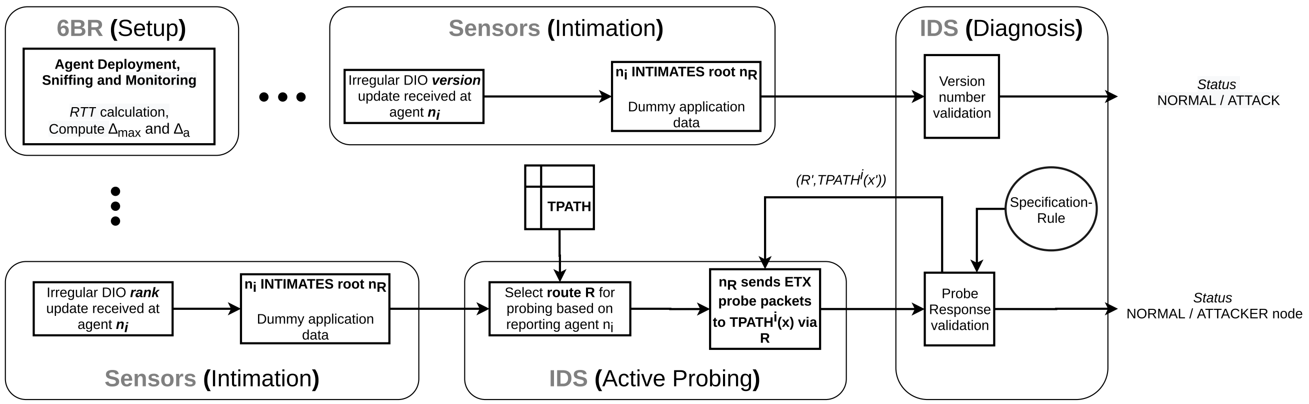


FIGURE 5. Workflow of proposed scheme.

TABLE 2. Table for *TPATH*^{2}.

Node	Link-local IPv6 address	MAC address	Rank	RTT
B	fe80::2ca:3fff:fed6:8d56	00:ca:3f:d6:8d:56	1	1.23s
C	fe80::3340:70ff:fedf:71f1	31:40:70:df:71:f1	2	4.15s
D	fe80::f6eb:3fff:fe92:3cd2	f4:eb:3f:92:3c:d2	3	5.7s
E	fe80::6fbf:35ff:fec6:1ffd	6d:bf:35:c6:1f:fd	4	7.68s
n_2	fe80::a68d:bcff:fe6c:89d4	a4:8d:bc:6c:89:d4	5	9.84s

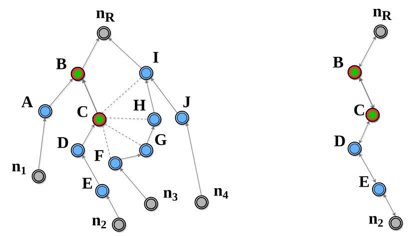


FIGURE 6. A DODAG instance (left) and path *TPATH*^{2} (right). IDS nodes are denoted as *gray* circles, non-attack nodes are denoted in *blue* circles, suspected attack nodes are denoted in *red green* circles.

attack node. The phases in our detection procedure are now sequentially demonstrated.

C. IDS SETUP

This phase consists of administrator intervention for parameter setup. Traffic is monitored, relevant data is collected and parameters are measured for Network Traffic Analysis (NTA) purposes. Regular monitoring and sniffing yield to our detection procedure by maintaining tables and computing essential parameters, respectively. An array of table pointers, **TPATH**, is used for storing the intermediate node information. *TPATH*^{2} in the example DODAG of Figure 6 is shown in Table 2. An element of the array, *TPATH*^{j}, stores the IP, MAC, RANK and RTT values of the intermediate nodes along the path connecting the IDS, n_R to an agent n_j. $\langle TPATH^j \rangle_{SIZE}$ represents the size of *TPATH*^{j}, i.e., the number of nodes along the path $\overline{n_R n_j}$, excluding the root node. Values such as maximum RTT and maximum round-trip delay for 1-hop are computed and continuously updated. Variables Δ_{max} and Δ_a hold the maximum delay and admissible delay values, respectively. Sniffers deployed at n_R capture the traffic of underlying network as demonstrated in the Figure 3. The sniffing component retrieves general information from the packets communicated. The retrieved information from the control and data packets consist of DODAG ID, packet type (i.e., DIO, DAO, DIS, DAO-ACK, application), sender IP, destination IP address, and forwarding path information. Rank and version number values are also looked into and stored when necessary. The agent intimation phase is demonstrated in the following subsection.

D. INTIMATION

Our scheme consists of pieces of software, which are small programs, as agents for reporting any suspicious activity to the IDS, n_R. Based on their reports, version and rank attacks are detected by the IDS using DES implemented at the root. The agents are event driven and perform minimally at leaf level in the monitored RPL-IoT network. They have no extra duties other than sensing suspicious activity and reporting. On receipt of an irregular DIO, the piggybacked information is obfuscated and reported to n_R. To prevent an attacker from profiling, the agents send the intimation packet with a random delay. Function of this component is explained using Algorithm 1. On receipt of a DIO packet *DIORQP* with an increased version number, an agent node n_j reports an intimation packet to n_R. If the DIO is a trickle timer update, with an used version number and incremented rank value, the DIO is marked suspicious. A DIO is also marked suspicious if update is trickle timer inconsistent with an increased rank value. Information regarding such DIO receipts are also reported to n_R. Active probing and diagnosis phases are demonstrated through the RQST_RSP_Handler and DES diagnoser, respectively, in the following subsections.

E. RQST_RSP_HANDLER()

The working our algorithm is described as follows. The input it takes are:

Algorithm 1 Agent Intimation Procedure

```
   Local Variables: rank, currVerNum
   Input: Received DIO packet DIORQP
   Output: Intimate received DIO packets DIOINMP,
           DIOvINMP
1  if (ipd(DIS) = ips(DIORQP)) and
    (macd(DIS) = macs(DIORQP)) then
2      if verNo(DIORQP) > currVerNum then
3          Send DIO receipt intimation DIOvINMP to
            n_R;
4      end
5      if DIORQP is Trickle Inconsistent then
6          if rank(DIORQP) > rank then
7              Send DIO receipt intimation DIOINMP to
                n_R;
8          end
9      end
10     else if DIORQP is Trickle Consistent then
11         if verNo(DIORQP) = currVerNum and
            rank(DIORQP) > rank then
12             Send DIO receipt intimation DIOINMP to
                n_R;
13         end
14     end
15 end
```

- DIO intimation packets that are reported from agents on receipt of irregular DIO packets.
- Probe request packets from the buffer that are yet to be sent (this becomes possible as RQST_RSP_HANDLER() is part of the modified RPL).
- Probe response packets.
- TEST_FLAG indicates when to detect and identify the attack by sending probe packets to intended nodes.

If the values Δ_{max} and Δ_a, have been computed, the diagnoser sets TEST_FLAG = 1 (Line 1). The two values are pre-computed during non-attack condition in the RPL instance in use as discussed in Section IV-C. The handler outputs events, namely, *PRQDP*, *PRSDP*, *DIOINMP*, *DIOvINMP*, *PR_TO*, *PRSDP**, *URDES*, which are all passed to the DES diagnoser. The model variables used are $c1$, *flag*, Δ_{max}, Δ_a, j, *lastSend*, *rtd* and *rch*. They are shared among the handler and the DES diagnoser. When the TEST_FLAG is set by the diagnoser, it means that the attack detection and identification phase can be started. The algorithm is now explained step-wise. The DES diagnoser gets executed and remains so till the DODAG remains operational. Diagnoser sets the TEST_FLAG = 1 which is its initial transition.

If a version update intimation is reported, it checks if TEST_FLAG = 1 (Line 3). The event *DIOvINMP* is sent to the diagnoser (Line 4). Diagnoser sets TEST_FLAG = 0 until a decision on the version inconsistency is made. If an irregular rank update is intimated, the event *DIOINMP* is passed to the diagnoser (Line 8). Model variable j stores the index of the

Algorithm 2 RQST_RSP_HANDLER()

```
   Data: c1, ver, rcvd, flag = FALSE, Δ_max, Δ_a, lastSend, rtd, j, rch
   Input: DIO intimation packets, Probe response packets, TEST_FLAG
   Output: Events: PRQDP, DIOINMP, DIOvINMP, PRSDP, PR_TO,
           PRSDP*, URDES
1  while Δ_max and Δ_a are not NULL do
2      if Version update is reported then
3          while (TEST_FLAG == 1) do
4              Generate event DIOvINMP;
5          end
6      end
7      if Rank update is reported then
8          Generate event DIOINMP;
9          j ← {i | n_i.IP == DIOINMP_IPS};
10         Generate event PRQDP;
11         Send ICMPv6 probe packet to TPATH^j[0] via stored downward
            route R;
12         Start clock timer c1();
13         lastSend = 0;
14     end
15     if Received packet is a probe response then
16         rtd ← TPATH^j[lastSend].RTT;
17         Increment lastSend;
18         if (c1() ≤ rtd + Δ_a) then
19             Generate event PRSDP;
20             rch ← lastSend − 1;
21             Stop clock timer c1();
22             Generate event PRQDP;
23             Send ICMPv6 probe packet to TPATH^j[lastSend] via stored
                downward route R;
24             Start clock timer c1();
25         end
26         else if (c1() > rtd + Δ_a) then
27             if (flag == FALSE) then
28                 Generate event PRSDP*;
29                 Stop clock timer c1();
30             end
31             else if (flag == TRUE) then
32                 Generate event PRSDP*;
33                 Stop clock timer c1();
34                 Generate event PRQDP;
35                 Send ICMPv6 probe packet to TPATH^j[rch] via DAO
                    advertised downward route R';
36                 Start clock timer c1();
37                 flag = FALSE;
38             end
39         end
40     end
41     if (c1() > Δ_max) AND (No response packet is received) then
42         Generate event PR_TO;
43         Stop clock timer c1();
44         Increment lastSend;
45         if (flag == TRUE) then
46             Generate event PRQDP;
47             Send ICMPv6 probe packet to TPATH^j[lastSend] via DAO
                advertised downward route R';
48             Start clock timer c1();
49         end
50         else if (flag == FALSE) AND (lastSend < TPATH^j_SIZE) then
51             Generate event PRQDP;
52             Send ICMPv6 probe packet to TPATH^j[lastSend] via stored
                downward route R;
53             Start clock timer c1();
54         end
55         else if (flag == FALSE) AND (lastSend == TPATH^j_SIZE) then
56             Generate event PRQDP;
57             Send ICMPv6 probe packet to TPATH^j[lastSend] via DAO
                advertised downward route R';
58             Start clock timer c1();
59             flag = TRUE;
60         end
61     end
62 end
```

TPATH array used. The variable is shared with the diagnoser (Line 9). *PRQDP* event is passed to the diagnoser and a probe packet is sent to the node at 1-hop distance from the root in the table $TPATH^j$ (Line 11). $TPATH^j$ stores a saved route R for agent node n_j. *lastSend* stores the index of the node in $TPATH^j$ to which the last probe request packet is sent. A clock timer is started to maintain a record of the transmission time of the packet that can be uniquely identified using a transaction identifier value, *transid*.

The module described through lines 15 to 37 is taken on receipt of a probe response packet. Variable *rtd* is set to the round-trip delay of the node to which the probe packet was last sent. The variable *lastSend* is incremented (Line 16). The total response time it takes for a particular node is computed using the clock variable, $c1$ and is compared against a pre-computed RTT (old). We use Δ_a to characterise the admissible delay while awaiting a probe response. In case a response packet is not received at n_R after a Δ_a time period beyond the expected RTT, we consider it as delayed response. If $c1$ does not exceed $rtd + \Delta_a$, the generated event *PRSDP* is passed to the diagnoser (Line 18). The variable *rch* is set to point to the last node whose packet is received before delay timeout occurs (Line 19). The clock timer is then stopped and another request packet is sent to a subsequent node (Line 21). Consequently, the event *PRQDP* is passed to the diagnoser. Clock timer is restarted to count the RTT via the stored route (Line 23). If $c1$ exceeds $rtd + \Delta_a$, then a *flag* variable is checked (Line 25). It is set equal to FALSE during the algorithm initialization. In case *flag* = FALSE and TEST_FLAG = 1, a delayed response received event *PRSDP** is passed to the diagnoser which sets it to 0 (Line 27). The clock timer is stopped. On the other hand, if *flag* is TRUE and a probe response packet is received from some node, suppose x, beyond $rtd + \Delta_a$, then the event *PRSDP** is generated and passed to the diagnoser and clock timer stopped (Line 32). A request packet is sent via current downward route R' to node x, clock timer is restarted and *flag* is set to FALSE (Lines 33-36).

The module described through lines 40 to 59 checks if $c1$ counts beyond a maximum probe timeout period and no response packet is received at n_R. We use Δ_{max} to characterise the maximum delay after next probe request is made. Consequently, a probe timeout event generated here is *PR_TO* which is passed to the DES diagnoser while the clock timer is stopped and *lastSend* is incremented by 1 (Line 42). Three conditions over the variables *flag* and *lastSend* are checked if they are met. If *flag* is determined to hold TRUE, then event *PRQDP* is passed to the diagnoser and an ICMPv6 probe packet is sent to $TPATH^j[lastSend]$ via a current downward route R' and clock timer $c1$ is started (Lines 45-47). On the other hand, if *flag* is found to be false while *lastSend* is less than the size of $TPATH^j$, then event *PRQDP* is passed to the diagnoser and an ICMPv6 probe packet is sent to $TPATH^j[lastSend]$ via the stored downward route R and clock timer $c1$ is started (Lines 50-52). If *flag* is found to be false while *lastSend* equals the size of $TPATH^j$, then event *PRQDP* is passed to the diagnoser and an ICMPv6 probe packet is sent to $TPATH^j[lastSend]$ via a current downward route R', clock timer $c1$ is started and variable *flag* is set to TRUE (Lines 55-58).

TABLE 3. List of symbols.

Symbol	Definition
H	DES model
Σ	Set of events of the DES model H
Σ_m	Set of measurable events of the DES model H
Σ_{um}	Set of unmeasurable events of the DES model H
V	Set of model variables of the DES model H
$\Im$	Set of transitions of the DES model H
τ	A transition $\tau \in \Im$
Y	Set of states of the DES model H
Y_0	Set of initial states of the DES model H
σ	Event on which a transition is enabled
$check(V)$	Condition(s) on a subset of model variables, V
$assign(V)$	Assignment(s) on a subset of model variables, V
$L(H)$	Set of all traces *generated* in H
A_i	i^{th} attacker
Y_N	Set of normal states of the DES model H
Y_F	Set of faulty states of the DES model H for fault type F
Y_{A_i}	Set of attacker states of the DES model H for attacker A_i
σ_{A_i}	Event corresponding to attack launched by attacker A_i
O	Diagnoser of DES H
Z	Set of nodes of the diagnoser, O, also called O-nodes
Z_0	Set of initial nodes of the diagnoser, O
A	Set of transitions of the diagnoser, O, also called O-transitions

F. DES MODEL AND DIAGNOSER

The DES modeling (see **Appendix VI**) of the IoT-RPL network is demonstrated here. The principle of detection and identification by the diagnoser is discussed. We later show that an attacker, if present, is correctly located in the DODAG.

Assumptions in the normal condition: After receiving an intimation from an agent n_j, a node is sent probe request packet along $TPATH^j$. Subsequent probes are then sent depending upon the measured RTT. During the normal condition, two cases can arise here. (i) While awaiting a probe RSP packet, destination unreachable message is received. (ii) After the rank update intimation is received, if a RSP packet is received after the delay timeout period, for a probe packet sent via current DAO advertised downward route. Both of these cases can occur due to a local repair operation and has been modeled as a normal DES.

Assumptions in the attack condition: In the presence of an attacker advertising illegitimate rank or version values, inconsistencies occur in the upward and downward routes. As a result, two cases can arise here as well. (i) Version inconsistency is intimated by agent node. (ii) A probe request packet sent to a child node of the attacker node along $TPATH^j$ (considering the reporting agent node to be n_j) responses with delay. Given an attack behavior due to node A, the above cases are modeled as attacker A type DES model. Since attacker can be located at multiple positions in the DODAG, there are multiple attacker type models. The diagnoser is constructed from the DES models. In both of these cases, since the diagnosability condition is satisfied each time because there are no uncertain states, an attacker location is identified. The

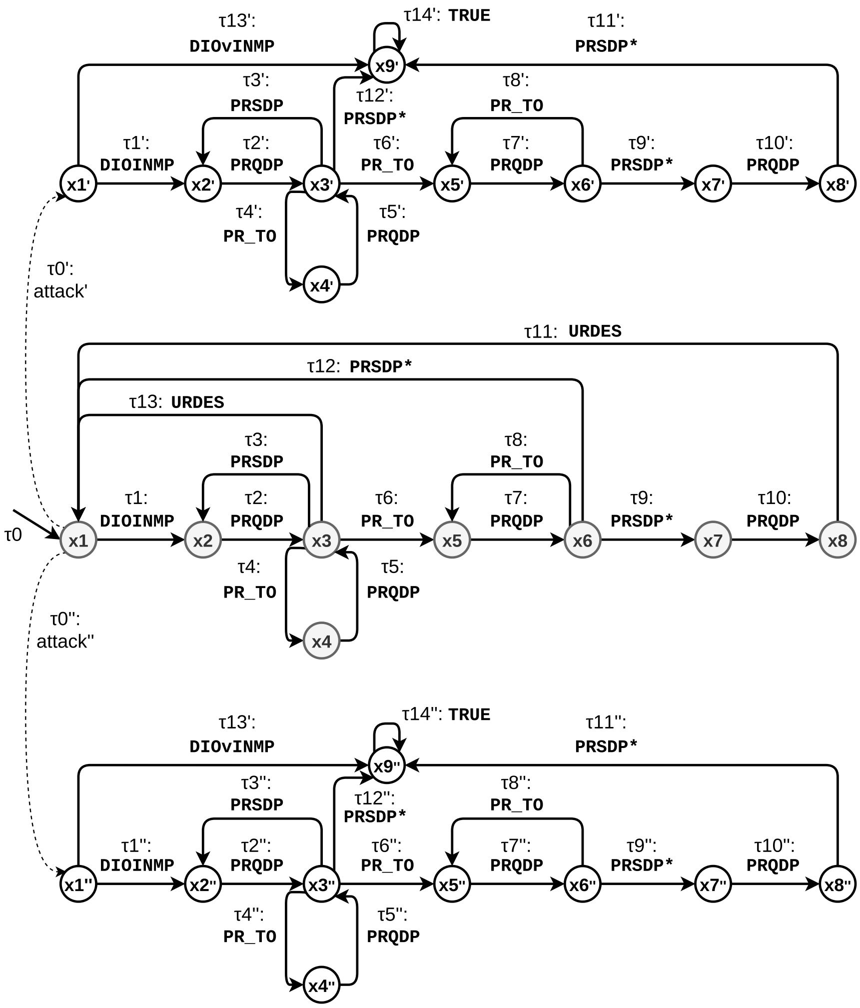


FIGURE 7. **DES model *H*.**

attack as well as the attacker type behavior are different from the normal or other attacker type behavior, respectively.

We consider the system model of a network consisting of resource constrained IoT nodes arranged in a RPL DODAG. The notations used and their definitions are listed in Table 3. The DES model which has been used to represent the *Probe Request Response* sequence during normal and rank or version attack conditions is drawn using Figure 7. The various components of the DES model $H = \langle X, X_0, \Gamma, V, C, \Sigma \rangle$ for the *Request Response* sequence after an irregular DIO intimation is received are discussed.

The state set X with initial set of states X_0 ($X_0 \subseteq X$) symbolise the control states of the RQST_RSP_HANDLER() component of the IDS. The normal DES model states and attacker type model states together constitute the state set $X = \{x1, x2, \ldots, x8, x1', x2', \ldots, x9', x1'', x2'', \ldots, x9''\}$. In our model, the set of model variables, $V = \{ips, ipd, transid, j, flag, lastSend, rtd, ver, rch, \{ips_1, ips_2, \ldots, ips_t\}\}$. The model variables correspond to program and data variables that are internal to the IDS. Certain program variables are designated as the clock variables, C which are absolute values of clock timer that can is SET and RESET using commands. In real-time applications, timing constraints are expressed by satisfying the conditions on the clock variables. We use a single clock variable in the set of clock variables, i.e., $C = \{c1\}$. Event set Σ contains the packet communication events. In our model, the set of events, $\Sigma = \{$*DIOINMP, DIOvINMP, URDES, PRQDP, PRSDP, PR_TO, PRSDP*, attack$', $attack$''\}$. A transition is enabled if the conditions are satisfied and is said to be taken on the occurrence of the associated event. The transitions set Γ consists of transitions $\{\tau 0, \tau 1, \ldots, \tau 13, \tau 1', \tau 2', \ldots, \tau 14', \tau 1'', \tau 2'', \ldots, \tau 14''\}$.

Considering that there is one attack node among n nodes, i.e., $\{A_1, A_2, \ldots, A_n\}$, in the IoT network, the state set, X, can be partitioned into disjoint sets $X_N, X_{A_1}, X_{A_2}, \ldots, X_{A_n}$, where, X_N represents the set of states belonging to the normal behavior of the network, while states of the form X_{A_i}, $1 \leq i \leq n$, $i \in \mathcal{N}$, represent the behavior of the network if A_i is the attack node. For simplicity, we model using 2 nodes, A_1 and A_2, among which one is an attack node, hence

$X = X_N \cup X_{A_1} \cup X_{A_2}$. In Figure 7, the non-primed states are the states when the system behaves normally while the single and double primed states represent the system under attack by the nodes A_1 and A_2, respectively. The events of the system is disjoint union of measurable events and unmeasurable events Σ_m and Σ_{um}.

1) DES BEHAVIOR UNDER NORMAL CIRCUMSTANCES

The behavior of H under normal circumstances is shown in Figure 7. The system, when functioning normally, is represented using the states $\{x1, x2, \ldots, x8\}$ and the transitions $\{\tau 0, \tau 1, \ldots, \tau 13\}$. The initial state of X_0 is $x1$. We next discuss the transitions in normal condition as follows:

- $\tau 0$, the initial transition leads to the initial state $x1$ as shown in Figure 7. It is assumed while modeling that the constant timeout values, Δ_{max} and Δ_a, have been computed and then $\tau 0$ takes place. There is no explicit event that triggers $\tau 0$. Occurrence of $\tau 0$ implies that the DES model is invoked when the timeout values are both not NULL. Table 4 shows $initial(\tau 0) = --$, i.e., there are no initial states and $final(\tau 0) = x1$. $\sigma = TRUE$ means that transition $\tau 0$ is always enabled and $x1$ is automatically reached at the start of the model. $check(V) = --$ implies that no condition over the model variables are checked and the condition is always satisfiable for the transition. Value 1 is assigned to variable TEST_FLAG as implied by $Assign(V) = \{TEST\_FLAG \leftarrow 1\}$, which in turn means that the detection of rank attacker can be started.
- $\tau 1 : (x1 \rightarrow x2)$ - *DIOINMP* : Since we model the rank attack scenario, the focus remains on DIO updates across the DODAG. So when the model is started and the current state is at $x1$, inconsistent DIO reports are looked into and is modeled using the transition $\tau 1$. Here, $initial(\tau 1) = x1$ and $final(\tau 1) = x2$. $\sigma = DIOINMP$ implies that transition $\tau 1$ is enabled when RQST_RSP_HANDLER() generates event *DIOINMP* (i.e., after an inconsistent rank update is reported from an agent). $check(V) = \{ips_j = DIOINMP_{IPS}, ipd = DIOINMP_{IPD}\}$ and $Assign(V) = --$. The parameters that validate a DIO packet intimation from an agent are source and destination IP. It is checked if the parameters equal the value stored in the model variables, ips_j and ipd, both of which are initialized to hold the IP address of agent node n_j and n_R, respectively, at the model start.
- $\tau 2 : (x2 \rightarrow x3)$ - *PRQDP* : At state $x2$, the transition $\tau 2$ implies that a probe request ICMPv6 packet is sent. $\sigma = PRQDP$ implies that $\tau 2$ is enabled when the RQST_RSP_HANDLER() generates the event *PRQDP* (i.e., after a RQST packet is sent). $check(V) = --$ meaning that no condition need to be satisfied and $Assign(V) = \{ips \leftarrow PRQDP_{IPS}, ipd \leftarrow PRQDP_{IPD}, transid \leftarrow PRQDP_{TRANSID}, TEST\_FLAG \leftarrow 0, lastSend \leftarrow lastSend + 1\}$. The parameters that uniquely identify a probe RQST packet are source IP, destination IP and a transaction identifier. Consequently, all the parameters that correspond to the RQST packet that is sent are stored in the model variables, *ips*, *ipd* and *transid*. TEST_FLAG is set to 0 such that no new probe packets are to be sent until a decision on normal or rank attacker can be ascertained. The model variable *lastSend* is incremented, keeping a note of the number of probe packets that are sent. The destination IP of the probe request packet, i.e., $PRQDP_{IPD}$, is the first IP address that is looked up in the table $TPATH^j$. The clock variable $c1$ is RESET to make note of the transmission time of the sent RQST packet.
- $\tau 3 : (x3 \leftarrow x2)$ - *PRSDP* : At state $x3$, the transition $\tau 3$ implies that a probe RSP packet has arrived from a node for some sent RQST packet. Here, $initial(\tau 3) = x3$ and $final(\tau 3) = x2$. $\sigma = PRSDP$ corresponds to enabling transition $\tau 3$ after the RQST_RSP_HANDLER() generates the event *PRSDP* implying that a probe RSP packet has arrived and the condition on the model variables in $check(V)$ are satisfied. $check(V) = \{ips = PRSDP_{IPD}, ipd = PRSDP_{IPS}, transid = PRSDP_{TRANSID}\}$. The conditions over the model variables, *ips*, *ipd* and *transid*, ensure that the RSP packet is a response to the probe request packet sent in $\tau 2$. $Assign(V) = \{TEST\_FLAG \leftarrow 1, rch \leftarrow PRSDP_{IPS}\}$. TEST_FLAG is set to 1 meaning that rank attacker detection can be started. The model variable *rch* holds the IP address of the latest node that responses to the probe packet before the delay timeout period is over, which again is ensured if the condition over $c1$, $\Phi(c1) = \{c1 < ipd.RTT + \Delta_a\}$, is satisfied.
- $\tau 4 : (x3 \rightarrow x4)$ - *PR_TO* : At state $x3$, the transition $\tau 4$ corresponds to probe timeout period being reached while waiting for a probe RSP packet for a probe RQST packet sent. $\sigma = PR\_TO$ implies that the transition $\tau 4$ is enabled when the RQST_RSP_HANDLER() generates the event *PR_TO*. $check(V) = \{lastSend < M^j\}$, $Assign(V) = \{TEST\_FLAG \leftarrow 1\}$ and $\Phi(c1) = \{c1 \geq \Delta_{max}\}$. The condition over the model variable *lastSend* ensures that the number of probes sent is lesser than the size of $TPATH^j$. TEST_FLAG is set to 1 meaning that rank attacker detection can be started. The condition over $c1$ ensures that it exceeds the probe timeout period.
- $\tau 11$ $(x8 \rightarrow x1)$ - *URDES* : At state $x8$, the transition $\tau 11$ implies that a destination unreachable message is received in response to a probe packet sent from n_R to the last reachable node along the current DAO advertised downward route. It rules out the presence of any loop created. $\sigma = URDES$ implies that the transition is enabled when the RQST_RSP_HANDLER() generates the event *URDES*. $check(V) = \{ips = URDES_{IPD}, transid = URDES_{TRANSID}\}$. The condition check on the model variables *ips* and *transid* are used to ensure that the destination unreachability packet is a reply to the probe request packet sent

TABLE 4. Transitions $\Im$ in H corresponding to network packet frames.

$\mathbf{E}vent(\sigma)$	Transition	$\phi(V)$	Assign(V)	$\phi(C)$	Reset(C)
DIOINMP	$\langle x1, x2\rangle,\langle x1', x2'\rangle,\langle x1'', x2''\rangle$	$ips_j \equiv DIOINMP_{IPS}$	-	-	-
		$ipd \equiv DIOINMP_{IPD}$	-	-	-
DIOvINMP	$\langle x1', x9'\rangle,\langle x1'', x9''\rangle$	$ips_j \equiv DIOvINMP_{IPS}$	-	-	-
		$ipd \equiv DIOvINMP_{IPD}$	-	-	-
		$ver < DIOvINMP_{VERNUM}$	-	-	-
PRQDP	$\langle x2, x3\rangle,\langle x2', x3'\rangle,\langle x2'', x3''\rangle$	-	$ips \leftarrow PRQDP_{IPS}$	-	-
	$\langle x4, x3\rangle,\langle x4', x3'\rangle,\langle x4'', x3''\rangle$	-	$ipd \leftarrow PRQDP_{IPD}$	-	-
		-	$transid \leftarrow PRQDP_{TRANSID}$	-	-
		-	$TEST\_FLAG \leftarrow 0$	-	-
		-	$lastSend \leftarrow lastSend + 1$	-	$c1 \leftarrow 0$
PRQDP	$\langle x7, x8\rangle,\langle x7', x8'\rangle,\langle x7'', x8''\rangle$	-	$ips \leftarrow PRQDP_{IPS}$	-	-
		-	$ipd \leftarrow PRQDP_{IPD}$	-	-
		-	$transid \leftarrow PRQDP_{TRANSID}$	-	-
		-	$TEST\_FLAG \leftarrow 0$	-	-
		-	$flag \equiv FALSE$	-	$c1 \leftarrow 0$
PRQDP	$\langle x5, x6\rangle,\langle x5', x6'\rangle,\langle x5'', x6''\rangle$	-	$ips \leftarrow PRQDP_{IPS}$	-	-
		-	$ipd \leftarrow PRQDP_{IPD}$	-	-
		-	$transid \leftarrow PRQDP_{TRANSID}$	-	-
		-	$flag \equiv TRUE$	-	$c1 \leftarrow 0$
PRSDP	$\langle x3, x2\rangle,\langle x3', x2'\rangle,\langle x3'', x2''\rangle$	$ips \equiv PRSDP_{IPD}$	-	-	-
		$ipd \equiv PRSDP_{IPS}$	$TEST\_FLAG \leftarrow 1$	-	-
		$transid \equiv PRSDP_{TRANSID}$	$rch \leftarrow PRSDP_{IPS}$	$c1 < ipd.RTT + \Delta_a$	-
PR_TO	$\langle x6, x5\rangle,\langle x6', x5'\rangle,\langle x6'', x5''\rangle$	-	-	$c1 \geq \Delta_{max}$	-
PR_TO	$\langle x3, x4\rangle,\langle x3', x4'\rangle,\langle x3'', x4''\rangle$	$lastSend < M^j$	$TEST\_FLAG \leftarrow 1$	$c1 \geq \Delta_{max}$	-
PR_TO	$\langle x3, x5\rangle,\langle x3', x5'\rangle,\langle x3'', x5''\rangle$	$lastSend = M^j$	$TEST\_FLAG \leftarrow 1$	$c1 \geq \Delta_{max}$	-
PRSDP*	$\langle x6, x7\rangle,\langle x6', x7'\rangle,\langle x6'', x7''\rangle$	$ips \equiv PRSDP^*_{IPD}$	-	-	-
		$ipd \equiv PRSDP^*_{IPS}$	-	-	-
		$transid \equiv PRSDP^*_{TRANSID}$	-	-	-
		$rch! = NULL$	-	-	-
		$flag \equiv TRUE$	-	$c1 \geq ipd.RTT + \Delta_a$	-
PRSDP*	$\langle x6, x1\rangle$	$ips \equiv PRSDP^*_{IPD}$	-	-	-
		$ipd \equiv PRSDP^*_{IPS}$	-	-	-
		$transid \equiv PRSDP^*_{TRANSID}$	-	-	-
		$rch = NULL$	-	-	-
		$flag \equiv TRUE$	-	$c1 \geq ipd.RTT + \Delta_a$	-
PRSDP*	$\langle x3', x9'\rangle,\langle x3'', x9''\rangle$	$ips \equiv PRSDP^*_{IPD}$	-	-	-
	$\langle x8', x9'\rangle,\langle x8'', x9''\rangle$	$ipd \equiv PRSDP^*_{IPS}$	-	-	-
		$transid \equiv PRSDP^*_{TRANSID}$	-	-	-
		$rch \equiv nip'$	-	-	-
		$flag \equiv FALSE$	$TEST\_FLAG \leftarrow 1$	$c1 \geq ipd.RTT + \Delta_a$	-
URDES	$\langle x3, x1\rangle,\langle x8, x1\rangle$	$ips \equiv URDES_{IPD}$	-	-	-
		$transid \equiv URDES_{TRANSID}$	$TEST\_FLAG \leftarrow 1$	-	-
attack'	$\langle x1, x1'\rangle$	-	-	-	-

in $\tau 10$. *Assign*(V) makes $TEST\_FLAG = 1$ which means that the attack detection phase can restart, i.e., RQST_RSP_HANDLER() can again receive inconsistent DIO version or rank updates from agents.

2) DES BEHAVIOR UNDER ATTACK CIRCUMSTANCES

The DES model under rank or version attack condition launched by attacker A_1 is shown using the states in $X_{A_1} = \{x1', x2', \ldots, x9'\}$ and transitions, $\{\tau 1', \tau 2', \ldots, \tau 14'\}$. Similarly for attacker type A_2, states and transitions are represented using double prime notation, $X_{A_2} = \{x1'', x2'', \ldots, x9''\}$ and transitions, $\{\tau 1'', \tau 2'', \ldots, \tau 14''\}$ as shown in Figure 7. The DES model behavior under different attackers are mostly identical except a few transitions that differentiate them which are discussed.

- At state $x1$, the system reaches an attacker type state $x1'$ or $x1''$ following an unmeasurable attack transition $\tau 0'$ or $\tau 0''$, respectively.
- $\tau 11'$ $(x8' \rightarrow x9')$ - *PRSDP**: At state $x8'$, the transition $\tau 11'$ corresponds to probe RSP packet that is received beyond the maximum 1-hop delay, i.e., ipd.RTT + Δ_a for a sent probe request packet. σ = *PRSDP** implies that the transition is enabled when the RQST_RSP_HANDLER() generates the event *PRSDP**. $check(V) = \{ips = PRSDP^*_{IPD}, ipd = PRSDP^*_{IPS}, transid = PRSDP^*_{TRANSID}, flag = FALSE, rch = nip'\}$. The conditions over the model variables, *ips*, *ipd* and *transid*, ensure that the RSP packet is a response to the probe request packet sent in $\tau 10'$. The condition over variable *flag* ensures that it is set to FALSE. The model variable *rch* holds the IP address of the last node that replied to the probe packet before the delay timeout period was over. $\tau 11'$ ensures that *rch* holds the IP address of attacker node A_1. A probe response beyond the delay period for probe packet meant for a node with IP address stored in *rch* via the currently advertised DAO route R' is a rank attack. $Assign(V) = \{TEST\_FLAG \leftarrow 1\}$. TEST_FLAG is set to 1 meaning that rank attacker detection can be started. $\Phi(c1) = \{c1 \geq ipd.RTT + \Delta_a\}$ means that $c1$ exceeds the delay timeout period.
- $\tau 13'$ $(x1' \rightarrow x9')$ - *DIOvINMP* : At state $x1'$, the transition $\tau 13'$ corresponds to the receipt of DIO

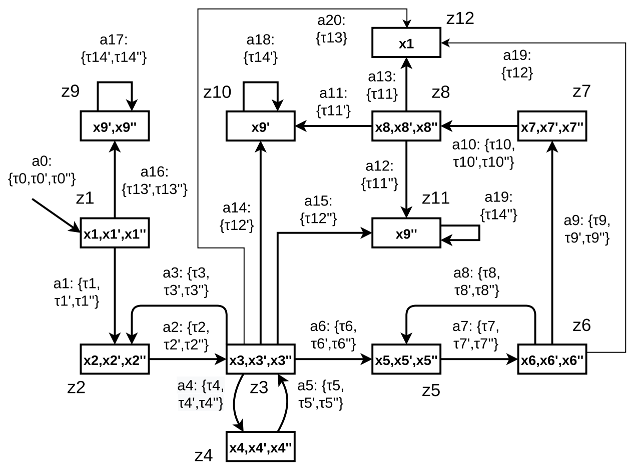


FIGURE 8. **Diagnoser *O* for DES model *H*.**

version inconsistent intimation from an agent leaf node. $\sigma = DIOvINMP$ implies that the transition is enabled when the RQST_RSP_HANDLER() generates the event $DIOvINMP$. $check(V) = \{ver < DIOvINMP_{VERNUM}, ips_j = DIOvINMP_{IPS}, ipd = DIOvINMP_{IPD}\}$ and $Assign(V) = --$. The parameters that validate a DIO packet intimation from an agent are source and destination IP. It is checked if the parameters equal the value stored in the model variables, ips_j and ipd, both of which are initialized to hold the IP address of agent node n_j and n_R, respectively, at the model start. The model variable ver stores the latest version number advertised. The condition over ver ensures that it is lesser than the DIO version number reported by the source agent node. It may be noted that a DIO broadcast in the DODAG with a version number higher than already advertised by the DODAG root is a version number attack.

3) DIAGNOSER

The DES diagnoser is basically an observer automaton. Given a measurable trace executed on the model, the diagnoser gives an estimate of membership of the current system state in the model among normal or any attacker type state from H. An alert is generated when it can be ascertained that the current state belongs to an attacker type. It is also notified in case it belongs to a set of attacker types.

We use a representation in directed graphs for our DES diagnoser $O =< Z, A, Z_0 >$, where Z is the set of diagnoser states, referred henceforth as *O-states*, Z_0 is the set of initial O-states of the diagnoser and A, the set of diagnoser transitions, also referred as *O-transitions*, $A \subseteq Z \times Z$. The sets we consider are finite sets. During diagnoser automaton construction, transitions and states are appended to the diagnoser based on measurable system traces from the initial set of states. Depending on source state of a transition, destination state of a transition and the equivalence relation that each of them share with other source states and destination states, respectively, an O-transition can be in one these following forms: (1) $\langle(x_a, x_a^+), (x_b, x_b^+)\rangle$ if $\langle x_a, x_b, \sigma_a, \phi_a(V), \Phi_a(C), Assign_a(V), Reset_a(C)\rangle \equiv \langle x_a^+, x_b^+, \sigma_{a^+}, \phi_{a^+}(V), \Phi_{a^+}(C), Assign_{a^+}(V), Reset_{a^+}(C)\rangle$ (2) $\langle(x_a, x_a^+), (x_b)\rangle$, $\langle(x_a, x_a^+), (x_b^+)\rangle$ if $\langle x_a, x_b, \sigma_a, \phi_a(V), \Phi_a(C), Assign_a(V), Reset_a(C)\rangle \not\equiv \langle x_a^+, x_b^+, \sigma_{a^+}, \phi_{a^+}(V), \Phi_{a^+}(C), Assign_{a^+}(V), Reset_{a^+}(C)\rangle$ and $x_a \equiv x_a^+$ (3) $\langle(x_a), (x_b)\rangle\langle(x_a^+), (x_b^+)\rangle$, otherwise.

The set of states contained in an initial O-state are the initial states of DES H and the states that are reachable from each of those initial states using sequences of unmeasurable transitions. The initial O-state thus comprises of states that belong to normal state set or any attacker type state from H. Consequently, any O-state may comprise of equivalent states from normal as well as attacker type states. On the other hand, the O-transitions are sets of equivalent transitions between sets of equivalent source and equivalent destination states in H.

Exposition 1 *Normal-certain O-state* : A O-state that consists of states in H, all of which only belong to X_N.

Exposition 2 *Attacker$_i$-certain O-state* : A O-state that consists of states in H, all of which only belong to X_{A_i}.

Exposition 3 *Attack-certain O-state* : A O-state that consists of states in H, all of which only belong to $X_{A_1} \cup X_{A_2}$.

Figure 8 shows the constructed diagnoser for our DES model H, considered in Figure 7. The working mechanism of our diagnoser is summarised here by showing one or more executions of sequences of measured events (transitions) as follows:

1) The initial state of the model H, $x1$, and states $x1'$ and $x1''$ reachable via unmeasurable attack transitions, $\tau0'$ and $\tau0''$, form the initial state of the diagnoser, $z1$.
2) Let $\Im_{z1_1} = \{\tau1, \tau1', \tau1''\}$, i.e., the outgoing transitions from model states $\{x1, x1', x1''\} \in z1$. All the transitions in $\Im_{z1_1}$ are equivalent and hence cannot be further subdivided and hence justifies O-transition $a1$. The O-state corresponding to the transition $a1$ is $z2 = \{x2, x2', x2''\}$.
3) Let $\Im_{z1_2} = \{\tau13', \tau13''\}$, i.e., the outgoing transitions from model states $\{x1', x1''\} \in z1$. All the transitions in $\Im_{z1_2}$ are equivalent and hence cannot be partitioned further and hence justifies O-transition $a16$. The O-state corresponding to the transition $a16$ is $z9 = \{x9', x9''\}$. Since, $z9$ consists exclusively of attacker type states only, it is an attack-certain O-state.
4) Let $\Im_{z2} = \{\tau2, \tau2', \tau2''\}$, i.e., the outgoing transitions from model states $\{x2, x2', x2''\} \in z2$. All of outgoing transitions in $\Im_{z2}$ are measurement equivalent belonging to one measurement equivalence class of transitions, hence cannot be further partitioned. Therefore, it justifies O-transition $a2$. The O-state corresponding to the transition $a2$ is $z3 = \{x3, x3', x3''\}$. In a similar manner, the diagnoser states $\{z4, z5, z6, z7\}$ can be constructed using the corresponding O-transitions $\{a4, a6, a7, a9\}$. The principle can be safely extended.
5) From the definition, we can compute the attacker$_i$-certain O-states and the Normal-certain O-states. In our

example, when $i = 1$ the attacker$_1$-certain O-state may be computed as $z10 = \{x9'\}$ since it exclusively consists of states only belonging to attacker 1. Similarly, attacker$_2$-certain O-state may be computed as $z11 = \{x9''\}$ and the normal-certain O-state can be computed as $z12 = \{x1\}$.

G. AN EXAMPLE OF RANK ATTACKER NODE IDENTIFICATION USING DES DIAGNOSER

Suppose the following events occur in the DODAG chronologically due to packets received or sent from the DODAG root: *DIOINMP*, *PRQDP*, *PRSDP*, *PRQDP*, *PRSDP*, *PRQDP*, *PRSDP**.

The diagnoser starts from the O-state $z1$ and on occurrence of the *DIOINMP* event, the diagnoser moves to O-state $z2$ via O-transition $a1$. The transition $a1$ might have been taken by the diagnoser due to the occurrence of any of the H-transitions, $\tau 1$, $\tau 1'$ or $\tau 1''$. Since the transitions $\tau 1$, $\tau 1'$ and $\tau 1''$ are measurement equivalent, it cannot be certainly said at this point if an attack has occurred. A probe request data packet is sent due to which the event *PRQDP* occurs and the diagnoser moves to O-state $z3$ via O-transition $a2$. Now, the response to the probe is received and the event *PRSDP* passed to diagnoser and O-state $z2$ is reached via $a3$. The O-states are then revisited due to the events *PRQDP*, *PRSDP* and *PRQDP* and the diagnoser reaches the O-state $z3$. Eventually, when the *PRSDP** event occurs, suppose the diagnoser moves from O-state $z3$ to O-state $z10 = \{x9'\}$ via O-transition $a14$ due to the model transition $\tau 12'$. Since the O-state $z10$ reached by the diagnoser is an Attacker$_1$-certain O-state, it is ascertained that the system is under attack condition due to attacker node 1. Moreover, since there are no A_i indeterminate cycles [57], [59], along all paths of the DES diagnoser, an unique malicious node i, when present, can be identified correctly. On each such occasion when the diagnoser reaches an Attacker$_i$-certain state due to an event trace, an alert is generated.

H. CORRECTNESS AND COMPLETENESS

DES modeling aids in formalizing a system to check correctness and completeness [58]. We demonstrate correctness and completeness of our proposed IDS here, by taking into consideration all possible cases of rank attack. For each case considered, we show that attacker node is correctly identified. We use the DODAG instance shown in Figure 6 for our proof, where n_R is the 6BR root and the set of agents $\mathcal{T} = \{n_1, n_2, n_3, n_4\}$. B and C are the two suspected rank attack nodes and can be related to nodes A_1 and A_2 used in our DES model. Since there are no A_i-indeterminate cycles in the diagnoser O, therefore the diagnosability condition is satisfied. This means that location of an attacker A_i in the DODAG, having launched a rank or version attack, is always diagnosable. We show using analysis that B or C is correctly identified as attack node when the corresponding attacker-certain state is reached in the diagnoser.

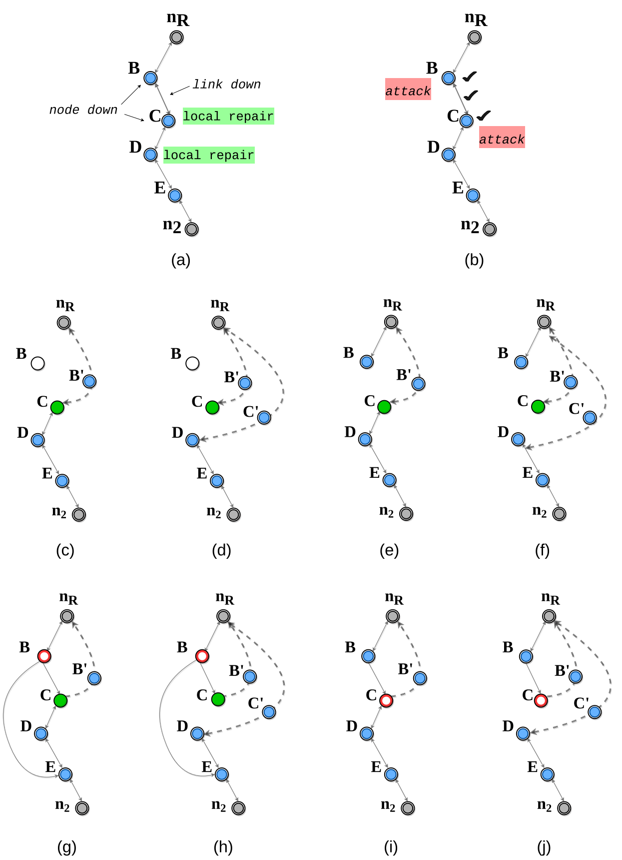


FIGURE 9. Normal and attack configurations.

We now prove the completeness by justifying why all attack cases can be detected from the traces in H. An irregular increased rank advertisement can be classified as a normal network condition if a local repair operation is undertaken, otherwise can be classified as an attack. As shown in Figure 9(a), we assume that nodes C and D undertake local repair operations due to the parent node being down, or link with the parent goes off or as part of loop avoidance. On the other hand, as shown in Figure 9(b), an attack might have been launched by node B or C. Though, the effects of attack mimics the normal scenario, however, there lies unique inconsistencies in the resulting topologies which can be made out from the probe response characteristics of nodes. We discuss the normal cases here first.

Case I: Node C undertakes local repair due to parent node B being down.

As shown in Figure 9(c) and 9(d), node C chooses alternate parent node B' for upward routing. Depending on the newly advertised rank, a successor node may conform to the update by not changing its preferred parent or may choose a better route instead. It may be noted that since B is down, any upward or downward path between the pairs (n_R, B) and (B, C) cease to exist. Our proposed procedure utilises the above facts. Firstly, n_2 reports the DIO update to n_R. On receipt of such intimation, the diagnoser moves from state

$z1$ to $z2$. Now, a probe RQST packet *PRQDP* is sent to node B via stored downward route $TPATH^2$ while the diagnoser reaches state $z3$. Since no response packets are received, event *PR_TO* is generated and the diagnoser consequently reaches state $z4$. Next, a probe request packet *PRQDP* is sent to C via $TPATH^2$ with the diagnoser reaching state $z3$. Again, no RSP packet is received before Δ_{max} since the request packet itself is not delivered via B. This behavior is repeated for the subsequent probe request packets sent to D and E with the diagnoser reaching state $z4$. Now n_2 is sent the probe request packet and Δ_{max} is again exceeded while waiting for a response. The diagnoser reaches state $z5$ this time, since all the nodes in $TPATH^2$ are probed. Now, a probe packet is sent to the first unreachable node via a currently advertised downward path. Since B is down, no routing information is updated for node B. Since C had chosen a path via B', a downward path from the root exists. On a request packet *PRQDP* being sent to C via B', the diagnoser reaches $z6$. As route through B' is longer, so delay is incurred while receiving the response. As a result, the delay timeout is exceeded. Consequently the diagnoser moves to state $z12$, since no node was reachable without delay prior to C which is a normal-certain O-state. So, a normal condition of local repair in the DODAG is correctly identified.

Case II: Node C undertakes local repair due to link (B, C) going down.

As in the situation discussed in Case I, the sequences of events are similar, except the fact that response from node B arrives before $RTT(B) + \Delta_a$. So, when the diagnoser moves to state $z2$, the model variable *rch* is set. Therefore, at state $z6$, when a delay timeout occurs, the diagnoser reaches state $z7$ instead of $z1$. A probe request packet is then sent to node B via node C along the current DAO advertised route. The diagnoser accordingly moves to state $z8$. A destination unreachable message is then received by n_R, and the diagnoser moves to normal-certain O-state $z12$ and it is ascertained that situation is normal, since a local repair operation was initiated as shown in Figure 9(e) and 9(f).

An attack launched by an attacker can be of the two following types: (i) The attacker illegitimately chooses a parent node that has higher rank, but does not lie in $TPATH^2$ (ii) the attacker illegitimately chooses a parent node that has higher rank, and is a successor node in $TPATH^2$. Type (i) is discussed as case III and type (ii) is discussed as Case IV.

Case III: Node C undertakes local repair due to loop detection while forwarding to B.

While forwarding packet upwards, suppose C detects a loop and initiates a local repair while forwarding through alternate parent node B'. Now, node B might be a direct attacker that chooses a successor node as its parent, fueling a loop creation. In that case, B must be a node in the subtree at C. As in the situation discussed in the normal scenario, B and C are probed. B responds before delay timeout occurs while C is unreachable. All the nodes successor to C are also unreachable. Consequently, the diagnoser node reaches state $z5$ after a probe timeout occurs while a probe packet is sent to the last node n_2. Now, a delay timeout occurs when a probe packet is sent to C via the current downward route. The diagnoser reaches $z8$ following the event *PRQDP*. The only difference arises when node B is sent a RQST packet via B, and a delayed response is received. The event *PRSDP** is generated and the diagnoser reaches state $z10$ depending on the value of the variable *rch*, which is the IP address of B, the last node that replies without delay. It is therefore ascertained that B is an attack node here since it lies in the subtree of node C. As shown in Figure 9(g) and 9(h), the red line indicates that the attacker has chosen E as its parent. If a *URDES* packet is received, the diagnoser again moves to normal-certain O-state $z12$, which is the case shown using the green line indicating the choice of B.

Case IV: Node C is an attack node that does not advertise DAO

In this case, C chooses a different parent in spite of an existing better parent for upward route. This situation is shown using the Figures 9(i) and 9(j). While probing nodes in $TPATH^2$, nodes B and C, both reply to the probe packets and the diagnoser reaches state $z3$ when a *PRQDP* packet is sent to D. Now, if a delay timeout occurs while awaiting the response, the diagnoser reaches state $z11$ depending on the value of the variable *rch* which holds the IP address of C. Consequently, it can be ascertained that the attacker node is C and the diagnoser correctly detects the attack since $z11$ is an A_2-certain node.

So, all the possible cases of attack by specific attacker nodes are analyzed. The diagnoser correctly reports the network condition by identifying the corresponding attacker type states, for each case.

I. OVERHEAD ANALYSIS

The extra communication overhead is added in our detection scheme due to probe requests and generated responses. The overhead is minimum when only 2 probe requests are sufficient to identify the malicious node. Such a scenario occurs if a probe request packet is sent to a node which responses in time and another probe packet sent subsequently to the child of this node is acknowledged beyond the admissible delay. We now discuss the scenario when maximum overhead is incurred in our solution. Suppose probe request packets are sent sequentially to nodes in $TPATH^j$. Now, the node with the lowest rank responses to the probe request in time. For, the subsequent probe requests sent, responses are not generated. Based on the DAO messages received after the IDS is setup, nodes with missing acknowledgements are sent probe requests through alternate routes. Only the node farthest from the root responses with after an admissible delay. Hence, assuming that the height of the tree is equal to the number of nodes in the RPL, n, then a total of $(1 + 2(n-2) + 1) \approx O(n)$ probe requests will be required here (1 for node with lowest rank, $2(n-2)$ for subsequent $(n-2)$ nodes that are probed twice and 1 for confirmation). Considering a balanced tree of n nodes, depth $= \log_k n$, for a branching factor k. In such

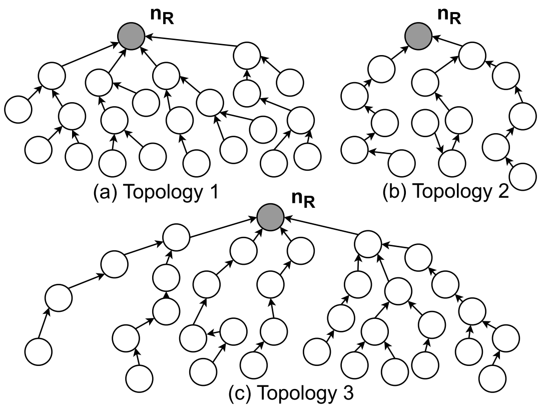


FIGURE 10. Topology considered for testbed and simulation experiments.

cases, the number of probes that will be required in the worst case is $2 \log n \approx O(\log n)$.

V. EXPERIMENTS, RESULTS, AND DISCUSSION

Three experiments are executed in Contiki Cooja [60] and one in a real testbed at FIT IoT-LAB [61]. Cooja is a network simulator explicitly developed to cater for IoT networks while the simulator builds on C base libraries of sensors and RFID chips, the FIT IoT-LAB is an open testbed and comprises of 117 mobile robots and 2728 low-power sensor nodes that are made available for conducting experiments in the heterogeneous environment (e.g., standardized protocol, OS, topologies, and hardware). Having unique hardware and node capabilities, interconnected locations are installed across France in FIT IoT-LAB and made available for experiments via a web portal. We used three different types of topology, as shown in Figure 10. In topology 1, the IoT nodes are distributed very densely, while a sparse distribution is used in topology 2. In topology 3, nodes are distributed in a mixed fashion. Furthermore, the hop count is more in topology 2 as compared to topology 1. We consider a OF0 implementation with hop count (HC) metric. The simulation or experimental parameters of Contiki Cooja and FIT IoT-LAB are presented in Table 5. To examine the performance of our proposed solution, three scenarios are designed as part of the experimental setup, namely, the non-rank attack scenario, increased rank attack scenario, and the increased rank attack scenario with the proposed solution, comprehensive analysis of which are demonstrated below.

A. EXPERIMENT 1: NON-RANK ATTACK SCENARIO

All the external and internal nodes demand the IoT services (i.e., temperature and humidity) using the *Sky-Websense* server. The experiment has been executed on 8, 16, 32, and 64 nodes. The flow of IoT network packets and their behavioural changes are noted. Figure 11a shows an RPL DODAG with 16 nodes. The node having Node ID 65 is the 6BR root running our IDS. Nodes with IDs 16, 13, 30, 52 and 62 are the 5 agents deployed as leaves and behave like regular nodes. *Wireshark* and *power trace* tool are used during simulations for network traffic analysis. In the testbed setup, we have used A8-type nodes utilizing various topologies with Grenoble areas. A8 is a TI SITARA AM3505 (Arm Cortex A8) combined with STM32 microcontroller and a radio interface. It is one of the powerful IoT-LAB node which allows running RIOT, Contiki, and FreeRTOS. The adopted parameters during the testbed experiments are specified in Table 5. Figure 11a shows the DODAG topology in a non-attack scenario. Throughput, energy usage of the network, and the average power consumption on a per-node basis with their respective run times are shown in Figures 12 (a) and (b), analysed using 64 nodes in Contiki Cooja and FIT IoT-LAB, respectively. Our analysis shows average throughput within 86.45% to 94.89%, average network energy usage ranging from 27854 mJ to 33648 mJ, and average power consumption lying within 1.2 mW to 1.46 mW in this scenario. The values are moderately good because during the non-rank attack scenarios, RPL control messages, Objective Function (OF), and Rank computation module are executed correctly with the required number of RPL control messages.

TABLE 5. Contiki Cooja and FIT IoT-LAB experimental parameters.

Parameter name	Value
Operating system	Contiki 3.0, Contiki 4.5
Simulator	Cooja
Testbed	FIT IoT-LAB, Grenoble
Network size	8, 16, 32, 64 nodes
Radio Environment	UDGM
Node Type	Tmote Sky , IoT-Lab A8
Routing Protocol	RPL
MAC/adaptation layer	ContikiMAC/6LoWPAN
Transmitter output power	(dBm) 0 to -25
Receiver sensitivity	(dBm) -94
Radio frequency	2.4 GHz
Attack Modeled	Rank and version number attack
Simulation Duration	Variable

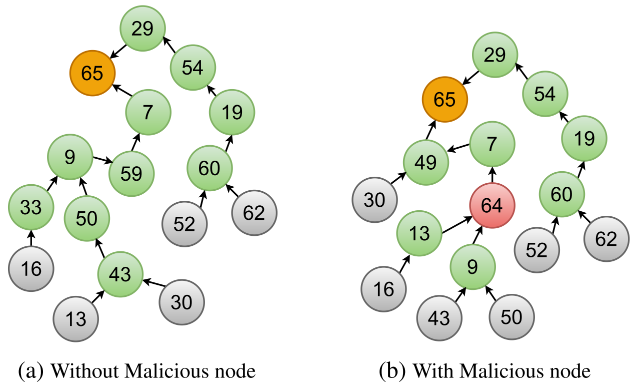


FIGURE 11. DODAG of the IoT ecosystem.

B. EXPERIMENT 2: INCREASED RANK AND VERSION NUMBER ATTACK SCENARIO

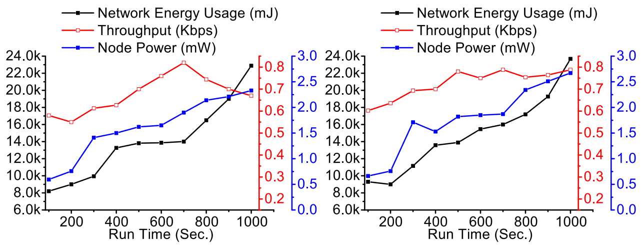
An increased rank attack is performed with 8, 16, 32, and 64 IoT nodes. The attack nodes, incorporated during our experiments, generate malicious RPL control messages

TABLE 6. Energy, Node Power, Throughput, and Packet Delivery Ratio for IoT ecosystem (During attack and after solution implementation in Contiki Cooja).

IoT Scenario	Energy (mJ)				Node Power (mW)				Throughput (Kbps)				Packet Delivery Ratio (%)			
During	**8N**	**16N**	**32N**	**64N**	**8N**	**16N**	**32N**	**64N**	**8N**	**16N**	**32N**	**64N**	**8N**	**16N**	**32N**	**64N**
attack	86615	10216	14425	17098	0.490	60.52	1.351	1.692	0.573	0.596	0.574	0.556	89.17	88.63	86.69	84.61
After solution	**8N**	**16N**	**32N**	**64N**	**8N**	**16N**	**32N**	**64N**	**8N**	**16N**	**32N**	**64N**	**8N**	**16N**	**32N**	**64N**
implementation	8261.4	8898.6	12129.5	16229.5	0.31	0.49	0.92	1.36	0.662	0.661	0.657	0.654	98.76	98.65	98.42	98.34

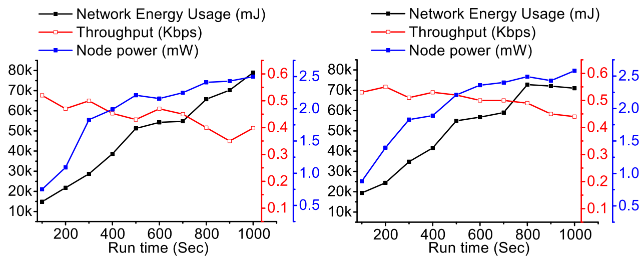
TABLE 7. Energy, Node Power, Throughput, and Packet Delivery Ratio for IoT ecosystem (During attack and after the solution implementation in FIT IoT-Lab).

IoT Scenario	Energy (mJ)				Node Power (mW)				Throughput (Kbps)				Packet Delivery Ratio (%)			
During	**8N**	**16N**	**32N**	**64N**	**8N**	**16N**	**32N**	**64N**	**8N**	**16N**	**32N**	**64N**	**8N**	**16N**	**32N**	**64N**
attack	9527.5	12259.2	17454.3	20176.6	0.59	0.74	1.54	1.81	0.463	0.504	0.487	0.478	80.31	78.11	73.12	73.92
After solution	**8N**	**16N**	**32N**	**64N**	**8N**	**16N**	**32N**	**64N**	**8N**	**16N**	**32N**	**64N**	**8N**	**16N**	**32N**	**64N**
implementation	7269.7	7919.2	10552.2	13307.8	0.27	0.49	0.83	1.19	0.559	0.543	0.572	0.552	91.58	90.88	88.86	87.12



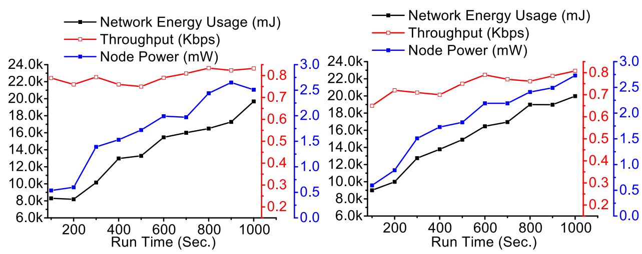
(a) Contiki Cooja (b) FIT-IoT-LAB

FIGURE 12. Average Energy, Throughput, Node Power over run time (nodes=64) (without malicious node).



(a) Contiki Cooja (b) FIT IoT-Lab

FIGURE 13. Average Energy, Throughput, Node Power over run time (nodes=64) (with malicious node).



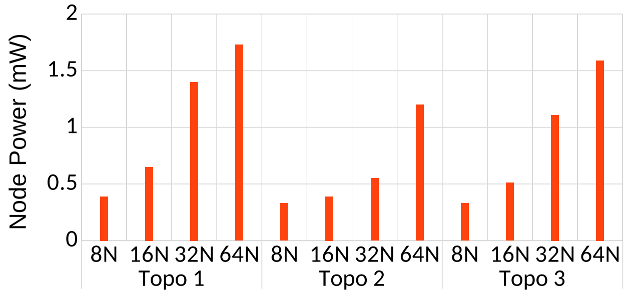
(a) Contiki Cooja (b) FIT-IoT-Lab

FIGURE 14. Average Energy, Throughput, Node Power over run time (nodes=64) (after solution implementation).

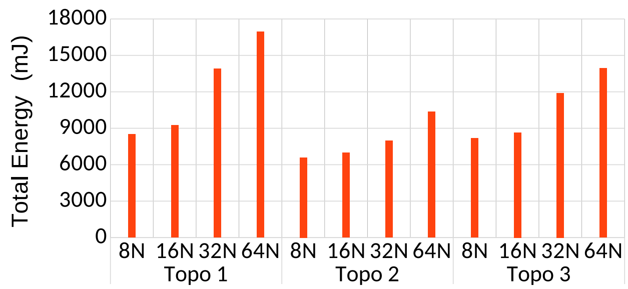
and create falsified non-optimal routes. The IoT network behavioural changes are examined with different malicious nodes while varying node density. Figure 11b shows IDS node at root with ID 65 and the node ID 64 is the malicious node. Among the remaining nodes, nodes with ID 30, 16, 43, 50, 52 and 62 are the agents deployed that perform sensing at the leaf levels. Traffic generated from the attack is analysed using `collect view modules` for analysis purposes in simulation. Consequently, we use `Sysstat` [62] and `iperf` tool [63] for real testbed analysis. We additionally perceive the average power consumption per node, and the energy usage of the complete RPL DODAG. Figure 13 (a) exhibits a considerable increase in the complete network's average energy usage and power consumption per node, i.e., 28.8% to 35.7% and 31.7% to 43.3%, respectively, in Contiki Cooja simulations. Figure 13 (b) shows similar outcomes in FIT IoT-LAB, i.e., 38.7% to 43.9% average energy usage and 36.5% to 52.4% power consumption per node. In both, the throughput graph can be seen to be going down significantly. The average throughput value is reduced and ranges from 37.3% to 43.5% in the attack scenario, both in simulation and real testbed. All experiments show huge network energy and node power consumption with reduced throughput because of a massive number of RPL control messages, malicious OF for routing, and unknown loop formations due to attack. During attack, the performance metrics that significantly affect RPL performance are listed in Tables 6 and 7 for Contiki Cooja and FIT IoT-Lab, respectively. The findings also demonstrate that a rise in the number of IoT nodes results in a significant increase in the amount of malicious RPL control messages, which consumes additional network energy due to node power and consumption. In addition, network performance and packet delivery ratio is shown to suffer and produce inferior outcomes.

C. EXPERIMENT 3: ATTACK SCENARIO WITH PROPOSED SOLUTION

Experiment 2 is executed with the proposed solution, both in simulation and real testbed. The performance of our proposed solution is illustrated in Figure 14. Both during simulation and in real testbed, we have considered 8, 16, 32, and 64 IoT nodes, while the experiments are run for 1000 sec. We consider the values Δ_{max} and Δ_a to be 13 seconds and

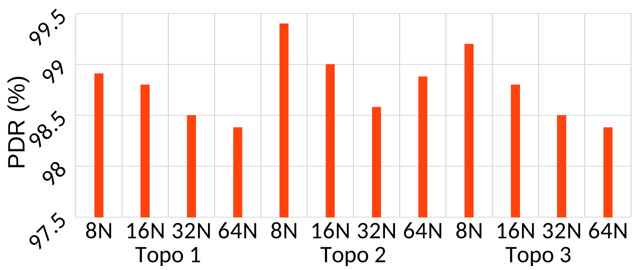


(a) Node Power comparison across topologies

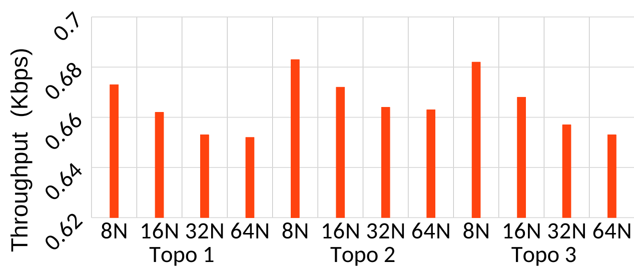


(b) Total energy comparison across topologies

FIGURE 15. Power and Energy for 50 *min* network execution with proposed solution.



(a) PDR comparison across topologies



(b) Throughput comparison across topologies

FIGURE 16. PDR and Throughput for 50 *min* network execution with proposed solution.

3.8 seconds, respectively (discussed in Section IV-C). The trickle timer is of 10 seconds duration. Each experiment is conducted by varying the number of nodes, i.e., from 8 to 64 nodes and hop counts. The performance analysis of all the experiments is based on various metrics like True Positive Rate (TPR) (also known as *sensitivity*), True Negative Rate (TNR) (also known as *specificity*), Accuracy (ACC), Energy usage (EU), Throughput, Packet delivery ratio (PDR), and scalability. The performance analysis metrics are defined as follows:

- *True Positive Rate (TPR)* is the ratio of accurately identified attacker nodes to all of the attacker nodes and is estimated by:

$$TPR = \frac{p}{p+q} \tag{1}$$

- *True Negative Rate (TNR)* is the ratio of wrongly identified genuine nodes to all of the genuine nodes and is estimated by

$$TNR = \frac{r}{r+s} \tag{2}$$

where, *p=Attacker nodes identified accurately q= Attacker nodes not identified correctly r=Genuine nodes identified accurately s=Genuine nodes not identified correctly.*

- *Accuracy (ACC)*: It calculates the overall rates of attacker nodes identification and false alarms. This result signifies the success rate of the proposed approach; it is estimated by

$$ACC = \frac{p+r}{p+q+r+s} \tag{3}$$

- *Energy Usage (EU)*: The amount of energy utilized for the proposed solution throughout its execution.

During the execution of our proposed approach, we consider three topologies, as shown in Figure 10. Figures 15(a) and 15(b) illustrate the node power consumption per node and network energy consumption after our solution is implemented for 50 minutes across various topologies and varying IoT nodes. The findings suggest that our proposed solution has a higher average total energy usage and node power consumption per node in topology 1 in comparison with other topologies and standard RPL with rank and version number attacks in place. When compared to the other possible topologies for this work, topology 2 has a lower average overall energy use and node power per node. The amount of energy consumed is proportional to the density of the individual nodes and DODAG configuration.

Figures 16(a) and 16(b) compare the proposed work's packet delivery ratio and throughput across three topologies with varying IoT nodes. As per the results, our proposed security approach has the lowest throughput (0.652 Kbps) and packet delivery ratio (98.4%) in topology 1 as compared to others. The performance of the suggested technique demonstrates promise in topologies 2 and 3, respectively. Topologies 2 and 3 have throughput of 0.664 Kbps, and 0.653 Kbps and packet delivery ratios of 98.55%, and 98.38%, respectively. Topology 1 has lower results than RPL with rank and version attacks due to packet loss and retransmission.

Figures 14(a) and 14(b) show the performance analysis of our proposed solution during simulation and in real testbed, respectively. A reduction in network energy usage and node power by 24.9% to 33.6% and 22.6% to 41%, respectively, can be noted. Throughput graph can be seen to significantly progressing upwards. The average throughput value was improved by 32.9% to 36.7% on the implementation of our solution in the IoT ecosystem. Tables 6 and 7 present the performance analysis during the recursive execution of our proposed solution across the various possible topologies involving the attack node. Based on the outcome, it can be noticed that different topologies take an unique amount of network energy and node power; it also varies with the number of nodes. It can be further observed that our solution requires minimum amount of network energy and node power. This is not only because we use only one centralized IDS node in our approach, but also because rank and version attacks are detected and identified accurately in lesser time.

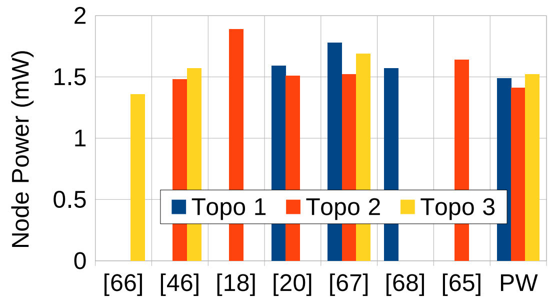


FIGURE 17. Node Power comparison with related works.

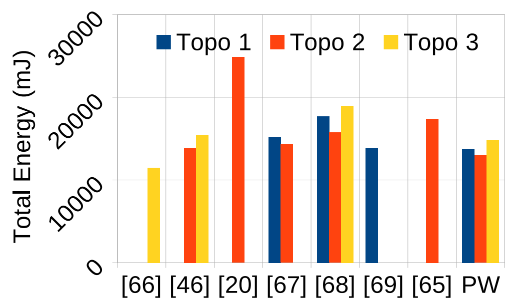


FIGURE 18. Energy comparison with related works.

D. COMPARISON WITH THE EXISTING WORKS

This subsection presents the comparative analysis of the proposed rank and version number attack detection approach with state-of-the-art solutions. Experiments are fairly repeated multiple times to create tight confidence intervals. In general, we compare our real-time testbed results obtained across the different topologies to the simulation results. We observe that both executions provide reliable results (approximately 10% - 30% over/under estimated experimental results). A comparison of our scheme is shown through Table 8 and graphs provided in Figures 17, 18, 19, 20 and 21. To measure the performance metrics, we use `collect view modules`, `Sysstat`, and `iperf tool`. Ten different performance metrics: Energy Usage (EU), Node Power, Throughput (THP), PDR, Control Message Overhead (CONMO), TPR, TNR, ACC (RAD for rank attack detection accuracy, VNAD for version number attack detection accuracy, RAI for rank attack node identification accuracy) and Scalability (SCAL) are considered. State-of-the-art methods [20], [64], [65] consume enormous energy, node power, and control message overhead. Hence they are not as suitable for a constrained IoT ecosystem. Figure 18 shows that our proposed approach takes 13759mJ, 12962mJ, and 14872mJ total energy with the 3 respective topologies. The state-of-the-art methods [18], [20], [40], [46], [65], [66], [67] consume more node power, energy, and have higher control message overhead, as shown in Figures 17, 18, and Table 8, respectively.

Basically, for comparison, we judiciously consider metrics that are maximum common with the state-of-the-art schemes. Further we consider those approaches that have maximum reported QoS metrics. We consider the derived parameters from the reported parameters, wherever required. Though DETONAR [52] achieves full accuracy in attack node identification, but achieves 80% in case of version attacks. Version attack detection accuracy using our proposed scheme fares better than DETONAR. Also, our approach is scalable while DETONAR is applicable to small networks only. The packet overhead (CONMO) in DETONAR is also significantly higher than our proposed scheme. InDReS [32] considers the QoS metrics but does not report the false positives, false negatives or accuracy of their algorithmic procedure. The results show that our proposed approach achieves comparatively better results overall with performance parameters, as shown in Figures 19, 20, and 21. The accuracy of our proposed approach is calculated based on TPR and TNR values shown in Figure 21, while a comparison of results is shown in Table 8.

E. DISCUSSION

In our scheme, attack is detected and the attacker, that launches the attack, is identified at the same time. Accurate identification of node implies that attack is also detected accurately. Conversely, attack is detected implies some node is identified as an attack node. A detection accuracy of 99.1% for our proposed solution, as shown in Table 8, means identification accuracy is also 99%. Our proposed design is inspired from intrusion detection using probing techniques that have been successfully applied to wired and wireless network security solutions [29], [70], [71]. The applicability

TABLE 8. Comparison of the proposed scheme with the closely related works.

References	Ⓢ/Ⓣ EN	EU (mJ)	POWCPN (mW)	THP	PDR (%)	CONMO (in Pkt.)	TPR (%)	TNR (%)	ACC (%) RAD	ACC (%) VNAD	RAI ACC (%)	SCAL
A. Le et al. (2011) [66]	Ⓢ	11479	1.36	N/A	N/A	N/A	93.50	94.41	94.32	N/A	94.32	✗
S. Usman et al. (2018) [46]	Ⓢ	15479	1.48	N/A	N/A	N/A	94.75	94.70	95.20	N/A	95.20	✗
M. Nikravan et al. (2018) [18]	N/A	13938	1.89	0.667	N/A	N/A	N/A	N/A	90.11	90.11	N/A	✓
D. Airehrour et al. (2019) [20]	Ⓣ	22580	1.52	0.738	93.97	5045	94.70	95.62	94.89	N/A	N/A	✗
ZA. Almusaylim et al. (2020) [68]	Ⓢ	18953	1.69	0.717	93.45	1095	94.46	95.12	94.82	98.30	94.82	✓
S. Sharma et al. (2020) [67]	Ⓢ	13890	1.57	0.694	94.13	1012	N/A	N/A	N/A	N/A	N/A	✓
R. Sahay et al. (2020) [69]	Ⓢ	17385	N/A	N/A	N/A	2068	93.3	94.12	94.50	N/A	94.50	✗
S. Nayak et al. (2021) [64]	Ⓢ, Ⓣ	N/A	N/A	N/A	N/A	N/A	93.45	93.60	93.58	70.4	N/A	✗
S. Ibrahim et al. (2022) [65]	Ⓢ	18839	1.62	0.718	97.98	950	N/A	N/A	99.01	99.00	N/A	✓
A. Mayzaud et al. (2017) [51]	Ⓢ	N/A	N/A	N/A	N/A	N/A	97.28	N/A	98.53	98.53	N/A	✓
A. Zeeshan et. al (2017) [50]	Ⓢ	N/A	N/A	N/A	N/A	N/A	95.00	89.00	N/A	92.00	N/A	✓
A. Andrea et. al (2021) [52]	Ⓣ	N/A	N/A	N/A	N/A	15430	N/A	N/A	100	80.00	100	✗
M. Surendar et. al (2016) [32]	Ⓢ	12492	N/A	0.949	95.41	750	N/A	N/A	N/A	N/A	N/A	✓
Proposed solution	Ⓢ, Ⓣ	14872	1.41	0.743	**99.34**	**680**	**98.43**	**99.73**	99.1	99.1	**99.1**	✓

Ⓢ: Simulation, Ⓣ: Testbed, POWCPN: POWer Consumption Per Node, THP: Throughput, PDR: Packet delivery ratio, CONMO: CONtrol Message Overhead
SCAL: Scalability, ACC: Accuracy, RAD: Rank Attack detection, VAD: Version Attack detection, RAI: Rank attack Identification, N/A: Not available

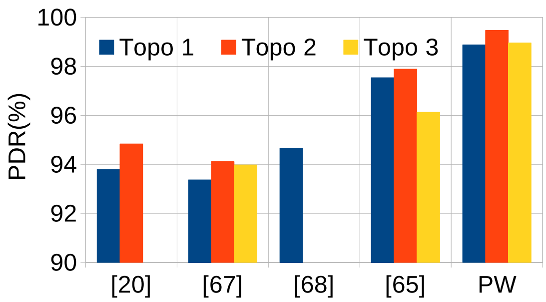


FIGURE 19. PDR comparison with related works.

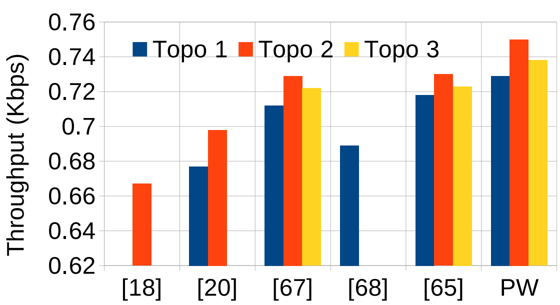


FIGURE 20. Throughput comparison with related works.

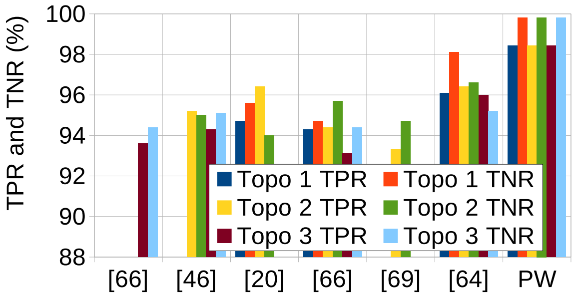


FIGURE 21. TPR and TNR comparison with related works.

of our approach in the IoT context has been shown through 6LoWPAN fragmentation [72] and CoAP request/response spoofing attack detection [53].

ICMPv6 probe request packets are sent with random payload. But, the receipt of an acknowledgement and the time of receipt of the acknowledgement only matter. Since the payload information is not of our interest, alteration of packets does not affect the detection procedure. A probe response transition is taken only if it is received from the same node to whom the probe was sent. Hence, spoofing will not help the attack motive. Probe packets may be communicated concurrently via different downward paths. To avoid self-identification, the attack node reports truly. Communication lags due to the underlying RPL-IoT network conditions will uniformly affect every node along a path in the DODAG. Response delay is an attack characteristic in our detection procedure. If a malicious node delays a packet, then it is identified more easily. If an attack node holds the packet for indefinitely long and does not forward it, then such a case is also an attack behavior. So delay or not responding does not deter the detection process. Furthermore, a DIO multicast simultaneously affects in route updation and inference of suspicious activity by multiple leaf agents. Hence, due to multiple leaf agents present, if any agent misses reporting, it does not hinder our identification mechanism. The case of malfunctioning leaf agents, if compromised, is not explicitly dealt with in this paper.

Studies in the literature have analyzed variants of rank attacks. In Le et al. [73], the authors propose few variants. Their impact on the DODAG topology from the perspective of end-to-end delay and packet delivery ratio is highlighted. Their work shows that there exists unique threats to RPL that evade regular detection techniques. This is so because such attacks do not consider changing the advertised rank value; rather, they create un-optimized paths, silently. These types of attacks pose a different nature to the traditional threats making it complex enough to be defended, for example, the blackhole attacks that add delay to transmissions. They have specifically considered four types of rank attack variations, namely, 1) Permanently and updates about the rank change to its neighbors, 2) Non-permanently (flipping between its choices between normal and abnormal) and updates about the rank change to its neighbors, 3) Permanently and does not update about the rank change to its neighbors and 4) Non-permanently and does not update about the rank change to its neighbors.

We show to detect version attacks apart from rank attack identification. As per knowledge, this is the first-of-a-kind attempt to mitigate RPL routing attacks using finite state automata based DES based IDS. State-of-the-art attacks that produce the same consequences as the increased rank attacks, can be also detected and malicious nodes can be uniquely located using our procedure, as it is. Attacks such as the worst parent attack, neighbour attack, etc., that result in similar consequences as covered in our rank attack procedure will also be detected in our scheme. Though we explicitly do not model these attacks, yet a class of worst parent attack with update, i.e., the worst parent choice is passed on to child nodes, falsely, is one of the attack cases that we consider. Hence, such an attack will be detected. Also, a class of neighbour attacks where the advertised parent node is out of range of the DIO recipient, and the attack results in a post-attack topology as dealt in our scheme will also be detected. Further, cross-layer attacks that use increased rank attacks are also detected using our scheme. RPL analysis on the packet exchange dynamics due to other attacks is thereby necessary. Decreased rank attacks, sinkhole attacks and blackhole attacks are also DIO specific attacks launched in a similar manner, i.e., a lower rank value is falsely advertised in DIO to attract nodes. The effects of these attacks are analogous to increased rank attack. DES based IDS can be extended to detect other attacks by adding relevant states and transitions for control or data packet communication behavior in the monitored RPL-IoT. As it is, decreased rank attack or sinkhole that are manifested towards increased rank attack can also be detected using our scheme with minimum customisation, even when combined with selective forwarding attack. This would require careful but minimum modifications to be made to our algorithm and extending our model for detection. Other forms of attacks, which directly map to an increased rank attack scenario, will also be detected using our scheme with minor changes. Moreover, one advantage of using DES based IDS is that false positives are minimal. A non-zero false positive in our experiments can be related to reasons such as, packet loss if considered as a missing acknowledgement response and network lags beyond the estimated values of Δ_{max} and Δ_a.The current solution can be further improved with generation of optimized sequences of probes for more early detection and also thereby reducing complexity. The placement of the agents can be improved such that the overhead is further reduced.

VI. CONCLUSION

A novel RPL rank attacker identification scheme that also detects version attack is presented. Our proposed scheme is centralized and uses an intelligent probing technique and DES based IDS. We augment traditional DES based IDS such that attacker type is also diagnosed. Using our scheme location of attack node is identified accurately. Active probe packets are used judiciously to capture a deviation of attack behavior from the normal behavior which is normally lacking. A DES diagnoser serves as our IDS engine that generates an alert when an attack node is identified. The correctness and completeness of our approach is also proved.

The performance analysis of our proposed scheme in simulation and real testbed considers both attack and non-attack behavior patterns, with a sufficiently large number of IoT devices. The average energy usage and accuracy of our proposed approach are $14872mJ$ and 99.1%, respectively. The observed results show our approach is energy-efficient with lowest packet overhead than existing works. It is scalable, achieves minimum false positives, and higher accuracy with lower detection time.

APPENDIX

A. BASICS OF DISCRETE EVENT SYSTEMS

This subsection presents the prerequisites of our proposed DES framework. Using the knowledge and demonstration of this section, we later show that the framework can be used to diagnose attacks in wireless sensor networks containing resource constrained nodes [53], [72].

1) DES MODEL

The DES model H is defined as a 6-tuple $H = \langle X, X_0, \Sigma, V, C, \Im \rangle$ [57], [59], [74], [75], [76]. Here, X is the set of states and is finite, $X_0 \subseteq X$ is the set of initial states, Σ is the finite set of events, V is the finite set of model variables, C is the finite set of clock variables and $\Im$ is the finite set of transitions. Elements of the set of model variables assume values from their respective domain sets. Suppose if $V = \{v_1, v_2, \ldots, v_n\}$ is the set of model variables (for some finite value of n) where each element v_i takes some values from its domain set Dom_i. The domain of each of the clock variables is the set of non-negative reals, R. A transition $\tau \in \Im$ is defined as a 7-tuple $\langle x, x^+, \sigma, \phi(V), \Phi(C), Reset(C), Assign(V) \rangle$, where x, x^+ are the source state and destination state of transition τ respectively. Due to the occurrence of the event $\sigma \in \Sigma$, the transition τ is enabled. $\phi(V)$ is defined as a boolean conjunction of equalities over some subset of the model variables, V, and which needs to hold true overall for a transition to be taken. $\Phi(C)$ is an invariant condition over some subset of the clock variables C. $Reset(C)$ is a subset of clock variables to be reset and $Assign(V)$ is a subset of model variables along with an assignment of values from their corresponding domains. Some of the fields in the tuple representing a transition maybe be denoted by "-". For example, if "-" is used for $\phi(V)$ or $Assign(V)$, then it would mean that no condition needs to be met (i.e., the condition is implicitly TRUE) or NO assignment is required respectively.

2) DEFINITIONS

Due to certain measurement limitations, some events cannot be measured. Such events are called unmeasurable events.

The event set can be expressed as a disjoint union of measurable and unmeasurable events. In notation, $\Sigma = \Sigma_m \cup \Sigma_{um}$.

Definition 1 (Measurable and Unmeasurable Transitions): A transition, τ, that is enabled under the influence of an event σ is said to be measurable if the corresponding event, σ, is measurable. Similarly, a transition associated with an unmeasurable event is said to be an unmeasurable transition. $\Im_m$ and $\Im_{um}$ denote the set of measurable and unmeasurable transitions.

Definition 2 (Measurement equivalent transitions (states)): A pair of transitions $\tau_1 = \langle x_1, x_1^+, \sigma_1, \phi_1(V), \Phi_1(C), Reset_1(C), Assign_1(V)\rangle$ and $\tau_2 = \langle x_2, x_2^+, \sigma_2, \phi_2(V), \Phi_2(C), Reset_2(C), Assign_2(V)\rangle$ are said to be measurement equivalent *iff* $\sigma_1 = \sigma_2$, $\phi_1(V) = \phi_2(V)$, $\Phi_1(C) = \Phi_2(C)$, $Reset_1(C) = Reset_2(C)$ and $Assign_1(V) = Assign_2(V)$. If a pair of transitions are equivalent, then their source states and destination states are equivalent states pair-wise. In simple terms, if the system current state is an initial state of a transition that has at least one more equivalent state, then the final states reached, from each of these states due to an equivalent transition, are also equivalent.

Definition 3 (Projection and Inverse Projection Operator): A projection operator $P : \Im^* \rightarrow \Im_m^*$ is defined as: $P(\epsilon) = \epsilon$ (null string); $P(\tau) = \tau$ if $\tau \in \Im_m$; $P(\tau) = \epsilon$ if $\tau \in \Im_{um}$; $P(s\tau) = P(s)P(\tau)$, where $s \in L_f(H)$, $\tau \in \Im$. The function P erases the unmeasurable transitions from the argument finite trace. $P(s)$ is termed as the measurable finite trace corresponding to the finite trace s.

Definition 4 (Normal H-state (H-transition) and Faulty H-state (H-transition)): States that are traversed by the system when operating without any fault are known as Normal H-states. X_N denotes the set of all normal states. A H-transition $\langle x, x^+\rangle$ is called a normal H-transition if $x, x^+ \in X_N$. States that are traversed by the system when operating under faulty circumstances are known as faulty H-states. X_{F_i} denotes the set of all faulty states. A H-transition $\langle x, x^+\rangle$ is called a faulty H-transition if $x, x^+ \in X_{F_i}$.

3) DIAGNOSABILITY

A key property relating to fault diagnosis in DES, diagnosability [58], [59], is discussed here. DES Diagnosability is a property related to event diagnosis where the earlier occurrence of certain events (faults) of interest are diagnosed. A diagnoser, constructed from DES models, tracks the system behavior and gives a decision on the diagnosis of monitored events. Now, a fault is diagnosable in finite time, if the diagnosability condition is met (F_i-Diagnosability property is satisfied). A lemma on the diagnosability property states that lack of fault indeterminate cycles guarantees diagnosability. It means that the diagnoser is able to give a decision in finite time on the occurrence of the event diagnosed, i.e, normal if the event has not occurred, and faulty if fault event has already occurred. Satisfaction of the diagnosability property, considering the limitations in measurement, ensures efficient fault detection as well as diagnosis of the fault type [77].

Definition 5 (F_i-Diagnosability): Let $\Psi(X_{F_i}) = \{s | s \in L_f(H)$ and $final(s) \in X_{F_i}$ and s ends in a measurable transition$\}$. A DES model H is said to be diagnosable for fault F_i *iff* the following holds:

$$(\exists n_j \in \mathcal{N})[\forall s \in \Psi(X_{F_i})](\forall t \in L_f(G)/s)[|t| \geq n_j) \Rightarrow D] \tag{4}$$

where, D is $\forall x \in \{P^{-1}[P(st)]\}, final(x) \in X_{F_i}$.

Construction of the diagnoser:

The diagnoser, O, is represented as a directed graph, $O = \langle Z, A\rangle$. Here, Z is the set consisting of the nodes of the diagnoser O, called O-nodes and A is the set consisting of the transitions (edges) of the diagnoser, called O-transitions, where $A \subseteq Z \times Z$. Each O-node z is an estimate of the actual system state and consists of one or more states of DES H, $z \in 2^X$, the power state of X, signifying membership uncertainty. On a similar note, each of the O-transition a consists of one or more measurement equivalent transition of DES H and represents an uncertainty in the actual measurable transition that takes place. They are of the form $\langle z_i, z_f\rangle$. We denote the unmeasurable successor set of a state set X as $\mathcal{U}(X)$ and is defined as $\mathcal{U}(X) = \bigcup_{x \in X}\{x^+ | \tau = \langle x, x^+\rangle \in \Im_u\}$. The unmeasurable reach of a state set X, $\mathcal{U}^*(X)$, is the reflexive-transitive closure of $\mathcal{U}(X)$. In Algorithm 3, the step-wise procedure for diagnoser construction is shown.

Algorithm 3 Diagnoser Construction O for DES Model H

Input: DES model H
Output: DES Diagnoser
/* PARTITION $X_0 \rightarrow$ Measurement equivalent classes, X_{01}, X_{02}, ..., X_{0m} */
for *all i*, $1 \leq i \leq m$ **do**
  $z_{0i} \leftarrow \mathcal{U}^*(X_{0i})$
end
$Z_0 \leftarrow z_{01} \cup \cdots \cup z_{0m}$
$Z \leftarrow Z_0$
$A \leftarrow \phi$
for *all* $z \in Z$ **do**
  /* Find the set of measurable H-transitions ($\Im_{mz}$) outgoing from z */
  $\Im_{mz} \leftarrow \{\tau | \tau \in \Im_m \wedge initial(\tau) \in z\}$
  /* Find the set of all measurement equivalent classes A_z, of $\Im_{mz}$ */
  for *all* $a \in A_z$ **do**
    $z_a^+ = \{final(\tau) | \tau \in a\}$
    $z^+ = \mathcal{U}^*(z_a^+)$
    $Z \leftarrow Z \cup \{z^+\}$
    $A = A \cup \{a\}$
  end
end

F_i-certain O-node and F_i-uncertain O-node are two types of diagnoser nodes that relate to occurrence of a fault type F_i. F_i-certain O-nodes consists purely of F_i-H-states while an F_i-uncertain O-node consists of states that may belong to F_i-H-states as well as states of DES H other than the fault type F_i.

REFERENCES

[1] L. Atzori, I. A. Iera, and M. Giacomo, "The Internet of Things: A survey," *Comput. Netw.*, vol. 54, pp. 2787–2805, May 2010.

[2] M. Humayun, N. Jhanjhi, and M. Z. Alamri, "Smart secure and energy efficient scheme for E-health applications using IoT: A review," *Int. J. Comput. Sci. Netw. Secur.*, vol. 20, no. 4, pp. 55–74, 2020.

[3] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on Internet of Things: Architecture, enabling technologies, security and privacy, and applications," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1125–1142, Oct. 2017.

[4] T. Winter, P. Thubert, A. Brandt, J. W. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J.-P. Vasseur, and R. K. Alexander, *RPL: IPv6 Routing Protocol for Low Power and Lossy Networks*, document RFC 6550, 2012, pp. 1–157.

[5] P. Pongle and G. Chavan, "A survey: Attacks on RPL and 6LoWPAN in IoT," in *Proc. Int. Conf. Pervasive Comput. (ICPC)*, Jan. 2015, pp. 1–6.

[6] L. Wallgren, S. Raza, and T. Voigt, "Routing attacks and countermeasures in the RPL-based Internet of Things," *Int. J. Distrib. Sensor Netw.*, vol. 9, no. 8, pp. 1–11, 2013.

[7] A. Mayzaud, R. Badonnel, I. Chrisment, and I. G. Est-Nancy, "A taxonomy of attacks in RPL-based Internet of Things," *Int. J. Netw. Secur.*, vol. 18, no. 3, pp. 459–473, 2016.

[8] A. Mayzaud, A. Sehgal, R. Badonnel, I. Chrisment, and J. Schönwälder, "A study of RPL DODAG version attacks," in *Proc. IFIP Int. Conf. Auto. Infrastructure, Manage. Secur.* Cham, Switzerland: Springer, 2014, pp. 92–104.

[9] C. Pu and L. Carpenter, "Digital signature based countermeasure against puppet attack in the Internet of Things," in *Proc. IEEE 18th Int. Symp. Netw. Comput. Appl. (NCA)*, Sep. 2019, pp. 1–4.

[10] C. Pu, J. Brown, and L. Carpenter, "A Theil index-based countermeasure against advanced vampire attack in Internet of Things," in *Proc. IEEE 21st Int. Conf. High Perform. Switching Routing (HPSR)*, May 2020, pp. 1–6.

[11] C. Pu and B. Groves, "Energy depletion attack in low power and lossy networks: Analysis and defenses," in *Proc. 2nd Int. Conf. Data Intell. Secur. (ICDIS)*, Jun. 2019, pp. 14–21.

[12] E. Y. Vasserman and N. Hopper, "Vampire attacks: Draining life from wireless ad hoc sensor networks," *IEEE Trans. Mobile Comput.*, vol. 12, no. 2, pp. 318–332, Feb. 2013.

[13] C. Pu and K.-K.-R. Choo, "Lightweight Sybil attack detection in IoT based on Bloom filter and physical unclonable function," *Comput. Secur.*, vol. 113, Feb. 2022, Art. no. 102541.

[14] C. Pu, "Spam DIS attack against routing protocol in the Internet of Things," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Feb. 2019, pp. 73–77.

[15] C. Pu and X. Zhou, "Suppression attack against multicast protocol in low power and lossy networks: Analysis and defenses," *Sensors*, vol. 18, no. 10, p. 3236, Sep. 2018.

[16] A. Raoof, A. Matrawy, and C.-H. Lung, "Routing attacks and mitigation methods for RPL-based Internet of Things," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1582–1606, 2nd Quart., 2019.

[17] G. Glissa, A. Rachedi, and A. Meddeb, "A secure routing protocol based on RPL for Internet of Things," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–7.

[18] M. Nikravan, A. Movaghar, and M. Hosseinzadeh, "A lightweight defense approach to mitigate version number and rank attacks in low-power and lossy networks," *Wireless Pers. Commun.*, vol. 99, no. 2, pp. 1035–1059, Mar. 2018.

[19] M. Zaminkar, F. Sarkohaki, and R. Fotohi, "A method based on encryption and node rating for securing the RPL protocol communications in the IoT ecosystem," *Int. J. Commun. Syst.*, vol. 34, no. 3, pp. 1–24, Feb. 2021.

[20] D. Airehrour, J. A. Gutierrez, and S. K. Ray, "SecTrust-RPL: A secure trust-aware RPL routing protocol for Internet of Things," *Future Gener. Comput. Syst.*, vol. 93, pp. 860–876, Apr. 2019.

[21] K. Iuchi, T. Matsunaga, K. Toyoda, and I. Sasase, "Secure parent node selection scheme in route construction to exclude attacking nodes from RPL network," in *Proc. 21st Asia–Pacific Conf. Commun. (APCC)*, Oct. 2015, pp. 299–303.

[22] N. Djedjig, D. Tandjaoui, F. Medjek, and I. Romdhani, "Trust-aware and cooperative routing protocol for IoT security," *J. Inf. Secur. Appl.*, vol. 52, pp. 1–25, Jun. 2020.

[23] M. Osman, J. He, F. M. M. Mokbal, N. Zhu, and S. Qureshi, "ML-LGBM: A machine learning model based on light gradient boosting machine for the detection of version number attacks in RPL-based networks," *IEEE Access*, vol. 9, pp. 83654–83665, 2021.

[24] S. Cakir, S. Toklu, and N. Yalcin, "RPL attack detection and prevention in the Internet of Things networks using a GRU based deep learning," *IEEE Access*, vol. 8, pp. 183678–183689, 2020.

[25] F. Y. Yavuz, D. Ünal, and E. Gül, "Deep learning for detection of routing attacks in the Internet of Things," *Int. J. Comput. Intell. Syst.*, vol. 12, no. 1, pp. 39–58, Nov. 2018.

[26] M. Osman, J. He, F. M. M. Mokbal, and N. Zhu, "Artificial neural network model for decreased rank attack detection in RPL based on IoT networks," *Int. J. Netw. Secur.*, vol. 23, no. 3, pp. 496–503, 2021.

[27] N. Hubballi, S. Biswas, S. Roopa, R. Ratti, and S. Nandi, "LAN attack detection using discrete event systems," *ISA Trans.*, vol. 50, no. 1, pp. 119–130, Jan. 2011.

[28] F. A. Barbhuiya, M. Agarwal, S. Purwar, S. Biswas, and S. Nandi, "Application of stochastic discrete event system framework for detection of induced low rate TCP attack," *ISA Trans.*, vol. 58, pp. 474–492, Sep. 2015.

[29] M. Agarwal, S. Biswas, and S. Nandi, "Discrete event system framework for fault diagnosis with measurement inconsistency: Case study of rogue DHCP attack," *IEEE/CAA J. Autom. Sinica*, vol. 6, no. 3, pp. 789–806, May 2019.

[30] V. Ramachandran and S. Nandi, "Detecting ARP spoofing: An active technique," in *Proc. Int. Conf. Inf. Syst. Secur.* Kolkata, India: Springer, Dec. 2005, pp. 239–250.

[31] F. A Barbhuiya, S. Biswas, and S. Nandi, "An active host-based intrusion detection system for ARP-related attacks and its verification," 2013, *arXiv:1306.1332*.

[32] M. Surendar and A. Umamakeswari, "InDReS: An intrusion detection and response system for Internet of Things with 6LoWPAN," in *Proc. Int. Conf. Wireless Commun., Signal Process. Netw. (WiSPNET)*, Mar. 2016, pp. 1903–1908.

[33] A. Dvir, T. Holczer, and L. Buttyan, "VeRA–version number and rank authentication in RPL," in *Proc. IEEE 8th Int. Conf. Mobile Ad-Hoc Sensor Syst.*, Oct. 2011, pp. 709–714.

[34] H. Perrey, M. Landsmann, O. Ugus, M. Wählisch, and T. C. Schmidt, "TRAIL: Topology authentication in RPL," in *Proc. Int. Conf. Embedded Wireless Syst. Netw.*, 2016, pp. 59–64.

[35] A. 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. 1–19, 2016.

[36] S. Raza, L. Wallgren, and T. Voigt, "SVELTE: Real-time intrusion detection in the Internet of Things," *Ad Hoc Netw.*, vol. 11, no. 8, pp. 2661–2674, Nov. 2013.

[37] T. ul Hassan, M. Asim, T. Baker, J. Hassan, and N. Tariq, "CTrust-RPL : A control layer-based trust mechanism for supporting secure routing in routing protocol for low power and lossy networks-based Internet of Things applications," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 3, Mar. 2021, Art. no. e4224.

[38] S. Y. Hashemi and F. S. Aliee, "Dynamic and comprehensive trust model for IoT and its integration into RPL," *J. Supercomput.*, vol. 75, no. 7, pp. 3555–3584, Jul. 2019.

[39] A. Seyfollahi, M. Moodi, and A. Ghaffari, "MFO-RPL: A secure RPL-based routing protocol utilizing moth-flame optimizer for the IoT applications," *Comput. Standards Interface*, vol. 82, pp. 1–19, Aug. 2022.

[40] Z. A. Almusaylim, A. Alhumam, and N. Z. Jhanjhi, "Proposing a secure RPL based Internet of Things routing protocol: A review," *Ad Hoc Netw.*, vol. 101, Apr. 2020, Art. no. 102096.

[41] A. A. Diro and N. Chilamkurti, "Distributed attack detection scheme using deep learning approach for Internet of Things," *Future Gener. Comput. Syst.*, vol. 82, pp. 761–768, May 2018.

[42] B. B. Zarpelão, R. S Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in Internet of Things," *J. Netw. Comput. Appl.*, vol. 84, pp. 25–37, Apr. 2017.
[43] N. Mishra and S. Pandya, "Internet of Things applications, security challenges, attacks, intrusion detection, and future visions: A systematic review," *IEEE Access*, vol. 9, pp. 59353–59377, 2021.
[44] S. U. Jan, S. Ahmed, V. Shakhov, and I. Koo, "Toward a lightweight intrusion detection system for the Internet of Things," *IEEE Access*, vol. 7, pp. 42450–42471, 2019.
[45] A. Althubaity, T. Gong, K.-K. Raymond, M. Nixon, R. Ammar, and S. Han, "Specification-based distributed detection of rank-related attacks in RPL-based resource-constrained real-time wireless networks," in *Proc. IEEE Conf. Ind. Cyberphys. Syst. (ICPS)*, vol. 1, Jun. 2020, pp. 168–175.
[46] U. Shafique, A. Khan, A. Rehman, F. Bashir, and M. Alam, "Detection of rank attack in routing protocol for low power and lossy networks," *Ann. Telecommun.*, vol. 73, nos. 7–8, pp. 429–438, Aug. 2018.
[47] A. D. Seth, S. Biswas, and A. K. Dhar, "LDES: Detector design for version number attack detection using linear temporal logic based on discrete event system," *Int. J. Inf. Secur.*, early access, pp. 1–25, Mar. 2023, doi: 10.1007/s10207-023-00665-3.
[48] H. Sedjelmaci, S. M. Senouci, and M. Al-Bahri, "A lightweight anomaly detection technique for low-resource IoT devices: A game-theoretic methodology," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–6.
[49] C. Cervantes, D. Poplade, M. Nogueira, and A. Santos, "Detection of sinkhole attacks for supporting secure routing on 6LoWPAN for Internet of Things," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM)*, May 2015, pp. 606–611.
[50] Z. A. Khan and P. Herrmann, "A trust based distributed intrusion detection mechanism for Internet of Things," in *Proc. IEEE 31st Int. Conf. Adv. Inf. Netw. Appl. (AINA)*, Mar. 2017, pp. 1169–1176.
[51] A. Mayzaud, R. Badonnel, and I. Chrisment, "A distributed monitoring strategy for detecting version number attacks in RPL-based networks," *IEEE Trans. Netw. Service Manage.*, vol. 14, no. 2, pp. 472–486, Jun. 2017.
[52] A. Agiollo, M. Conti, P. Kaliyar, T.-N. Lin, and L. Pajola, "DETONAR: Detection of routing attacks in RPL-based IoT," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 2, pp. 1178–1190, Jun. 2021.
[53] D. Ray, P. Bhale, S. Biswas, S. Nandi, and P. Mitra, "DAISS: Design of an attacker identification scheme in CoAP request/response spoofing," in *Proc. TENCON IEEE Region Conf. (TENCON)*, Dec. 2021, pp. 941–946.
[54] J. Yi, T. Clausen, and Y. Igarashi, "Evaluation of routing protocol for low power and lossy networks: LOADng and RPL," in *Proc. IEEE Conf. Wireless Sensor (ICWISE)*, Dec. 2013, pp. 19–24.
[55] J. Vasseur, M. Kim, K. Pister, N. Dejean, and D. Barthel, *Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks*, document RFC 6551, IETF, 2012, pp. 1–30.
[56] P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko, *The Trickle Algorithm*, document RFC 6206, Internet Eng. Task Force, 2011.
[57] C. G. Cassandras, *Discrete Event Systems: Modeling and Performance Analysis*. Boca Raton, FL, USA: CRC Press, 1993.
[58] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Diagnosability of discrete-event systems," *IEEE Trans. Autom. Control*, vol. 40, no. 9, pp. 1555–1575, Sep. 1995.
[59] S. H. Zad, R. H. Kwong, and W. M. Wonham, "Fault diagnosis in discrete-event systems: Framework and model reduction," *IEEE Trans. Autom. Control*, vol. 48, no. 7, pp. 1199–1212, Jul. 2003.
[60] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki—A lightweight and flexible operating system for tiny networked sensors," in *Proc. 29th Annu. IEEE Int. Conf. Local Comput. Netw.*, Nov. 2004, pp. 455–462.
[61] C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele, and T. Watteyne, "FIT IoT-LAB: A large scale open experimental IoT testbed," in *Proc. IEEE 2nd World Forum Internet Things (WF-IoT)*, Dec. 2015, pp. 459–464.
[62] S. Godard. *SAR—Collect, Report, or Save System Activity Information*. Accessed: Mar. 10, 2023. [Online]. Available: https://manpages.ubuntu.com/manpages/lunar/en/man1/sar.sysstat.1.html
[63] A. Tirumala. (1999). *IPERF: The TCP/UDP Bandwidth Measurement Tool*. [Online]. Available: http://dast.nlanr.net/Projects/Iperf/
[64] S Nayak, N. Ahmed, and S. Misra, "Deep learning-based reliable routing attack detection mechanism for industrial Internet of Things," *Ad Hoc Netw.*, vol. 123, pp. 1–11, Dec. 2021.
[65] I. S. Alsukayti and A. Singh, "A lightweight scheme for mitigating RPL version number attacks in IoT networks," *IEEE Access*, vol. 10, pp. 111115–111133, 2022.
[66] A. Le, J. Loo, Y. Luo, and A. Lasebae, "Specification-based IDS for securing RPL from topology attacks," in *Proc. IFIP Wireless Days (WD)*, Oct. 2011, pp. 1–3.
[67] S. Sharma and V. K. Verma, "Security explorations for routing attacks in low power networks on Internet of Things," *J. Supercomput.*, vol. 77, pp. 4778–4812, Oct. 2020.
[68] Z. A. Almusaylim, N. Z. Jhanjhi, and A. Alhumam, "Detection and mitigation of RPL rank and version number attacks in the Internet of Things: SRPL-RP," *Sensors*, vol. 20, no. 21, pp. 1–25, 2020.
[69] R. Sahay, G. Geethakumari, and B. Mitra, "A novel blockchain based framework to secure IoT-LLNs against routing attacks," *Computing*, vol. 102, no. 11, pp. 2445–2470, Nov. 2020.
[70] H. Neminath, S. Biswas, S. Roopa, R. Ratti, S. Nandi, F. A. Barbhuiya, A. Sur, and V. Ramachandran, "A DES approach to intrusion detection system for ARP spoofing attacks," in *Proc. 18th Medit. Conf. Control Autom. (MED)*, Jun. 2010, pp. 695–700.
[71] F. A. Barbhuiya, S. Biswas, N. Hubballi, and S. Nandi, "A host based Des.approach for detecting ARP spoofing," in *Proc. IEEE Symp. Comput. Intell. Cyber Secur. (CICS)*, Apr. 2011, pp. 114–121.
[72] D. Ray, P. Bhale, S. Biswas, S. Nandi, and P. Mitra, "ArsPAN: Attacker revelation scheme using discrete event system in 6LoWPAN based buffer reservation attack," in *Proc. IEEE Int. Conf. Adv. Netw. Telecommun. Syst. (ANTS)*, Dec. 2020, pp. 1–6.
[73] A. Le, J. Loo, A. Lasebae, A. Vinel, Y. Chen, and M. Chai, "The impact of rank attack on network topology of routing protocol for low-power and lossy networks," *IEEE Sensors J.*, vol. 13, no. 10, pp. 3685–3692, Oct. 2013.
[74] K. T. Cheng and A. S. Krishnakumar, "Automatic functional test generation using the extended finite state machine model," in *Proc. 30th Int. Design Autom. Conf. (DAC)*, 1993, pp. 86–91.
[75] R. Sekar, M. Bendre, D. Dhurjati, and P. Bollineni, "A fast automaton-based method for detecting anomalous program behaviors," in *Proc. IEEE Symp. Secur. Privacy (S&P)*, May 2000, pp. 144–155.
[76] R. Sekar, A. Gupta, J. Frullo, T. Shanbhag, A. Tiwari, H. Yang, and S. Zhou, "Specification-based anomaly detection: A new approach for detecting network intrusions," in *Proc. 9th ACM Conf. Comput. Commun. Secur.*, Nov. 2002, pp. 265–274.
[77] P. K. Biswal, H. P. Sambho, and S. Biswas, "A discrete event system approach to on-line testing of digital circuits with measurement limitation," *Eng. Sci. Technol., Int. J.*, vol. 19, no. 3, pp. 1473–1487, Sep. 2016.

DIPOJJWAL RAY received the B.Tech. degree in electronics and communication engineering from the Institute of Engineering and Management (IEM), Kolkata, West Bengal, India, in 2010. He is currently pursuing the dual master's and Ph.D. degree in computer science with the Department of Computer Science and Engineering, Indian Institute of Technology (IIT) Guwahati, Guwahati, Assam, India. Prior to joining IIT Guwahati, he was with Tata Consultancy Services as a Windows Phone Application Developer and later as a Research Fellow of big data with the Department of Computer Science and Engineering, Indian Institute of Technology (IIT) Bombay, Mumbai, Maharashtra, India. His current research interests include formal methods and verification, discrete-event systems, computer network security, and hardware security.

PRADEEPKUMAR BHALE (Student Member, IEEE) received the B.Eng. degree in computer science and engineering from the Government College of Engineering, Aurangabad, Maharashtra, India, in 2010, and the M.Tech. degree in information security from the ABV-Indian Institute of Information Technology Gwalior, Madhya Pradesh, India, in 2014. He is currently pursuing the D.Phil. degree with the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, Assam, India. He was with Tech Mahindra Pvt. Ltd., as a Technical Associate, for two years. He joined as an Assistant Professor with the Dr. B. R. Ambedkar National Institute of Technology, Jalandhar, Punjab, India, for two years. His current research interests include the IoT security, wireless networks, cloud security, blockchain, and SDN security. He was awarded the State of Maharashtra Post Graduate Fellowship for the master's degree.

SANTOSH BISWAS (Senior Member, IEEE) received the B.E. degree from the National Institute of Technology Durgapur, in 2001, the M.S. degree (Hons.) from the Department of Electrical Engineering, Indian Institute of Technology Kharagpur, in 2004, and the Ph.D. degree from the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, in 2008. He joined the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, in 2009. Currently, he is the HOD of the EECS Department, IIT Bhilai. He has been involved in several research projects sponsored by industry and government agencies. He is engaged with academic and industry-sponsored research related to VLSI testing and design for testability. He has published about 150 research articles. His current research interests include VLSI testing and design for testability, fault tolerance, network security, discrete event systems, and embedded systems.

PINAKI MITRA (Member, IEEE) received the B.Tech. degree in computer science and engineering from Jadavpur University, Kolkata, in 1987, India, the M.Tech. degree in computer science and engineering from the Indian Institute of Science, Bengaluru, India, in 1989, and the Ph.D. degree from Simon Fraser University, Canada, in 1994. He was with the Department of Computer Science and Engineering, Jadavpur University. He joined the National Institute of Management, Kolkata, and served as an Assistant Professor. He joined IIT Guwahati, in December 2004, where he is currently an Associate Professor with the Department of Computer Science and Engineering. His current research interests include cryptography, network security, computer graphics, and multimedia.

SUKUMAR NANDI (Senior Member, IEEE) received the Ph.D. degree in computer science and engineering from the Indian Institute of Technology Kharagpur, India, in 1995. He was a Visiting Senior Fellow with NTU, Singapore, from 2002 to 2003. He is currently a Senior Professor with the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, India. He has published more than 300 papers in reputed journals and conferences. His current research interests include traffic engineering, wireless networks, information security, computer architecture and algorithms, and VLSI design and testing. He is a Senior Member of ACM. He is a fellow of the Indian National Academy of Engineering (INAE), the Institution of Engineers, India, and the Institution of Electronics and Telecommunication Engineers, India.