ARTICLE IN PRESS

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Highlights

A Softwarized Intrusion Detection System for the RPL-based Internet of Things networks

Future Generation Computer Systems xxx (xxxx) xxx

George Violettas*, George Simoglou, Sophia Petridou, Lefteris Mamatas

- *ASSET* is a novel Intrusion Detection System (IDS) for RPL that mitigates at least 13 attacks with diverse profiles.
- *ASSET* inspired by the network softwarization paradigm, supports a novel, extendable workflow bringing together three anomaly-detection and four RPL specification-based mechanisms, a novel attacker identification process, as well as multiple attack mitigation strategies.
- Our IDS also supports an adaptable control & monitoring protocol, trading overhead for accuracy, depending on the network conditions.
- We provide evaluation results validating all *ASSET* features, its reduced control overhead and robustness, from both quantity and quality view-points.

Graphical abstract and Research highlights will be displayed in online search result lists, the online contents list and the online article, but will not appear in the article PDF file or print unless it is mentioned in the journal specific style requirement. They are displayed in the proof pdf for review purpose only.

Please cite this article as: G. Violettas, G. Simoglou, S. Petridou et al., A Softwarized Intrusion Detection System for the RPL-based Internet of Things networks, Future Generation Computer Systems (2021), https://doi.org/10.1016/j.future.2021.07.013.

FUTURE: 6180

ARTICLE IN PRESS

Future Generation Computer Systems xxx (xxxx) xxx

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

A Softwarized Intrusion Detection System for the RPL-based Internet of Things networks

George Violettas *, George Simoglou, Sophia Petridou, Lefteris Mamatas

Department of Applied Informatics, University of Macedonia, Egnatia 156, Thessaloniki, Greece

A R T I C L E I N F O

Article history:
Received 11 February 2021
Received in revised form 3 June 2021
Accepted 10 July 2021
Available online xxxx

Keywords:
Internet of Things
RPL protocol
RPL attacks
IoT security
Intrusion Detection System

A B S T R A C T

Internet of Things (IoT) constitutes a pivotal contributor to the Industry 4.0 (I 4.0) vision, technologically transforming production and societies. It enables novel services through the seamless integration of devices, such as motes carrying sensors, with the Internet. However, the broad adoption of IoT technologies is facing security issues due to the direct access to the devices from the Internet, the broadcasting nature of the wireless media, and the potential unattended operation of relevant deployments. In particular, the Routing over Low Power and Lossy Networks (RPL) protocol, a prominent IoT solution, is vulnerable to a large number of attacks, both of general-purpose and RPL-specific nature, while the resource-constraints of the corresponding devices are making attack mitigation even more challenging, e.g., in terms of involved control overhead and detection accuracy.

In this paper, we introduce *ASSET*, a novel Intrusion Detection System (IDS) for RPL with diverse profiles to tackle the above issues that mitigate at least 13 attacks. At the same time, other solutions go up to eight. *ASSET*, inspired by the network softwarization paradigm, supports a novel, extendable workflow, bringing together three anomaly-detection and four RPL specification-based mechanisms, a novel attacker identification process, as well as multiple attack mitigation strategies. Our IDS also supports an adaptable control & monitoring protocol, trading overhead for accuracy, depending on the network conditions. The proof-of-concept experiments show that *ASSET* entails a low overhead for the different modes of operation it supports (i.e., 6.28 percent on average) compared to other solutions reaching up to 30 percent. At the same time, it also keeps the power consumption at acceptable levels (from 0.18 up to 1.54 percent more). Moreover, it provides 100 percent accuracy for specific attacks and can identify the attacker in far more attacks than any other similar solution.

1. Introduction

Internet of Things (IoT) does rapidly develop and, among others, is the technological enabler for smart-x ecosystems and the next-generation advanced manufacturing, referred to as I 4.0 (Industry 4.0), that includes smart products, smart production, and smart services. Indeed, recent advances in communication technology, e.g., 5G Networks, along with the Industrial IoT (IIoT), evolve the request for mass production and automation from the principle idea to connect everything in the production chain to the more sophisticated context of broader and more fine-grained interconnections [1]. For example, a network of geographically distributed factory branches requires sharing resources and assets to improve orders' fulfillment. Data transfer among different entities is an essential but also a critical issue in such an automation ecosystem. The facility of exploiting everyday Internet-enabled devices as endpoints of accessing resources is an asset. Still, it entails hundreds of smart devices, sensors, and actuators communicating throughout large-scale IoT deployments, where, among others, security is an essential requirement.

1.1. Motivation

A prominent, standardized routing solution for IoT is the Routing for Low Power and Lossy Networks (RPL) [2,3], characterized by significant benefits. These include IPv6 support, moderate control overhead, and efficient low-power operation under challenging conditions, e.g., lossy links, heterogeneous and constraint devices with respect to their power, storage, memory and processing capabilities [4,5]. Despite its advantages, RPL still has open issues, the most important of which are related to attacks since it is based on the IP(v6) open stack and primarily uses wireless media for the nodes' communication.

According to the literature [6], RPL-related attacks include malicious actions aiming at: (i) exhausting nodes' resources as a means of significantly reducing the network's lifespan and availability, (ii) disrupting the structure of the Destination-Oriented

* Corresponding author.
E-mail address: georgevio@uom.edu.gr (G. Violettas).

https://doi.org/10.1016/j.future.2021.07.013
0167-739X/© 2021 Published by Elsevier B.V.

Directed Acyclic Graph (DODAG), upon which nodes' communication is based, affecting network's performance in respect to packet losses and end-to-end (E2E) delays. Passive attacks that monitor and intercept network traffic, e.g., sniffing, traffic analysis, are not part of the paper's scope since they do not exclusively concern RPL.

In fact, some attacks have no significant impact as standalone events, but they can be critically detrimental to the network in conjunction with others. Indicatively, impersonation attacks leave space for malicious activities to originate inside the network, against which encryption is not a suitable solution [7] because, for example, an insider attacker getting access to symmetric keys bypasses the applied RPL security mechanisms. Authenticated security could be a solution, but RPL RFC [2] does not specify any mechanisms for public key cryptography [8], which possibly cannot be supported by constrained nodes [9]. Hash schemes have been used for topology authentication without being able to mitigate rank-replay attacks [10].

On the protocol bulletproofing front, the RPL standard [2] specifies three modes of operation, i.e., unsecured mode, preinstalled mode, and authenticated mode. At the same time, it also defines mechanisms for data confidentially and authenticity, and replay protection [11,12]. Nevertheless, up to this time, RPL implementations on the most commonly used operating systems (e.g., Contiki OS and TinyOS) assume the unsecured mode of operation, putting aside RPL's security features, which are essentially characterized as optional. Authors in [11,13] elaborate on a partial implementation of such features, while according to [8], future versions of RPL will address such issues as authenticated security.

Until then, a suitable approach to encounter malicious activities is the Intrusion Detection Systems (IDSs) [6,7,12]. IDSs refer to a set of methods designed toward: (i) *detecting an attack*, (ii) *identifying the attacker*, and (iii) *mitigating* the event. They aim to detect several attacks concurrently, and ideally, they can be extended to deal with attacks that are not originally included in their design goals. Compared to the standalone mechanisms, they require some degree of collaboration among the network's nodes [12].

Regarding the RPL security, the design, development, and evaluation of an IDS should satisfy a set of requirements that reflect the solution's *width* and *depth*. We define the metrics of *robustness* and *extendability* for quantitative evaluation (*width*), referring to the range over which the impact of an IDS can be spread with respect to the number of attacks detected. Furthermore, given that new attacks and security issues emerge following the IoT research's progress, IDSs should be developed as a set of software components (mechanisms) to be quickly and on-the-fly modifiable to encounter attacks beyond their initial scope.

Moreover, we define the metrics of *accuracy* and *mitigation time* for qualitative evaluation (*depth*). In fact, an IDS should exhibit a high accuracy rate regarding both the event and the adversary; this means that the system does not misinterpret normal events or nodes' behavior as attacks or attackers, respectively, while minimizing the cases that attacks or intruders are overtaken. Once an attack/attacker has been detected, a mitigation strategy should be employed to rapidly handle the malicious nodes and restore the network's operation.

The research field of IDSs in the IoT domain is generally vast. Still, only a restricted subset of them is appropriate for Low-power and Lossy Networks (LLNs) [14,15], i.e., they take into consideration limitations regarding their lossy links, heterogeneous and resource-constrained devices. In fact, most of them have been proposed in the recent bibliography, i.e., from 2013 to 2020 [6,12,14]. An overview of these works makes clear that there is no one-for-all solution that succeeds in all three axes, i.e., to *detect* several attacks at once, to *identify* the intruder, and to *mitigate* the event, and at the same time, meet the aforementioned requirements of *robustness*, *extendability*, *high accuracy* and *rapid mitigation*.

1.2. Contribution

Along these lines, we introduce *ASSET*, a softwarized Intrusion Detection System that offers a holistic approach to shield an RPL-based IoT network against different types of attacks. Our system is inspired by the Software-Defined Networking (SDN) paradigm, i.e., it transfers functionality from the constraint end-nodes to central premises, i.e., the *Controller*, offloading both computational and communication overhead. At the same time, it follows a modular architecture that allows adaptations.

In particular, *ASSET* offers *a novel workflow* hosting well-known mechanisms for data analysis, e.g., the K-Means algorithm, that can efficiently collaborate in data exchange toward detecting several attacks and multiple intruders in the network. The challenging point is that we managed to appropriately synthesize a framework of independent components that are not merely put one next to the other, but they work as an integrated whole. Moreover, *ASSET's* workflow provides the background for further enhancements and extensions regarding detection or mitigation of attacks.

Next, we experiment with a minimum *set of mechanisms* for anomaly and RPL specification-based detection, able to address as many as 13 different types of RPL-related attacks with high accuracy and moderated cost. We exploit our literature review findings showing that combining detection methods as well as placement strategies brings advantages to the system [14]. In particular, *ASSET* hosts three anomaly detection methods on the node and/or on the *Controller*-level to provide the alternatives of a lightweight and a computationally-intensive solution, and four specification-based ones.

Most importantly, we develop *an adaptable control & monitoring protocol* enabling centralized network supervision. In practice, the protocol offers: (i) monitoring of RPL-related data, like UDP packets or ICMP statistics in an adaptable fashion, i.e., trading the amount of communicating information for control overhead in respect to the network's conditions; (ii) configuring RPL parameters on-the-fly as a means of enforcing centralized decisions to the network nodes once a mitigation action should be taken; and (iii) communicating node-level anomaly detection events that should trigger further investigation centrally, e.g., detailed monitoring by the *Controller*. To achieve adaptability, we define three modes of the protocol's operation, i.e., *slim-mode* that offers "baseline" monitoring at regular periods, *essential-mode* that indicates the first level of surveillance due to detected anomalies in more than three nodes, and *full-function-mode* that denotes the need of intensive surveillance due to detected anomalies that require detailed data from IoT nodes.

Novelties of *ASSET* could be summarized as follows: (i) detection and mitigation have been automated since all the mechanisms are incorporated under the umbrella of one workflow, orchestrated by the central controller; (ii) existing node-level features became centralized to offer a better balance and response capabilities; (iii) node-level features are programmable, with some addressing several attacks, providing a holistic view; (iv) the modular architecture makes it easy to add new features or alter existing ones; (v) it can be easily deployed over any kind of RPL network, anywhere in the central infrastructure, by only materializing the connection with the sink node; (vi) the bespoke fully parameterizable GUI provided, makes it a powerful tool in the hands of network administrators.

The rest of the paper is outlined as follows. We briefly present the RPL protocol and the attacks associated with it in Section 2. In Section 3 we elaborate on the proposed system, including details of its architecture, interfaces, and mechanisms. Our evaluation results are illustrated and discussed in Section 4. Related IDSs along with a comparative overview are presented in Section 5, while conclusions along with further-step ideas are summarized in the final section.

2. Background

2.1. RPL protocol

Our work elaborates on the RPL protocol [2] since it is the state-of-the-art routing protocol for LLNs. RPL is a distance-vector IPv6 protocol operating over the 6LoWPAN (IPv6 over Low-Power Wireless Personal Area Networks) protocol stack where each node builds the so-called DODAG to maintain an updated network topology [4,5]. RPL primarily supports multipoint-to-point communications, i.e., from the leaf-nodes upwards to the sink-node(s), which operates as a border router connecting the LLN with fixed infrastructure, e.g., via a serial connection.

RPL constructs the DODAG by utilizing an Objective Function (OF), which evaluates the different possible pathways from every node to the sink by solving a multi-variable, multi-objective optimization problem for routes' discovery. The default Minimum Rank with Hysteresis Objective Function (MRHOF) [16] considers the number of hops to the sink-node and/or the quality of each link between participating nodes into the above pathway(s) by utilizing the Expected Transition Count (ETX) metric. Other more sophisticated OFs are also described in the bibliography [17].

To avoid DODAG loops, RPL assigns each node a rank value related to the rank of the attached parent-node and the distance from the sink. A node can be (re-)attached to the graph with a lower rank than its current one upon discovering a new preferred parent. The opposite case (an updated greater rank) triggers a *Global Repair* self-healing mechanism, i.e., recalculating ranks for all network nodes [18], to avoid count-to-infinity problems. Moreover, a node resets its rank and re-solicit neighbors (i.e., *Local Repair*) once it loses its parent, i.e., without waiting for the whole network to reset [19]. To avoid exploitation of the above mechanisms that cause overhead and delays, RPL RFC [2] suggests a maximum threshold per hour for the repairs.

RPL's RFC [2] also defines four ICMPv6 (Internet Control Message Protocol) messages for information exchange and facilitating the DODAG construction. The DIO (DODAG Information Object) message is first fired by the sink, multicasted and populated downwards until all reachable nodes receive it. Among others, it includes timer settings, DODAG version, and mode of operation (storing/non-storing). DAO (Destination Advertisement Object) messages travel upward, advertising each node's ancestor until reaching the sink. The same information (node–ancestor pair) is also stored by each node the DAO went through. This way, each node maintains a version of the DODAG. DIS (DODAG Information Solicitation) is a unicast message beaconed periodically by a parentless node to solicit potential parents in its radio-coverage vicinity. DAO-ACK is an optional message for DAO acknowledgment that is usually omitted since it causes heavy overhead.

As the fundamental pillar of RRL, the DODAG needs to be updated and maintained frequently. A dedicated algorithm—the *Trickle Timer*—handles the frequency of DIO messages, upon which the graph's convergence time is based. The algorithm balances preserving the node's power consumption and keeping the network information up-to-date and trustworthy. To achieve this trade-off, DIO messages dispatching frequency varies from a few seconds, up to 17.5 min, since the *Trickle Timer's* duration is doubled each time it fires [5]. Any change in the DODAG, e.g., unreachable parent, DIO or DAO mismatch, or new parent selection, causes a *Trickle Timer Reset* for the particular node. As a result, DIO messages are dispatched at a higher rate when the network is unstable and at a slower rate otherwise, preserving energy and reducing network traffic.

DODAG, as well as the RPL messages and mechanisms, are the origin of the so-called RPL-related attacks described in the next section.

2.2. RPL-related attacks

Routing in the RPL networks is challenging due to the resource constraints of the connected devices. Moreover, such networks support dynamic topologies and are based on the wireless medium's passive nature. Consequently, they attract malicious actions, including but not limited to denial of service attacks (DoS), physical damages, and/or extraction of sensitive information, e.g., DODAG version, nodes' rank values, and nodes' IDs. In fact, legitimate nodes can be compromised by exploiting the RPL mechanisms themselves. Suppose a compromised node is located near the sink. In that case, a combination of attacks can be launched with severe effects, spanning from resource-depletion of nodes, due to a sharp increase in the control overhead, to delays in data delivery, owing to graph repairs.

A. Raoof et al. [12] provide an interesting classification of the attacks that are due to the WSN (Wireless Sensor Networks) inherited features and those designed to explicitly exploit the protocol's mechanisms or vulnerabilities. Along these lines, we briefly present a comprehensive list of the most common and disrupting attacks on the RPL protocol in the light of their origin rather than their impact, e.g., *Sinkhole* attack can degrade the quality of service in the network and eventually results in DoS to some parts of it [12].

In RPL networks, similarly to the WSNs, topology exploitation is an obvious starting point of malicious actions since packets' routing depends on the DODAG. Typical routing disruption attacks, such as *Wormhole* [15,20,21], *Blackhole* [15,22], and *Selective Forwarding* [15,23] (also known as *Grayhole*), cause network traffic loss, topology inconsistencies, and significant delays since parts of the network can get disconnected. A malicious node may either drop packets (completely or partially) or alter its standard routes once it gains an important position in the graph, e.g., a parent-node with many other nodes attached.

Other typical network attacks, like *Flooding* [24], *Replay* [25] or *Neighbor* [24] attacks, execute repetitive or falsified message-sending in order to deceive their victims and introduce inconsistencies. This subtle manipulation can yield severe topology issues and excessive energy consumption, especially in dynamic networks with mobile nodes [26]. Unlike *Replay attacks* in WSNs, which are performed with data packets, in RPL, the idea is to record legitimate control messages and forward them later.

Impersonation attacks, such as *Clone-ID* [6], or the more sophisticated *Sybil* [23] attack, are originated from a malicious node embezzling the identity(ies) of one or several legitimate(s) node(s). The goals vary from disrupting the routing topology to submitting forged data in the network or deceiving/manipulating a reputation-based/voting-based system. These types of attacks need a centralized authority to be tackled successfully [27].

Besides the above, several attacks exploit specific RPL features, such as the rank and version fields of control messages, the protocol's self-healing mechanisms, or operation modes. Rank attacks include: (i) *Decreased Rank* [28] or *Sinkhole* [23] attack, where the malicious node advertises a low-rank value to force all neighboring nodes to select it as a parent; (ii) *Increased Rank* [29,30] attack, where an adversary near to the sink advertises a high-rank value to compel all neighboring nodes to avoid it and eventually sub-optimize their parent choice; and (iii) *Worst Parent* [30] attack, where the adversary intentionally makes the worst parent selection for itself to forward the received packets via non-optimal paths. Eventually, an attacker can powerfully reshape the topology to diverge from the optimum one [31] with subsequences regarding increased traffic, high energy consumption, packet delay, and even routing loops.

DODAG inconsistencies are an ordinary situation that is normally addressed by the protocol's self-healing mechanisms as a means of nodes' s energy conservation. Unfortunately, in several cases, an adversary can take advantage of them. Well-known examples include *DODAG Version* or *DODAG Inconsistency* [32], *Global Repair* [33,34], *Local Repair* [15], *DIS message* [24,35], and *DAO inconsistency* [6] attacks. Indicatively, *Local Repair* messages from a malicious node cause all neighboring nodes to unnecessarily re-calculate their paths, causing control overhead and resource exhaustion. Even worse is the case of exploiting the *Global Repair* feature (by advertising a higher version number than the current one) to reconstruct the whole DODAG from scratch. The malicious node at the network edge may result in severe topology inconsistencies, routing loops, and delays.

The *Routing Table Overload* [24], and *Routing Table Falsification* [30] attacks resemble *Flooding* and *Replay* attacks, in the sense that an adversary sends plenty of bogus routes. The goal is to either to disorient compromised nodes, or saturate their routing tables directly and not accept legitimate DAO messages upon which correct routes can be built up. Memory depletion, packet loss, and delays are among their effects.

In the aftermath, elaborating on security issues stemming from the attacks is very challenging due to the diversity of attacks, the particularity of malicious nodes' placement in the network, and the detrimental effects of combining simple attacks, among others. Since many of the attacks share common features regarding either their origin, e.g., local repair self-healing mechanism exploitation, or their impact, e.g., irregularities in the data and/or control packets rates of the affected nodes, our proposal invests in this observation. Thus, *ASSET* accommodates a minimum set of mechanisms for anomaly and RPL specification-based detection able to address as many as 13 different types of RPL-related attacks with high accuracy and moderated cost. Next, we present and discuss *ASSET*'s details.

3. Proposed system

Here, we provide the design artifacts of *ASSET*, including its high-level architecture and details of the control channel interface. Furthermore, we describe the basic workflows for *attack detection*, *intruder identification*, and *attack mitigation*, along with the relevant incorporated mechanisms.

ASSET can mitigate a large number of attacks with a high accuracy since it exploits the softwarization paradigm in computer networks that allows: (i) centralized monitoring and control of the network; (ii) co-existence of multiple mechanisms while being extendable to support new algorithms; and (iii) consideration of both global and local viewpoints of the IoT network. For example, anomaly detection at the node (or a central) level may trigger other specification-based detection mechanisms. At a functional level, *ASSET* mainly consists of a network *Controller* with attack detection, attacker identification and mitigation algorithms, a *control channel interface* with adaptable control overhead, and *node-level features* for anomaly detection, network control and monitoring.

The *Controller* can collect information, both passively and actively, from different layers, i.e., we currently utilize network-layer and application-layer data. Such a cross-layer approach helps to maintain a detailed network view towards accurate decision-making. Attacks' mitigation is possible by mandating RPL-parameters changes in real-time, e.g., like in [36,37]. In practice, it provides a front-end to the administrator, supporting several mechanisms for detecting both the attacks and the attackers, along with and threat(s) mitigation.

The *Controller* communicates with the nodes through the *Southbound Interface*, utilizing a lightweight protocol to lookup or configure particular RPL parameters on-the-fly, monitoring the network in an adaptable fashion, i.e., trading information accuracy for control overhead, and communicating anomaly detection events from the data communication to the application plane. Such information is derived by lightweight monitoring and fast anomaly detection on a node-level, to reduce communication overhead with the *Controller*.

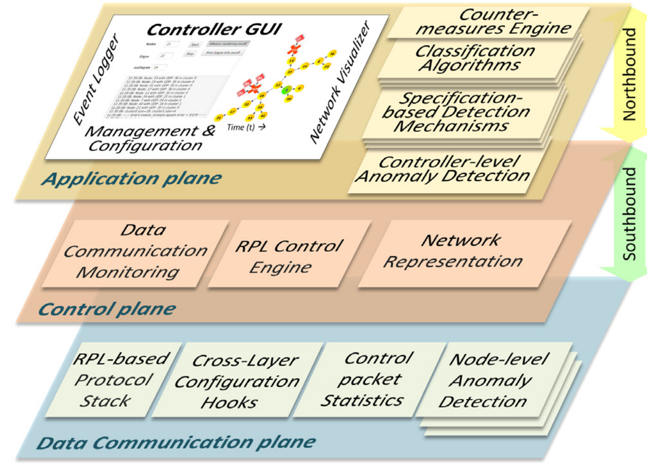

Fig. 1. The architecture of *ASSET* IDS.

The proposed IDS has been implemented in Contiki OS [38] and Java, also utilizing the Weka [39], and Graphstream libraries [40] featuring a unified workflow that embodies several mechanisms addressing multiple attacks. In practical terms, the code is under refactoring, targeting goals such as full modularity and extendability, e.g., the ability to add or replace an anomaly detection mechanism. We released the IDS as an open-source,[1] under GPLv3.0.

Regarding nodes' heterogeneity, although we used Zolertia Z1 firmware, we noticed that other node types are also compatible (e.g., Sky motes). More experiments with heterogeneous hardware and software can benefit *ASSET*.

We now detail the IDS architecture and its primary interfaces.

3.1. Architecture & interfaces

The *ASSET* IDS adopts a three-tier architecture, aligned to the SDN paradigm [41]. In Fig. 1, we depict the *Data Communication*, *Control*, and *Application Planes* as well as their main components detailed below.

The *Data Communication Plane* concerns the IoT infrastructure, including the *RPL-based protocol stack* of the corresponding nodes. We enable *cross-layer configuration hooks* to the protocol stack [36,37] allowing the *Controller* to read or apply configuration settings, e.g., to instantly enforce changes in RPL operation to mitigate attacks. Furthermore, the nodes support *control packet statistics* being either processed locally, i.e., by manifesting *per-node anomaly detection* capabilities, or communicated to the *Controller*. The *Data Communication Plane* interacts with the *Control Plane* through the *Southbound Interface*, carrying either *packet statistics* from the *nodes* to the *Controller* or configuration actions towards the opposite direction.

The other two layers, i.e., the *Control* and *Application Planes*, reside at the *Controller* and interact between each other through the *Northbound Interface*, which is REST-based. The *Control Plane* is responsible for the network control aspects, while the *Application Plane* for the IDS data analysis and GUI features.

The *Control Plane* is attached to the sink node, employing passive and active *data communication monitoring* of the nodes,

[1] https://github.com/SWNRG/ASSET.

i.e., retrieving data communication statistics from the sink or the nodes, respectively. The *RPL control engine* is responsible for enforcing particular RPL configuration processes and receiving node-level anomaly detection events from the nodes. The data communication statistics and the anomaly detection events are being communicated to the *Application Plane* through the *Northbound Interface* for further actions. Furthermore, the *Control Plane* maintains a real-time *network representation* based on the Graphstream library [40,42].

The *Application Plane* provides the GUI and configuration aspects of the IDS. It supports a real-time visualization of the IoT topology, which also designates potential IoT nodes acting as attackers. Furthermore, it provides handles to the administrator for *management and configuration* aspects of the IoT network and the intrusion detection process. Finally, it is responsible for the data analysis tasks of the *Controller*, including *controller-level anomaly detection* algorithms, *specification-based detection mechanisms*, *classification algorithms* for the attacker identification, as well as a *counter-measures engine*, being responsible for triggering attack mitigation processes, as a result of the data analysis.

We now move on to discussing *ASSET's* interfaces. Since the *Northbound Interface* is an internal interface of the *Controller*, we mainly focus on the *Southbound Interface*, which is essential for the performance of *ASSET*, especially towards reducing the involved control overhead.

3.1.1. The Southbound Interface

The *Southbound Interface* utilizes a lightweight application-level protocol that allows the *Controller* to communicate with the nodes via the sink. The protocol maintains compatibility with the RPL standard while being flexible to incorporate new features, such as a novel mechanism for mitigating a newly discovered attack. It supports either pulling of information, i.e., the *Controller* retrieving monitoring information or configuration parameters from nodes, or pushing information, i.e., the nodes notify the *Controller* regarding their monitored data periodically. The implemented protocol configuration hooks [4,5,37], based on the relevant interfaces implemented in the context of the WiSHFUL project (i.e., called UPIs), enable the *Controller* to act as a centralized network control facility, especially for enforcing attack mitigation measures.

The *Southbound Interface* is responsible for the following aspects: (i) monitoring nodes on the statistics of packets exchanged and RPL behavior, with different levels of accuracy and communication overhead, depending on the criticality of network conditions; (ii) enforcing changes in RPL protocol behavior of nodes to mitigate an attack; and (iii) communicating node-level anomaly (or specification-based) detection events—from nodes to the *Controller*—for triggering further actions. In practical terms, the interface operates in three different modes, i.e., *slim-mode*, *essential-mode*, and *full-function-mode*, described as follows:

(1) In *slim-mode*, *ASSET* operates with the minimum number of monitoring messages, being essential to construct the complete graph of the network centrally. Either the *Controller* requests the parent of a node, or the nodes are periodically reporting all parent changes. This mode is in place in networks without attack indications.

(2) In *essential-mode*, the nodes transmit to the *Controller*—besides the *slim-mode* notifications—periodic ICMP statistics, which enable controller-level anomaly detection. This mode is enabled when a node detects an attack through its node-level anomaly detection process.

(3) In *full-function-mode*, the nodes complement the previous modes with additional information, i.e., the node's rank and neighbors information for *ASSET* to detect—among others—*Rank* and *Sybil* attacks with higher precision. The *ASSET* administrator can configure and enable this mode when certain criteria are met (a given number of nodes detect an anomaly).

We now describe in detail the messages exchanged between the *Controller* and the nodes. In Table 1, we enlist all messages, and their design primitives, supported by the *Southbound Interface* and its corresponding network control and monitoring protocol. The last column depicts the specific mode they are utilized with (i.e., *slim, essential, full-function*).

In RPL, nodes collect information about their neighbors (i.e., nodes within the wireless radio coverage) and nominate a preferred parent within time instances specified by the *Trickle Timer* algorithm. This way, a network graph, i.e., the DODAG, is constructed in a distributed manner. Since this information is local, we implemented a notification feature in every node triggered by any parent-change event. In such a case, the node transmits a message to the *Controller* indicating the latest chosen parent with its rank, i.e., a [NP] message. Consequently, the *Controller* is aware of all nodes' current parent and can form the topology graph. Alternatively, the *Controller* may proactively request the node's parent information if such information is missing through a [SP] message. *Slim-mode* uses these two messages only.

Other messages from nodes to the *Controller* include the [IS], [NR], and [NN], communicating ICMP statistics (e.g., total sent and received messages), node's current rank, and available neighbors with their ranks, respectively. Whenever a node detects an outlier in its ICMP statistics, it dispatches an [AD] message. Furthermore, the [VN] and [RN] messages inform the *Controller* for a *DODAG Inconsistency* or *Local Repair* attack, detected by a node, respectively.

The *Controller* uses designated messages to: (i) solicit missing node's parent or node's neighbors' information with [SP] and [SN] messages, respectively; (ii) enable or disable ICMP statistics, and neighbor information notifications with [EI] and [NL] messages, respectively; and (iii) implement actions to mitigate attacks, including disabling *Trickle Timer* resets with [TT], blacklisting a node from becoming a parent with [BL], and disable *Local* and *Global Repair* features with [LR] and [GR] messages, respectively.

Consequently, the *Southbound Interface* enables novel *ASSET* capabilities, i.e., balancing control overhead to given network conditions and the support of multiple intrusion detection features.

In the following subsections, we elaborate on the intrusion detection workflow of *ASSET* and its corresponding mechanisms for attack detection, attacker identification, and attack mitigation.

3.2. Intrusion detection workflow

ASSET operates over the *Controller* and the IoT nodes interchangeably, as depicted in Fig. 2, offloading processes traditionally handled by the nodes to a centralized *Controller*, for a better intrusion detection accuracy and resource efficiency.

When the network runs stably, in terms of ICMP and data traffic behavior, the *Controller* collects only the active topological structure (i.e., *slim-mode*). In parallel, the nodes perform anomaly detection based on their own measured ICMP statistics. In case they detect one or more outliers, they enable the *essential-mode* of the *Southbound Interface*, i.e., start communicating the ICMP statistics to the *Controller*. Both nodes and *Controller* complementarily support RPL specification-based attack detection, like monitoring the number of recent local topology repairs and DODAG inconsistencies.

The *Controller* performs anomaly detection on data statistics to detect *Blackhole* and *Grayhole* attacks. Furthermore, it may utilize the *full-function-mode* to request additional information, such as the node's rank and its neighbors with their corresponding ranks to detect a *Decreased Rank* attack by comparing the rank

Table 1
Messages exchanged between the *Controller* and the nodes.

	ID	MESSAGE FORMAT	DESCRIPTION	M
Nodes initiated	NP	[IPv6][IPv6][int]	Node's current parent	S
	IS	[IPv6][int]	ICMP statistics	E
	AD	[IPv6][boolean]	Anomaly detection notification	
	VN	[IPv6][boolean]	Version attack notification	
	RN	[IPv6][boolean]	Local Repair attack notification	
	NR	[IPv6][int]	Nodes' current rank	F
	NN	[IPv6][IPv6 neighbors][list]	Available neighbors and their ranks	
Controller initiated	SP	[IPv6][int]	Requests the node's parent	S
	SN	[IPv6][list]	Solicits node's neighbors information	E
	EI	[IPv6 or multicast][boolean]	Enable/Disable ICMP notifications	
	TT	[IPv6 or multicast][boolean]	Enable/Disable *Trickle Timer* reset	
	BL	[IPv6][boolean]	Node blacklisted (Y/N)	
	LR	[IPv6 or multicast][boolean]	Enable/Disable *Local Repair*	
	GR	[IPv6 or multicast][boolean]	Enable/Disable *Global Repair*	
	SN	[IPv6][list]	Solicits node's neighbors information	F
	NL	[IPv6 or multicast][boolean]	Enable/Disable neighbors information	

(**M**)ode: S: Slim, E: Essential, F: Full-function.

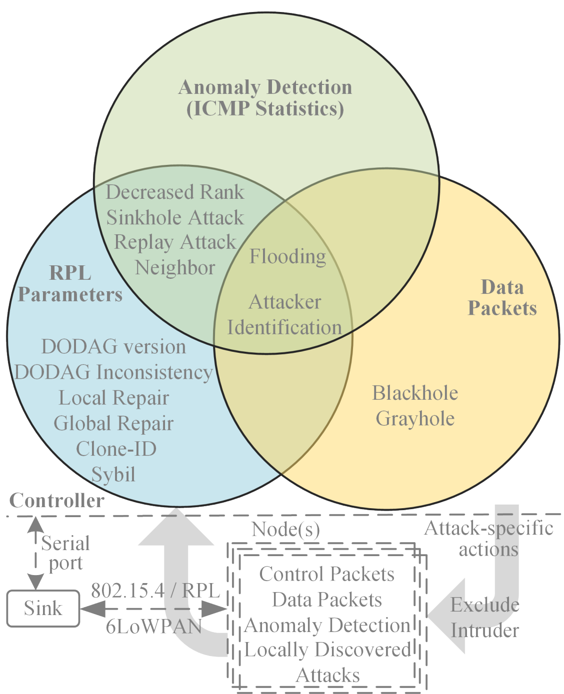

Fig. 2. An abstract view of *ASSET's*, workflow both on the *Controller* and node-level.

declared by each node with those reported by its neighboring nodes. The current version of workflow also supports the detection of *Flooding* and Replay/*Neighbor* attacks from the ICMP anomalies created and *Clone-ID* attacks by continuously comparing all nodes' IDs reported. Depending on the type of attack detected, the workflow implements an attacker(s') identification process and several attack-mitigation processes concerning identified malicious nodes, including node blacklisting, suspension of *Local Repairs*, or *Trickle Timer Resets*.

We now elaborate on the particular attack detection, attacker identification, and attack mitigation mechanisms implemented by the *ASSET* IDS workflow.

3.3. Attack detection mechanisms

ASSET exploits the distributed capabilities of RPL to enable a relatively lightweight anomaly detection on a node level, as the first line of defense. By residing on the central infrastructure, it embraces a centralized approach to provide a resource-consuming but more accurate controller-level anomaly detection process, along with several attack-specific detection mechanisms. Moreover, it utilizes RPL specification-based mechanisms to improve its capability to tackle more attacks.

The following subsections detail both anomaly detection processes and the attack-specific detection mechanisms, supported by *ASSET*.

3.3.1. Anomaly detection

ASSET is utilizing anomaly detection mechanisms without the need of training data, both at node- and *Controller*-level.

The node-level anomaly detection operates on every individual node autonomously by monitoring the ICMP messages (DIO, DAO, DIS) produced by the node. Any irregularity found is communicated with the *Controller* for further action(s). Anomaly detection at a node-level is considered rapid and efficient [43, 44], because of the locality of detected attacks. Furthermore, relevant mechanisms should be lightweight, i.e., consider the resource-constraint nature of IoT devices. We currently use a low-complexity and a memory-efficient mechanism that detects irregularities, i.e., Dixon's or Dixon-Q Test. The same method was successfully used for detecting malicious users in a cognitive radio networks setting, outperforming Grubb's and boxplot tests [45], with the limitation of considering one malicious user only. Since the Dixon-Q test runs on every node and communicates the possible outlier to the *Controller*, *ASSET* can employ Dixon-Q to detect multiple concurrent intruders. Dixon-Q is also widely used in other scientific disciplines, for example, as a method for rejecting grossly deviant (outlying) values of data sets [46]. The test assumes a normal (Gaussian) distribution of data, a typical assumption of significance tests, which was found to be true for the ICMP data produced by the nodes in random tests we conducted. The behavior of the particular anomaly detection mechanism in our results implicitly validated this assumption.

In detail, Dixon-Q test is based on calculating a Q-value defined as the ratio given by the distance of the value to be tested from its nearest neighbor, divided by the range of values. If it exceeds the tabulated critical Q-test value (i.e., called Q_{crit}) for a given Confidence Level (CL) and a number of samples N, then this value can be rejected with a probability of erroneous rejection (type I error) that is a function of the selected confidence level. For example, probabilities $p = 0.01, 0.05$, and 0.10, correspond to CLs of 99, 95 and 90 percent, since $CL = (1 - p) * 100$, named as confidence values q99, q95, q90, respectively. The test's sensitivity can be adjusted by altering the size N of data (i.e., *wsize*), along with the probability p of Type I error (or confidence level, CL). Dixon-Q test is lightweight and easy to implement for resource-constrained devices since it only needs a couple of subtractions and one division with every two newly arrived samples. For example, if the samples are 3-digit, the total added complexity is $\Theta(3)+O(M(3))log3$, which associates with negligible overhead for resource-constrained devices. Each time an outlier is detected, it is communicated to the *Controller* through the *Southbound Interface* as an "orange" alert to trigger further intrusion detection actions, such as a *Controller*-level anomaly detection process.

The *Controller* can implement more resource-consuming attack detection approaches than the nodes, however with additional control overhead, i.e., the IDS switches to *essential-mode*, allowing for a global view of the network, to investigate anomalies both in the control and data traffic. Regarding the control traffic, the relevant process is enabled whenever Dixon-Q detects an anomaly in the neighborhood of one or more nodes. *ASSET* currently employs Chebyshev's inequality [47], acting as a more accurate but also complex example, compared to Dixon-Q.

When the data distribution is unknown, Chebyshev's inequality theorem guarantees that at least $1-\frac{1}{K^2}$ of data from a sample fall within K standard deviations from the mean. This can be the basis of an outlier detection method [47] by calculating relevant lower or upper outlier detection value (ODV) limits. Any data value outside these limits is considered to be an outlier. For calculating the ODV limits, there is a need to define a p_1 threshold, trimming a small percentage of extreme values at the beginning of the outlier detection process, so outliers do not bias the standard deviation calculation. Indicative p_1 values are 0.01, 0.05, or 0.10. Additionally, a second p_2 threshold represents the expected probability of an outlier appearance. The p_2 threshold is used to determine outliers, and is usually lower than p_1, taking values like 10^{-2}, 10^{-3}, 10^{-4}. Both p_1 and p_2 control the outlier detection process's sensitivity and determine the k values for the outlier pre-filtering (first phase) and actual outlier detection (second phase) processes, respectively.

Regarding the detection of anomalies in data traffic (*Blackhole* or *Grayhole* attacks), *ASSET* monitors data packet reception based on the K-means algorithm [48] implemented in Weka library [39]. Given n measurements of nodes to be clustered, a distance measure d to capture their dissimilarity, and the number of clusters to be created (i.e., $k=2$ in our case), the algorithm initially selects k random points as the clusters' centers. It assigns the rest of the $n-k$ points to the closest cluster center (according to d). Then, within each of these k clusters, the cluster representative (also known as centroid or mean) is computed. The process continues iteratively with these representatives as the new clusters' centers until convergence. Although this is an NP-hard problem, it is simplified by heuristic algorithms to converge to a local optimum [49].

Next, we describe the specification-based mechanisms of the *Controller*.

3.3.2. Specification-based detection

To highlight the *extendability* benefits of *ASSET*, we introduce basic building blocks that can form alternative RPL specification-based detection methods, including: (i) *RPL subsystem or parameter monitoring*, which relates to *ASSET* following the behavior of RPL, reflected to particular parameters, through the *Southbound* interface, e.g., number of *Trickle Timer Resets*, nodes' rank values, etc; and (ii) a number of fixed or adaptable thresholds, indicating an abnormal RPL status, in case they are crossed. At this point, *ASSET* supports four specification-based mechanisms (i.e., *Rank Validation*, *Node ID Validation*, *Fixed Threshold F* and *Adaptable Threshold* λ based detection), which brief description follows.

A *Decreased Rank* attack is detected upon discrepancies of nodes' and nodes' parents' advertised rank via [NR] messages. More specifically, according to an algorithm introduced in [35], if a node's rank, plus the RPL stabilizing parameter *MinHopRankIncrease* [2] is lower than its parent's rank, then the latter is considered as an attacker. We also monitor all advertised ranks to be higher than the sink's rank plus the *MinHopRankIncrease*. Furthermore, the *Controller* detects a *Clone-ID* attack via a mechanism named *Node ID Validation* (Δ) to detect two nodes with the same ID.

At this point of the investigation, *ASSET* uses configurable fixed thresholds F to monitor crucial parameters at the *Controller* or node level, including the number of triggered *Local* and *Global Repairs* , and *Trickle Timer Resets*; whenever they exceed the particular thresholds, the *Controller* is notified for further attack detection actions.

Furthermore, we apply an adaptable threshold λ, which we elaborate on here. Several attacks relate to fabricated control messages causing RPL performance issues. For example, the sink-node avoids routing loops and topology inconsistencies by increasing the DODAG version whenever a global topology repair occurs. Intruders can inject continuously increasing DODAG versions into DIO messages they dispatch, causing the receiving nodes to reset their *Trickle Timer*, implement local topology repairs, and consequently face increased communication overhead. The protocol reduces the effects of such attacks by limiting the number of *Trickle Timer Resets* based on a fixed RPL threshold with the value 20. Any malformed packets, i.e., with the 'R' flag IPv6 header option set, upon reaching this threshold, are being dropped by the receiving node without triggering *Trickle Timer Resets*.

Here, we utilize the adaptable $\lambda(r)$ threshold function introduced in [32], which is more effective than RPL's fixed threshold in terms of reacting to varying attack patterns. We use a fixed threshold F at the node-level in practice, while we introduced a centralized variation of the above algorithm at the controller-level, as $\lambda(r)=[\alpha+\beta\cdot e^{1-\gamma\cdot r}]$, where $r=\frac{\sum_{i=1}^{n}E_{pkts}^{i}}{\sum_{i=1}^{n}D_{pkts}^{i}}$, $\alpha=5$, n is the number of nodes communicating packets, E_{pkts} the number of received packets with 'R' flag set true, D_{pkts} the total number of packets received. The β is chosen to lead to a default $\lambda(r)$ value of 20 (i.e., as suggested by RPL RFC [2]) and α ensures that $\lambda(r)$ cannot be zero. The value of γ, according to the authors, should be $20<\gamma<25$, i.e., we set it to value 22 in our case. Such centralized variation brings the advantage of having a λ value characterizing the whole topology, so a local attack incident leads to the corresponding protection of all nodes in the network.

In our case, the adaptable threshold λ appears more conservative compared to the one introduced in [32], since the r value reduces with the topology size. However, it produces excellent results in the particular experiments we carried out. A possible improvement could be a normalization of the equation concerning the number of nodes.

In a similar way, other mechanisms monitoring particular RPL subsystems or parameters and applying thresholds could be implemented to detect additional attacks. Right below, we proceed with the description of our attacker identification mechanism introduced here.

3.4. Attacker identification

Several attacks require identifying the intruder(s) before their mitigation, e.g., blacklisting a node causing a *Sinkhole* attack. In specific cases, intruder detection may be straightforward. For example, a duplicated ID could signify a Clone-ID attack, especially if the IDs are pre-assigned. In such cases, the recommended

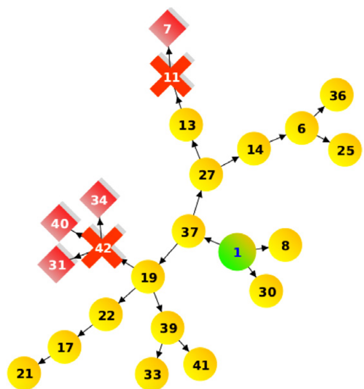

Fig. 3. *ASSET* identifies two concurrent intruders.

action could be to engage a human administrator for further steps or to mark the node that appeared second as a suspect while considering possible network delays as indications of an attack.

We propose a novel intruder identification process that can handle multiple co-existing attacks in high accuracy for other cases. Example usage of the *ASSET* platform and its GUI locating two intruders (marked with red X's) as well as the affected nodes (marked as red diamonds) is shown in Fig. 3.

In Algorithm 1 we detail the proposed attacker identification process. In particular, such a process is being triggered by the detection of an anomaly at the *Controller*-level, i.e., by Chebyshev's inequality approach (line 3). This is based on information related to the implemented monitoring mode, e.g., ICMP statistics in the case of *essential-mode*. Moreover, Algorithm 1 depicts in line 8, how the *Controller* continuously monitors each node's data packets for irregularities.

If the K-Means algorithm succeeds into *clustering the network nodes* into two groups with high confidence, the smallest group will be considered under attack (line 15). It will be further processed for *subgraph(s) division*, representing multiple co-existing attacks, i.e., defined as a clique. Here, we apply Kosaraju's algorithm [50], which locates strongly connected components as a directed graph $G = (V, E)$ in linear time (i.e., $\Theta(V + E)$ time) [51]. In particular, we utilize the Depth First Search (DFS) recursive algorithm from [51]. Our main assumption is the following. In the case of multiple intruders, the network faces several neighborhoods with disrupted regular operations. Hence, all affected nodes along with the equivalent intruders form strongly connected sub-graphs. The final step applies *root nodes identification* for each of the detected sub-graphs, i.e., representing the attacker(s) (line 17). The roots are defined as mother-vertices and located through applying the mother-vertex algorithm. The mother-vertex of a (strongly connected) graph $G = (V, E)$, is a vertex v such that a path from v can reach all other vertices in G. The algorithm has to check if v is a mother-vertex by executing DFS one more time. Consequently, the complexity of the algorithm is $\Theta(V + E) + \Theta(V + E) = \Theta(V + E)$.

As soon as one or more intruders are identified, a blacklisting process may be initiated, disallowing the attacker(s) from being part of the RPL DODAG. In the following subsection, we discuss the mitigation features supported by *ASSET*.

3.5. Attack mitigation

The final step of *ASSET* intrusion detection workflow concerns the attack mitigation. The selection of the appropriate mitigation

Algorithm 1: Intrusion Detection Process

```
Input  : Data / ICMP packets
Output: Intruder node(s) to be blacklisted
1  /* Continuously monitoring for anomalies */
2  while ICMP_Statistics do
3      if Chebyshev(ICMP packets) then
4          /* Essential mode */
5          intruder_detection(ICMP packets);
6      end
7  end
8  foreach node do
9      while new data_packets do
10         intruder_detection(data_packets);
11     end
12 end
13 Function intruder_detection(data_in):
14     /* k-means creates 2 groups of nodes */
15     if (affected_group = k_means(data_in)) then
16         affected_graphs = kosaraju(affected_group);
17         foreach (affected_graphs g) do
18             intruder = graph_mother(g);
19         end
20     end
21 End Function
```

Algorithm 2: Parent selection considering blacklisted nodes.

```
Input  : Candidate parents p1 and p2
Output: Selected parent
begin
    if (p1 && p2) in blacklist then
        return null;
    else if p1 in blacklist then
        return p2;
    else if p2 in blacklist then
        return p1;
    else
        // Standard RPL-MRHOF objective function
        return p1.ETX < p2.ETX ? p1 : p2;
    end
end
```

method to enforce depends on the detection algorithm that precedes, i.e., corresponding to particular types of attacks. In this context, *ASSET* supports the following mitigation methods:

(i) *Blacklist Intruder:* A large number of attacks can be mitigated by excluding the intruder(s) from being considered as a parent by all nodes in the network. To preserve full compatibility with the RPL standard, we implemented a node blacklisting mechanism (described in Algorithm 2) as an extension of the default OF [16]. In detail, each node maintains a local blacklisting array, which is updated by [BL] messages received by the *Controller*. Blacklisted nodes are excluded from the parent selection process, even if they appear as more suitable options, as shown in Algorithm 2. (ii) *Ignore Global Repairs and Stop Local Repairs:* Since both those mandates may consume significant resources if they are the result of an attack (e.g., *DODAG Inconsistency* attack), the *ASSET* IDS may decide to suspend one or both of them, i.e., the former at the sink, and the latter at the concerning nodes, resulting in the suspension of exchanging corresponding DIO packets. The Ignore *Global Repair* mitigation method is triggered by the [GR] message transmitted from the *Controller* to the sink. The Stop *Local Repair* mitigation method is being triggered either

Table 2
Attacks and designated actions supported by the IDS.

Categories	Description and effects of the attack(s)	DM	PS	DI	IA	AM
Topology exploitation	Cause traffic loss, topology inconsistencies or significant delays					
Blackhole	Messages to be forwarded are dropped	K	C	U	Y	B
Grayhole	Messages to be forwarded are selectively dropped	K	C	U	Y	B
Network attacks	Capture control messages and forward or replay them maliciously					
Flooding	All legitimate messages are replicated	Di,Ch	H	I,U,R	N	G,L,P
Replay	Specific control messages (i.e., DIO) are replicated	Di,Ch	H	I,R	N	G,L,P
Neighbor	Replicates control messages originated from a neighboring node	Di,Ch	H	I,R	N	G,L,P
Impersonation attacks	Steal the identity(ies) of one or more node(s)					
Clone-ID / Sybil	Pretends to be a "legitimate" node by confiscating its ID	Δ	C	I,R	Y	B
RPL specific attacks	Exploit specific RPL features					
Decreased Rank/Sinkhole	Advertises a closer to the sink position than the real one	Di,Ch,RV	H	I,R	Y	B
DODAG Inconsistency	Applies an inconsistent DODAG which forces nodes to probe neighbors	λ(C,n)	H	T,R	N	G,L,P
DODAG Version	Increases DODAG version periodically, triggering resets of network probing timers	λ(C,n)	C	T,R	N	G,L,P
Global Repair	Resets routing tables and probes all nodes, i.e, to repair topology λ(C)	C	R	N	G	
Local Repair	Nodes reset their local routing tables, i.e., triggering neighbors' probing	λ(C),F(n)	H	T,R	N	L,P

DM: Detection Method - Anomaly Detection [(Di)ixon, (Ch)ebyshev, (K)-Means], Specification Based [Adaptable Threshold (λ(C:Controller, n:node)), Fixed Threshold (F), Rank Validation (RV), Node ID Validation (Δ)].
PS: Placement Strategy - (C)ontroller, (H)ybrid.
DI: Data Input - (I)CMP Statistics, (U)DP Statistics, (T)rickle Timer Resets Counter, (R)PL Control Messages.
IA: Identification of Attacker - Y/N.
AM: Attack Mitigation - (B)lacklist Node, I(G)nore Global Repairs, Stop (L)ocal Repairs, Sto(P) Trickle Timer Resets.

locally or through the [LR] message sent from the *Controller* to the corresponding node(s).

(iii) *Stop Trickle Timer Resets:* Equivalently, the *Trickle Timer Resets* cause significant control overhead since RPL control messages are being exchanged more frequently. A Stop *Trickle Timer Resets* mitigation method can either be triggered locally or from the *Controller* ([TT] message) allowing for the node(s) to ignore all *Trickle Timer Resets*, for a particular period.

3.6. Summary

In Table 2, we summarize how all the above IDS features are associated with all handled attacks, including their brief descriptions. More specifically, we enlist for all attacks: (i) the detection method applied (i.e., whether it is anomaly detection or specification based) as well as the specific detection features utilized; (ii) the placement of the detection method, i.e., at the *Controller* only or also at the nodes (hybrid); (iii) the required data input for the particular detection method; (iv) whether the identification of an attacker is needed for its mitigation; and (v) the mitigation method which is appropriate to this type of attack.

The table highlights that *ASSET* handles diverse types of attacks through different combinations among the supported IDS features. We note that anomaly detection can even detect unknown attacks causing communication disruptions. Furthermore, new specification-based building blocks can be integrated to increase its supported number of attacks further. Although the IDS could be implemented with different relevant algorithms performing even better, our selection performed decently in our experimentation exercise and enough to validate the main *ASSET* novelties.

Moreover, in Fig. 4, we illustrate the threat model [52,53] we consider in this work, i.e., which is a visualized analysis of network security breach strategies, along with our IDS's matching mitigation techniques. To establish this risk assessment, we begin by pinpointing the assets upon which the RPL network's mission is based. Next, we explore the potential threats in high and low risk, originating either from malicious actions or known RPL weaknesses, i.e., due to RPL's constrained nature. Finally, we complete the model by introducing the IDS's defenses serving as a shield from threats and vulnerabilities.

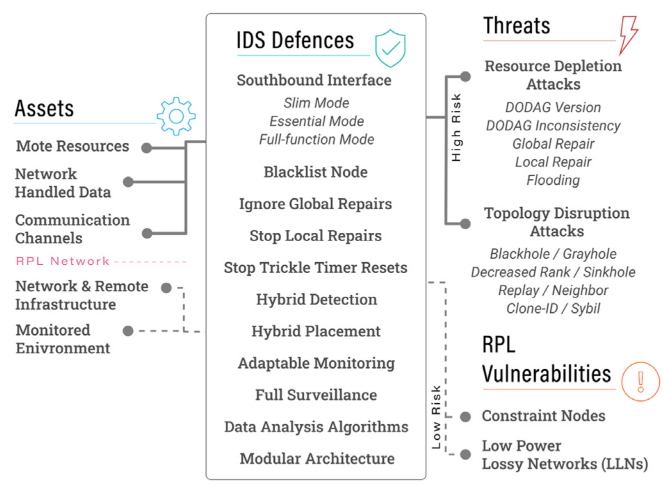

Fig. 4. Threat model.

4. Evaluation results

We evaluate *ASSET* in line with *robustness* and *extendability* that reflect the *width* of our solution, as well as *accuracy* and *mitigation-time* that express its *depth*. More specifically, we begin with discussing our evaluation methodology and, then, we present: (i) proof-of-concept simulation results that demonstrate attack incidents, along with *ASSET's* response in terms of detection and mitigation, as well as attacker's identification; and (ii) the *ASSET's* robustness with an evaluation of its operation under a range of attacks triggering all discussed mechanisms.

4.1. Evaluation methodology

For the *ASSET's* performance evaluation, we utilize the Cooja emulator in Contiki OS [38]. The simulations carried out are considering one sink node, a set of legitimate nodes, and one attacker node. Although *ASSET* can potentially mitigate attacks caused by multiple malicious nodes, we left the relevant experimentation as future work. The network setup parameters are described in detail in Table 3.

Table 3
Network setup parameters.

Parameter	Value	Notes
Network layer	RPL	Storing mode
MAC layer	802.15.4	
Implementation	Contiki 3.0 - Cooja	
Sink node(s)	1	Serial Port Connection
Mote type	Zolertia Z1	
Nodes placement	Random	
Number of nodes	25 or 50	
Area	800 m × 800 m	
Simulated time	3 h	10,800,00 ms
Data (UDP) transmission period (P)	5 min	Unless otherwise stated
ICMP probing frequency	5 min	Avoiding zero probings
Packet size	70 B	Average size
TX range	50 m	
Interference range	50 m	
TX/RX success ratio	100%	
Trickle timer duration	4 ms–17.5 min	Contiki RPL defaults

We only consider attacks where the intruder is part of the active RPL topology i.e., responds promptly to the *Controller*'s solicitation messages, e.g., it would be rather trivial for an IDS with centralized components to detect and, consequently, blacklist as possible intruder a node that does not respond to such messages. Once being blacklisted, the intruder cannot be chosen as a parent-node, and hence, it cannot successfully launch most of the RPL attacks described in Section 2.2. In practice, we consider that the attacker node(s) are running multiple modified Contiki OS versions[2] (also available under GPLv3.0) to execute one or more attacks in conjunction. Right afterward, we present proof-of-concept results demonstrating *ASSET's* operation under various attacks.

4.2. Proof-of-concept results

To evaluate the different aspects of *ASSET* and reveal the potential of its mechanisms, we conducted several experiments, as presented below. Those proof-of-concept experiments focus on demonstrating *ASSET's* functionalities along with the required *width* and *depth*. Comparing *ASSET* with other similar solutions is considered as a future work since (i) we have to identify common use-cases in terms of required security level and affordable control overhead or processing cost; and (ii) we have to determine the type of involved mitigation action and its impact since this determines the communication or performance issues that a false positive can cause.

4.2.1. Detection mechanisms evaluation

The first proof-of-concept simulation is associated with anomaly detection mechanisms of *ASSET*. As illustrated in Fig. 5, we consider a network with 50 nodes (marked with yellow) randomly placed around the sink-node (the green one), while an intruder (ID = 54, purple color) compromises the network by unleashing a *Decreased Rank* attack advertising a lower rank value than all other legitimate nodes in its wireless coverage (i.e., the green range). As a result, most of the nodes within range, i.e., nodes with ID 27, 32, 33, 42 and some others around it, i.e., nodes with ID 4, 17, 44, increase the number of ICMP packets exchanged, in their effort to recalculate paths to the sink.

The Dixon-Q test mechanism in every node detects the anomaly in the number of ICMP messages sent and received, as shown by the *PANIC* entries in the log file illustrated in the right-hand window in Fig. 5. In our simulation, we configure the

[2] https://github.com/SWNRG/contiki-malicious.

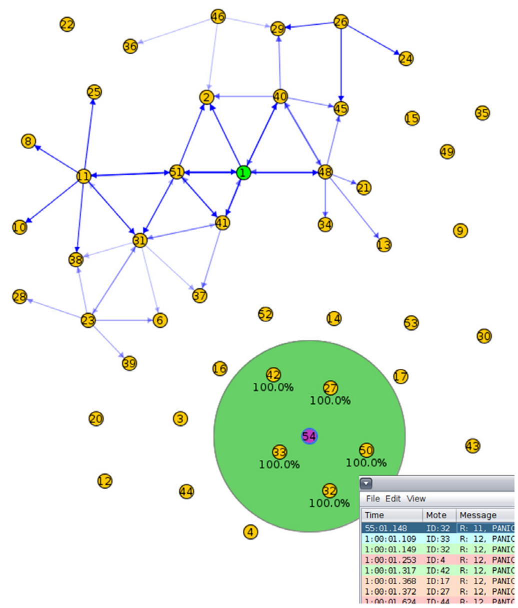

Fig. 5. An RPL network under *Decreased Rank* attack.

Table 4
Node-level anomaly detection: Dixon-Q test, $wsize = 7$.

ICMP	NODE	t_6	t_5	t_4	t_3	t_2	t_1	t_0
SEND	4	4	4	4	5	4	4	18
	17	5	2	5	3	3	4	15
	27	5	3	6	4	4	5	19
	32	4	4	4	3	6	4	19
	33	7	4	6	5	7	7	17
	42	8	7	6	6	9	8	13
	44	3	5	3	3	4	5	8
RECV	4	3	4	3	1	5	4	39
	17	12	5	4	5	5	4	42
	27	10	6	5	4	4	6	82
	32	9	4	2	3	3	3	64
	33	11	6	5	5	7	6	91
	42	6	6	5	5	9	8	58
	44	4	3	3	7	3	3	20

Dixon-Q window-size as $wsize = 7$. Table 4 shows for each of the above nodes that the latest of seven values, regarding both the incoming (RECV) and outgoing (SEND) ICMP packets, is an outlier, causing seven nodes to dispatch the [AD] message at t_0 (nodes within the attacker's range are with gray background in Table 4). Since the number of nodes sending a [AD] message exceeds the threshold of three, *ASSET* activates controller-level anomaly detection by Chebyshev's inequality mechanism for further investigation of the attack instance, i.e., attacker's detection and mitigation.

4.2.2. Control overhead & power consumption

The holistic approach provided by *ASSET* is illustrated in Fig. 6 which is the outcome of our second proof-of-concept simulation. In practice, we simulated for three hours (x-axis) a multi-hop network with 25 nodes randomly placed around one sink, considering a combination of *Decreased Rank* and *Blackhole* attacks, and we observe the network's control overhead to validate our intuition regarding the impact of attacks over it. Fig. 6 shows that attacks are launched at 01:20 hour (vertical red line), detected at 01:32 hour (vertical yellow line), and mitigated at 01:47 hour (vertical green line).

We chose a typical combination of attacks. The intruder-node discards data packets, e.g., UDP, once it successfully deceives

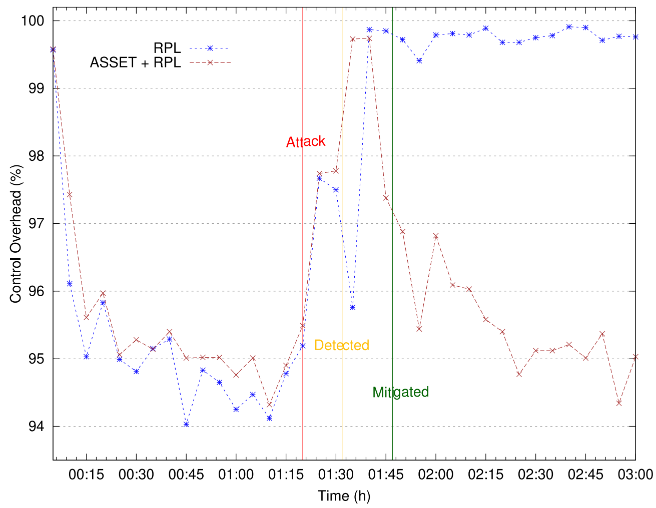

Fig. 6. Control overhead over time for a combined *Decreased Rank* and *Blackhole* attack on a network of 25 nodes.

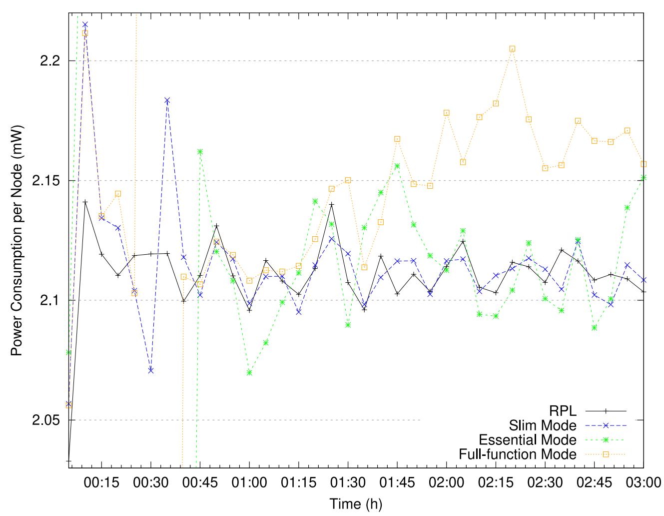

Fig. 7. Average power consumption of nodes under *ASSET's* different modes of operation.

several nodes that choose it as a routing node (i.e., parent) for their packets. Fig. 6 does imprint the impact of the *Decreased Rank* attack, which precedes the *Blackhole* one. Once the attack has taken place, the Dixon-Q test detects outliers in control packets on six nodes at 01:25 hour and three more nodes at 01:30. These nodes notify the *Controller* with [AD] messages, activating the Chebyshev's inequality mechanism for a more fine-grained detection. For this purpose, apart from a [NP] message, nodes also dispatch their latest chosen parent-node, i.e., ICMP statistics ([IS] messages), node's current rank ([NR] messages), and available neighbors ([NN] messages), assisting the *Controller* in identifying the intruder. Once the intruder is identified, the *Controller* at 01:32 dispatches a [BL] message to all nodes as a mitigation action. Fig. 6 provides evidence that, at 01:47 hour, the network graph is concise again, i.e., network nodes selected legitimate parents, after excluding the attacker as a candidate parent.

Regarding power consumption, we conducted the same experiment under four different modes of operation, i.e., standard RPL compared with the three operation modes of *ASSET* (i.e., slim-, essential-, full-function-modes). The results are presented in Fig. 7, where after the initial anticipated initial power "spikes"

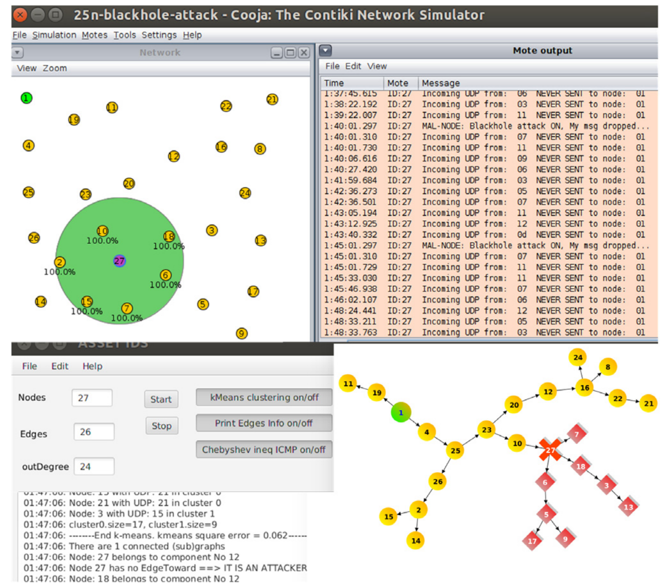

Fig. 8. An RPL network under *Blackhole* attack.

until the network settles down, the power consumption is minimal, with only full-function mode consuming slightly more energy. In total, compared with RPL, the slim-mode consumes 0.18 percent more power per node, the essential mode consumes 0.71 percent, while the full-function mode consumes 1.54 percent more energy. Compared to other similar solutions, SVELTE [43] has a 30 percent overhead compared to RPL.

4.2.3. ASSET's *modes of operation*

Moreover, Fig. 6 confirms that *slim-mode* operation of *ASSET* does not overload the network. In the period from the beginning of the simulation until the attacks (vertical red line), *ASSET* operates with the minimum number of monitoring messages, i.e., [NP] messages from nodes to report parents' changes and/or [SP] messages from the *Controller* to the nodes, requesting missing information regarding their parents. The purple curve, corresponding to the RPL network with the IDS functionality, is only slightly higher, i.e., 6.28 percent on average in our simulation, compared to the blue line, representing the standard RPL operation.

The *full-mode* operation of *ASSET* succeeds in the attacker's identification and mitigation at the cost of increased control overhead. However, this overhead remains lower, 49.87 percent on average, than when the RPL protocol is left unshielded. Indeed, within the time frame between the red and green verticals, node and controller-level anomaly detection are taking place, additional messages ([IS], [NR], and [NN]) are sent to the *Controller*, who then activates the three steps described in Section 3.4 to identify the attacker. However, despite these demanding processes, *ASSET* controls network topology disruptions and updates, moderating *Local* and *Global Repair* ([LR] and [GR] messages) and, thus, holding the peak in the purple curve.

Finally, mitigating the attack brings as much as 95.96 percent benefit to the network in control overhead. In the period from the attacks' mitigation (vertical green line) until the end of the simulation, *ASSET* manages to establish a new DODAG consisted of legitimate nodes while allowing the network to continue its mission, i.e., data gathering.

4.2.4. *Attacker's identification*

Our last proof-of-concept outcome elaborates on the attacker's identification mechanism. In Fig. 8, in a three-hour run, we operate another random, multi-hop topology (illustrated on the up-left part), where 25 nodes (the yellow ones) are under *Blackhole* attack by the purple node (ID = 27), while they route their data packets to the sink (green node). The intruder is placed within the direct reach of six nodes (ID 2, 6, 7, 10, 15, 18) and presents a legitimate behavior until 01:20 hour when it starts dropping all received data packets in their routing towards the sink (including the attacker's own ones to make the scenario more challenging).

In a network with scheduled UDPs and a pre-defined dispatching period, the impact of a *Blackhole* attack is to differentiate affected by non-affected nodes in terms of the UDP packets number arrived at the sink. Indeed, the K-Means algorithm running in the *Controller* has successfully divided the network into two distinct groups, i.e., clusters 0 and 1 (bottom left window), also illustrated in the right part of Fig. 8, i.e., cluster 0 contains the yellow nodes along with the sink (non-affected as indicated by the high number of UDP packets). In contrast, cluster 1 shown in red, consists of the affected nodes (due to the low number of UDP packets).

A closer look at the affected sub-graph reveals that only nodes 6, 7, and 18 within the intruder's coverage are affected by the attack. In contrast, the other three ones, i.e., 2, 10 and 15, are not affected because they do not select the intruder as a parent (indeed, the parent of the nodes 2,15 is node 26, while the parent of node 10 is node 23). Simultaneously, nodes 3, 13 and 5, 9, 17 select as a parent the affected nodes 18 and 6, respectively, and consequently are also influenced by the *Blackhole* attack, although they are not within the intruder's coverage.

At this step, it is crucial to distinguish among cluster members to identify the malicious one. K-means feeds Kosaraju's algorithm with the red sub-graph. Kosaraju then defines one sub-graph (or more, in case of multiple attacks) and passes the graph to the mother node algorithm. The algorithm recognizes node 27 as the "root" of this sub-graph, identifying this ID as the malicious node. In our simulation, the attack begins at 01:20 hour, and our system recognizes the attacker at 01:47 hour. Right afterward, the *Controller* blacklists this node to not be selected as a parent node.

In this scenario, we noticed that leaving unmitigated such an attack reduces the packets that the sink successfully received by as much as 17.3 percent. Our system helps the network lose only 5.7 percent of the packets that would eventually arrive at the sink in a non-attack case.

Next, we carry on discussing the results on the robustness of *ASSET*.

4.3. Robustness results

Our results regarding ASSET's robustness are summarized in Table 5 and show that our proposed system can handle 13 attacks. We excluded from our analysis *Sinkhole*, *Neighbor*, and *Sybil* attacks due to their high similarities with *Decreased Rank*, *Replay*, and *Clone-ID* attacks, respectively. Moreover, *Decreased Rank* and *DODAG Inconsistency* attacks appear twice in the Table to highlight how alternative mechanisms can handle them.

Each row of Table 5 represents a three-hour simulation, divided into 5 min time-slots, regarding the same 25-nodes' network. The first two rows refer to Chebyshev's and Dixon's operations in case of non-attack. In contrast, each of the rest rows represents a type of attack (1st column), occurring at the 80th min, along with the detection mechanism (2nd column) in place.

Regarding basic implementation details and configurations, in *Blackhole* attack, the malicious node suspends forwarding of all UDP data packets traveling towards the sink. In contrast, for *Grayhole* the attacker decides to forward or not the received data packet based on a fair coin toss. In *Decreased Rank* attack, a malicious node is advertising a fake rank calculated after subtracting four times the RPL's parent switching threshold (*MinHopRankIncrease*) from the attacker's actual rank (i.e., *fake_rank = actual_rank - 4*MinHopRankIncrease*). For *DODAG Version* attack, an adversary keeps sending DIO messages with increasing version numbers, triggering continuous *Trickle Timer Resets*, in addition to *Global* and *Local Repairs*. *DODAG Inconsistency* attack is applying erroneous headers in RPL messages [32] triggering also *Trickle Timer Resets*, *Global* and *Local Repairs*. *Global* or *Local Repair* attacks, are replicated with a *DODAG Inconsistency* attack. *Flooding* attack was implemented with the attacker continuously dispatching forged RPL & data packets, limited by Cooja processing capabilities since a high communication load crashes the (emulated) serial port. We implemented the *Replay* attack in a similar way to *Flooding* attack by assuming an adversary continuously re-sending the RPL messages it receives. Finally, the *Clone-ID* attacker duplicates existing RIME, MAC, and/or IPv6 addresses, i.e., leading to duplicated node IDs.

The specific attack detection mechanism employed for each attack is also indicated in Table 5. Chebyshev's inequality's and Dixon's settings are $wsize = 8, p_1 = 0.95$ and $wsize = 5$, $confidence = q99$, respectively. The configuration of threshold F was set to 10 (half of the one proposed by RPL, assuming a hostile environment), and adaptable λ is implemented as defined in Section 3.3.2. These mechanisms operate both on the node and *Controller* side, depending on the attack type. K-Means confidence was set to 0.1.

The central cells in Table 5 indicate the number of nodes signaling an attack at the given time-slot, based on the mechanism referenced in the particular row. We indicate with bold the time-slot that attacks start, e.g., we selected slot 16 on 80th min for all different cases. We color differently the cells where the attacks are detected (gray) and mitigated (dark gray-white fonts), as well as those reflecting false positives (light gray). Single nodes cause a few false positives. As previously discussed, an event is considered an attack when at least three nodes declare its detection, except for *Clone-ID* and *Global Repair* attacks, because the corresponding mechanisms do not cause false positives, e.g., the *Global Repair* attack is being handled at the sink only. Moreover, regarding *Decreased Rank* detection, although four rank inconsistencies are reported in time-slot 18, the dedicated RV mechanism needs to mandate the nodes to enable *full-function mode* to send all neighbor's data (i.e., [SN] message) and compare all declared ranks for discrepancies before identifying the attacker.

We consider an attack as mitigated when the proper mitigation action is enforced, independently of the time it takes. An indication of the latter appears in Table 5 through the declining number of nodes signaling the attack immediately after the mitigation time-slots. Once we described our notation, we proceeded with our observations based on each row's results.

The first two rows consider simulations without attacks to highlight the overhead of *ASSET* during regular system operation. On the one hand, Chebyshev's inequality did not produce any false positives. However, we had some rare false positives with more relaxed confidence levels (e.g., $p_1 = 0.90$) without triggering attack detection. On the other hand, the Dixon-Q test faces 5 cases of single-node detecting outliers, e.g., node 22nd on time-slots 23, 24, and 25. We also note that Dixon-Q detects some infrequent outliers even after an attack is mitigated since the network settles down progressively. This causes a minor communication overhead increase in the particular nodes, i.e., enabling the transmission of ICMP statistics to the *Controller*, and highlights that *ASSET's* control overhead adaptability aspects require further investigations, which we consider as future work.

Blackhole and *Grayhole* attacks impact data rather than control packets. We employ the K-Means algorithm, which continuously clusters the nodes into two groups based on their UDP packets

Table 5
ASSET's robustness evaluation.

Time (180 min)		*5*	*10*	*15*	*20*	*25*	*30*	*35*	*40*	*45*	*50*	*55*	*60*	*5*	*10*	*15*	***20***	*25*	*30*	*35*	*40*	*45*	*50*	*55*	*60*	*5*	*10*	*15*	*20*	*25*	*30*	*35*	*40*	*45*	*50*	*55*	*60*
Time-slot		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	**16**	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
No Attack	**DM**																																				
Chebyshev's Inequality	Ch	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Dixon-Q Test	Di	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0	0	0	0	0	1	0	0	0	0	0
Attack																																					
Blackhole	K	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	**0**	0	0	1	2	4	5	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Grayhole	K	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	**0**	0	0	0	0	1	1	2	4	5	0	0	0	0	0	0	0	0	0	1	0
Decreased Rank	RV	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	**0**	0	4	5	5	2	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
Decreased Rank	Ch	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**	2	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DODAG Version	$\lambda(C, n)$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**	15	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DODAG Inconsistency	λ(C,n)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**	16	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DODAG Inconsistency	Ch	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Global Repair	$\lambda(C)$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Local Repair	$\lambda(C)$,F(n)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**	11	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Flooding	Ch	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**	2	5	9	10	2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
Replay	Ch	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**	0	7	11	12	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Clone-ID	Δ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Attack initiation
False Positives
Attack Detection
Attack Mitigation

Ch: Chebyshev's Inequality, Di: Dixon-Q Test, K: K-Means
λ: Adaptable Threshold, F: Fixed Threshold, RV: Rank Validation, Δ: Node ID Validation
C: Controller, n: node

arrived at the sink. We consider a true positive whenever a small cluster with nodes present a low number of UDP packets, i.e., assuming that the attack does not impact most nodes. Consequently, the sporadic false positives do not cause any issue. We noticed that topology-size and severity of attack impact false positives and attack mitigation time. For example, it takes three more time-slots for *ASSET* to mitigate the less severe *Grayhole* attack, compared to *Blackhole*. Such issues deserve a dedicated analysis.

Regarding the *Decreased Rank* attack, we provide results for both Rank Validation and Chebyshev mechanisms. The former needs four time-slots until its mitigation time, while the latter can detect the attack in just two time-slots. However, Chebyshev is not equipped to mitigate this particular attack. In this execution, RV is characterized by two false positives, before and after the attack, without impacting the attack detection process. These results highlight the need for dedicated specification-based mechanisms.

DODAG Version attack is mitigated within two time-slots because of frequent DIO packets with increasing DODAG versions. In the first and second time-slots, the adaptable λ thresholds are being crossed at the node- and controller-levels, respectively, i.e., the latter confirming the attack detection. We have an equivalent result for *DODAG Inconsistency* attack since their outcome is similar, given the attacker's same spatial position. Here, we mitigate the attack's outcome, i.e., suspend resetting *Trickle Timer*, *Global*, and *Local Repairs* since identifying the attacker requires additional software or equipment [25], considered out of the paper's scope.

We also provide the outcome of Chebyshev's mechanism in the case of *DODAG Inconsistency* attack, highlighting its inability to detect the latter and the advantages of *ASSET's* specification-based mechanisms. We note that Chebyshev with a lower sensitivity (e.g., $p_1 = 0.90$ and the same $wsize$) can detect the attack at time-slot 20 and mitigate it at 21, i.e., later than the adaptable λ. Such aspects highlight that anomaly detection and specification-based mechanisms can be operating in a parallel manner, complementing each other.

In the case of *Global Repair* attack, *ASSET* needs three time-slots to mitigate it (i.e., the sink ignores further *Global Repair* mandates). This process involves the communication of nodes with the sink and the follow-up involvement of the *Controller*. The mitigation time is shorter by one time-slot for *Local Repair* attacks, where nodes signal an attack as soon as their fixed threshold F is reached, which is confirmed by the *Controller* with its adaptable threshold λ.

It takes four time-slots for *ASSET* to mitigate both *Flooding* and *Replay* attacks because of the gradual control traffic increase among the nodes. One node detects an outlier for the *Replay* attack at the 28th time-slot, which is ignored by the *Controller*. Mitigation for both attacks involves disabling *Global* and *Local Repairs*, as well as *Trickle Timer Resets*. Since Cooja faces stability issues with these two attacks, conducting these experiments in a test-bed environment and studying the network's behavior under real network conditions is another open issue.

Clone-ID attackers are rapidly identified by the *Controller* with 100 percent accuracy, due to the centralized nature of *ASSET*, i.e., nodes with duplicated IDs are immediately detected and black-listed. *Sybil* attacks will also be equivalently mitigated.

The above results demonstrate that *ASSET*, under the given scenario, configuration settings and network conditions: (i) can detect 13 attacks (i.e., including *Sinkhole*, *Neighbor*, and *Sybil* attacks that exhibit a very similar behavior with *Decreased Rank*, *Replay*, and *Clone-ID*, respectively) without false positives in attack detection, i.e., we noticed only some rare false alarms from nodes to the *Controller*; (ii) handles effectively the infrequent false alarms due to the requirement that at least three nodes should signal an attack before a mitigation action being triggered; (iii) employs multiple attack detection mechanisms, including three anomaly detection and four specification-based, contributing to both *width* and *depth* of attack detection; (iv) mitigation time depends on the attack type, severity, and behavior; and (v) manages to identify and exclude the attackers for *Blackhole*, *Grayhole*, *Decreased Rank*, and *Clone-ID* attacks, while for the rest of them it mitigates the outcome of the attack, i.e., the attack may still be present.

Due to our experiments' high complexity, we consider a more thorough investigation of *ASSET's* performance, including its statistical evaluation and comparison with other similar solutions, as future work. However, we argue that the current results suffice to confirm *ASSET's* novelties, as defined in the paper.

4.3.1. *Open* ASSET *vulnerabilities*

Here, we discuss several *ASSET's* security vulnerabilities that are outside the scope of this paper and deserve further investigation. These open challenges can be summarized as follows.

For simplicity, we currently assume that *ASSET Controller* and corresponding communication (e.g., packets carrying measurements from nodes to the *Controller*) is safe and not tampered. For example, attacks oriented to Software-Defined IoT solutions could be relevant to *ASSET*, e.g., targeting a centralized *Controller*.[3] Consequently, there is a need for hardening the related security. Several techniques could be potentially applied, including Byzantine Fault Tolerance [54], n-versioning, or secure tokens and enclaves. Moreover, a sophisticated attack could possibly tamper with the measurements traveling to the sink to "hide" an ongoing attack or to work around an *ASSET* mechanism. This may be challenging for *ASSET* since it operates many attack detection mechanisms in parallel, i.e., another one may detect the attack. We consider such aspects complementary with our solution but complicated enough to deserve an independent study.

Furthermore, our proposal may be vulnerable to more sophisticated attacks than the considered ones. For example, neighboring nodes may collude to exclude nodes from the graph or apply a Clone-ID attack after collapsing the node to be duplicated. In the latter case, reputation-based mechanisms can be implemented as a scheme with multi-path duplication of messages, i.e., to verify node's compliance. Although this is always the case with IDSs, we consider *ASSET* as a descent solution to many different attacks, in contrast to the related works.

5. Related works

In the context of RPL, the associated IDSs gain popularity following the protocol's evolution [7,12,14,55]. Literature classifies these RPL-related IDSs according to two main criteria [56]: (i) the detection method they employ, and (ii) their placement strategy. Based on the detection method, the IDSs are distinguished in: *signature detection*, *anomaly detection*, *RPL specification-based* systems, while *hybrid detection* IDSs combine at least two of the aforementioned categories. Regarding their placement strategy, RPL-related IDSs are classified into: *centralized*, *distributed*, and *hybrid placement* systems; the latter that blend the rationale of centralized and distributed by keeping the "heavy" tasks for the root or central node(s) and delegating the more lightweight ones to the rest.

In our survey paper published in 2021 [14], we have investigated the 22 most recently introduced RPL-related IDSs in the

[3] Although *ASSET* adopts ideas originating from the SDN world, the scope of this paper covers RPL-related attacks only, rather than the security of SDN IoT systems.

Table 6
Comparative overview of existing hybrid IDSs related to our work.

IDS	DM	EE	NA	E	IA	AM
[43]	AD, SB, SD	S	7	Y	Y	WE
[57]	AD, SB	S	3	Y	Y	N
[58]	AD, SD	S	5	–	N	N
[44]	AD, SD	S	3	Y	N	N
[59]	AD, SD	C	8	Y	Y	MF
ASSET	AD, SB	S	13	Y	Y	MM

DM: Detection Method - Anomaly Detection (AD), Specification-Based Detection (SB), Signature Detection (SD).
EE: Evaluation Environment - (S)imulation, (C)onceptual.
NA: Number of Attacks.
E: Extendability - Y/N.
IA: Identification of Attacker AM: Attack Mitigation - White List Exclusion (WE), Mini Firewall (MF), Multiple Methods (MM).

literature (2013 – 2020) and concluded the outcome that combining detection methods as well as placement strategies brings positive results. The competetive advantage was found to be the number of attacks the system detects; this ranges from three to five (3 to 5) for the hybrid detection systems [44,58] and goes up to eight (8) for the full hybrid ones [43,59]. Table 6 provides a brief comparative overview of hybrid systems, which are found the most advanced of the recent literature [14] and relevant to our proposed one.

Further benefits include the ability of some systems to identify the attacker [57,59] and/or mitigate the attack [43,59], the extendability as a feature that enables the IDS evolution towards detecting new attacks, as well as the detection accuracy rate in conjunction with low resource overhead, especially when the developed mechanisms are appropriately located both in central and distributed nodes.

In particular, appropriately tuning the parameters of *SVELTE* [43] can offer as much as 100 percent of detection accuracy and zero false positives. However, the system trades its advantages with resource requirements regarding storage, the signatures' repository, and computational power for anomaly detection algorithms. In comparison, Bostani et al. [57] show an average of 93.3 percent accuracy with less than 3.3 false positives for multiple runs.

Game Theory IDS [58] reports an average of 98.6 percent accuracy and less than 2.5 percent of false positives for a variety of setups. In comparison, *CHA–IDS* [44] shows an accuracy within 85.2 – 100 percent and up to 0.058 percent false positives, in the worst case. Although they keep a good balance between accuracy, false positives, and overhead, they neither deal with the attacker's identification nor with mitigation actions. These limitations probably stem from the fact that *Game Theory IDS* employs a distributed placement strategy not taking advantage of the results of a central analysis, and vice versa, *CHA–IDS* is a centralized system, not exploiting distributed mechanisms. Indeed, in the case of [59], signature and anomaly detection are used in combination, exploiting, further, the rationale of a hybrid placement strategy. The system brings a high score of as many as 8 attacks detected.

Comparing the above hybrid systems is a challenging and not straightforward task since it is associated with the considered use-case in terms of required security level and reasonable control overhead or processing cost, depending on how an IDS covers the addressed attack(s). Our literature study reveals that different approaches span from simulating all or some of the attacks to conceptually supporting coverage for all or subset of the attacks under invistigation. Indicatively, authors in [59] introduce a full-conceptual framework, where they discuss but do not evaluate their IDS. Also, in the case of simulation approaches, differences concern the simulation environments and the metrics used to assess the IDSs' performance. Among different approaches, Contiki Cooja [38] is a common choice; it is also adopted in our work.

Another challenging issue considering comparison is the lack of a common framework for IDS evaluation in real environments, i.e., test-beds. This challenge is reflected in 3rd column of Table 6 which shows that all approaches with evaluation results use simulation. Our previous experience with test-beds participating in the FED4FIRE [60] and GENI [61] federations, in the context of 5G network slicing research [62–64], shows that it would be interesting, but also very challenging, to deploy complete IDSs in test-beds for evaluation reasons and address possible issues that arise. Currently, the Sharing Artifacts in a Cybersecurity Community Hub (SEARCCH) project [65] offers a facility that provides validation, repeatable sharing, and reuse of security-related research results. A relevant initiative for IoT security could establish a common framework where open-source IDS code could be released and comparatively evaluated, e.g., in a common environment with the same methodology and evaluation scenarios.

In this work, we exploit observations derived by the recent bibliography to develop a novel softwarized IDS by-design, in the sense that it assigns lightweight tasks, such as monitoring and first-place detection, to the constraint end-nodes and transfers the demanding tasks to central premises. Besides, *ASSET* follows a modular architecture that allows adaptations and/or extendability. It combines anomaly and specification-based detection and, to the best of our knowledge, is the most robust system compared to its peers. It detects 13 RPL-related attacks, supports attacker's identification, and offers several mitigation actions depending on the attack detected.

Conclusion

ASSET's evaluation has shown that handling attacks against the RPL protocol is challenging and highly dependent on the implemented mechanisms targeting one or more specific attack(s). Moreover, transferring node-level functions to the centralized infrastructure is more stable and accurate and provides new capabilities to the network administrators. Some attacks can be handled with high accuracy, while some can be mitigated, leaving the identification of the intruder as an open issue. In addition, inspired by the softwarization paradigm, by offering centralized intelligence and extendability, *ASSET* is an ideal platform for new mechanisms and tools to be tested in the areas of anomaly detection and SDN-like solutions for RPL and the IoT in general.

ASSET exhibits the following advantages: (i) a holistic workflow handling 13 well-known RPL-related attacks; (ii) 3 anomaly and 4 specification-based attack detection mechanisms, operating both at node and controller-level and exhibiting a low number of false positives; (iii) a set of alternative mitigation actions and an original attacker identification process; and (iv) an adaptable control and monitoring protocol, trading communication overhead for attacker detection accuracy.

Our next steps include the following aspects: (i) to further improve (i.e., in *width* and *depth*) the attack detection and mitigation, the attacker identification mechanisms, as well as the control channel adaptability, including employing change-point analysis for anomaly detection [66,67], (ii) to conduct extensive experimentation with multiple attacks (also co-existing), attackers, topology structures and sizes, experiment configurations, including based on real IoT test-beds, to accurately measure the implications of ASSET to network latency among others, (iii) to incorporate a separate control channel with a long-range interface, inspired by [68,69], which can significantly improve *ASSET's* operation, in terms of communication overhead and attack mitigation capability, (iv) to assess the node's mobility and wireless interference impact and how they can affect attack detection since it can also increase control overhead, e.g., they may cause false positives in anomaly detection.

CRediT authorship contribution statement

George Violettas: Conceptualization, Software, Original draft preparation, Investigation, Writing. **George Simoglou:** Visualization, Data curation, Writing. **Sophia Petridou:** Writing, Validation. **Lefteris Mamatas:** Methodology, Writing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Kyriakos Vougioukas provided the testing framework[4] for Dixon-Q and Chebyshev's Inequality tests.

References

[1] M. Wollschlaeger, T. Sauter, J. Jasperneite, The future of industrial communication: Automation networks in the era of the internet of things & industry 4.0, IEEE Ind. Electron. Mag. 11 (1) (2017) 17–27.
[2] T. Winter, et al., RPL: IPv6 routing protocol for low-power and lossy networks, RFC 6550 (2012) 1–157.
[3] O. Gaddour, A. Koubâa, RPL In a nutshell: A survey, Comput. Netw. 56 (14) (2012) 3163–3178.
[4] G. Violettas, S. Petridou, L. Mamatas, Evolutionary software defined networking-inspired routing control strategies for the internet of things, IEEE Access 7 (2019) 132173–132192.
[5] G. Violettas, S. Petridou, L. Mamatas, Routing under heterogeneity & mobility for the Internet of Things: a centralized control approach, in: IEEE Global Commun. Conf. (GLOBECOM), 2018, pp. 1–7.
[6] A. Mayzaud, R. Badonnel, I. Chrisment, A taxonomy of attacks in RPL-based internet of things, Int. J. Netw. Secur. (2016).
[7] A. Verma, V. Ranga, Security of RPL based 6LoWPAN networks in the internet of things: A review, IEEE Sens. J. 20 (11) (2020) 5666–5690.
[8] P. Kamgueu, E. Nataf, T. Ndie, Survey on RPL enhancements: a focus on topology, security and mobility, Comput. Commun. 120 (2018) 10–21.
[9] J. Granjal, E. Monteiro, J. Silva, Security for the internet of things: a survey of existing protocols and open research issues, IEEE Commun. Surv. Tutor. 17 (3) (2015) 1294–1312.
[10] M. Landsmann, M. Wahlisch, T. Schmidt, Topology authentication in RPL, in: 2013 IEEE Conf. on Comput. Comm. Workshop (INFOCOM WKSHPS), 0000, pp. 73–74.
[11] A. Arena, et al., Evaluating and improving the scalability of RPL security in the internet of things, Comput. Commun. (2020).
[12] A. Raoof, A. Matrawy, C.-H. Lung, Routing attacks and mitigation methods for RPL-based internet of things, IEEE Commun. Surv. Tutor. 21 (2) (2018) 1582–1606.
[13] P. Perazzo, et al., An implementation and evaluation of the security features of RPL, in: Int. Conf. on Ad-Hoc Netw. and Wireless, Springer, 2017, pp. 63–76.
[14] G. Simoglou, et al., Intrusion detection systems for RPL security: A comparative analysis, Comput. Secur. (ISSN: 0167-4048) 104 (2021) 102219.
[15] P. Pongle, G. Chavan, A survey: Attacks on RPL & 6LoWPAN in IoT, in: 2015 IEEE Int. Conf. on Pervasive Computing (ICPC), 0000, pp. 1–6.
[16] O. Gnawali, P. Levis, The minimum rank with hysteresis objective function, RFC 6719 (2012).
[17] O. Gaddour, et al., OF-FL: QoS-aware fuzzy logic objective function for the RPL routing protocol, in: 2014 IEEE 12th Int. Symp. on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Netw. (WiOpt), 0000, pp. 365–372.
[18] T. Clausen, U. Herberg, M. Philipp, A critical evaluation of the IPv6 routing protocol for low power and lossy networks (RPL), in: 2011 IEEE 7th Int. Conf. on Wireless and Mobile Computing, Networking and Commun. (WiMob), 0000, pp. 365–372.
[19] J. Tripathi, J.C. de Oliveira, J.P. Vasseur, A performance evaluation study of RPL: Routing protocol for low power & lossy networks, in: 2010 44th Annual Conf. on Inf. Sciences and Syst. (CISS), 0000, pp. 1–6.
[20] P. Pongle, G. Chavan, Real time intrusion and wormhole attack detection in internet of things, Int. J. Comput. Appl. 121 (9) (2015).
[21] D. Airehrour, S. Ray, Secure routing for internet of things: A survey, J. Netw. Comput. Appl. 66 (2016) 198–213.
[22] K. Chugh, L. Aboubaker, J. Loo, Case study of a black hole attack on LoWPAN-RPL, in: Proc. of the Sixth Int. Conf. on Emerging Secur. Inf., Syst. and Technol. (SECURWARE), 2012, pp. 157–162.
[23] L. Wallgren, S. Raza, T. Voigt, Routing attacks and countermeasures in the RPL-based internet of things, Int. J. Distrib. Sens. Netw. 9 (8) (2013) 794326.
[24] A. Le, et al., The impacts of internal threats towards routing protocol for low power and lossy network performance, in: 2013 IEEE Symp. on Comput. and Commun. (ISCC), 0000, pp. 000789–000794.
[25] P. Perazzo, et al., DIO Suppression attack against routing in the internet of things, IEEE Commun. Lett. 21 (11) (2017) 2524–2527.
[26] T. Umer, et al., Information and resource management systems for internet of things: Energy management, communication protocols & future applications, Future Gener. Comput. Syst. 92 (2019) 1021–1027.
[27] J.R. Douceur, The sybil attack, in: Int. Workshop on Peer-to-Peer Systems, Springer, 2002, pp. 251–260.
[28] A. Le, et al., The impact of rank attack on network topology of routing protocol for low-power and lossy networks, IEEE Sens. J. 13 (10) (2013) 3685–3692.
[29] W. Xie, et al., Routing loops in DAG-based low power and lossy networks, in: 24th IEEE Int. Conf. on Adv. Inf. Networking and Appl., 2010, pp. 888–895.
[30] A. Kamble, V. Malemath, D. Patil, Security attacks and secure routing protocols in RPL-based internet of things: Survey, in: Int. Conf. on Emerging Trends Innovation in ICT (ICEI), 2017, pp. 33–39.
[31] D. Airehrour, J.A. Gutierrez, S.K. Ray, SecTrust-RPL: A secure trust-aware RPL routing protocol for internet of things, Future Gener. Comput. Syst. 93 (2019) 860–876.
[32] A. Sehgal, et al., Addressing DODAG inconsistency attacks in RPL networks, in: 2014 IEEE Global Inf. Infrastructure and Netw. Symp. (GIIS), 0000, pp. 1–8.
[33] A. Aris, S.F. Oktug, S. Berna Ors Yalcin, RPL version number attacks: In-depth study, in: NOMS 2016 - 2016 IEEE/IFIP Network Operations and Manage. Symp., 0000, pp. 776–779.
[34] A. Mayzaud, et al., A study of RPL DODAG version attacks, in: IFIP Int. Conf. on Auton. Infrastructure, Manage. and Secur., Springer, 2014, pp. 92–104.
[35] A. Le, et al., A specification-based IDS for detecting attacks on RPL-based network topology, Information 7 (2) (2016) 25.
[36] T. Theodorou, et al., A multi-protocol software-defined networking solution for the internet of things, IEEE Commun. Mag. 57 (10) (2019) 42–48.
[37] G. Violettas, et al., An experimentation facility enabling flexible network control for the Internet of Things, in: IEEE 2019 Conf. on Comput. Commun. Workshops, 0000, pp. 992–993.
[38] A. Dunkels, B. Gronvall, T. Voigt, Contiki - a lightweight and flexible operating system for tiny networked sensors, in: 29th Annual IEEE Int. Conf. on Local Comput. Netw., 2004, pp. 455–462.
[39] I.H. Witten, et al. (Eds.), The WEKA workbench, in: Data Mining, Morgan Kaufmann, 2017, pp. 553–571.
[40] Graphstream, 2018, https://github.com/graphstream.
[41] S. Schaller, D. Hood, Software defined networking architecture standardization, Comput. Stand. Interfaces 54 (2017) 197–202.
[42] A. Dutot, et al., GraphStream: A tool for bridging the gap between complex systems and dynamic graphs, in: EPNACS'2007, 0000, pp. 63.
[43] S. Raza, L. Wallgren, T. Voigt, SVELTE: Real-time intrusion detection in the internet of things, Ad Hoc Netw. 11 (8) (2013) 2661–2674.
[44] M.N. Napiah, et al., Compression header analyzer intrusion detection system (CHA - IDS) for 6LoWPAN communication protocol, IEEE Access 6 (2018) 16623–16638.
[45] S. Kalamkar, A. Banerjee, A. Roychowdhury, Malicious user suppression for cooperative spectrum sensing in cognitive radio networks using Dixon's outlier detection method, in: 2012 National Conf. on Commun. (NCC), IEEE, 0000, pp. 1–5.

[4] https://github.com/boygioykaskyriakos/outliers_platform.

[46] C. Efstathiou, Estimation of type i error probability from experimental dixon's "Q" parameter on testing for outliers within small size data sets, Talanta 69 (5) (2006) 1068–1071.
[47] B. Amidan, T. Ferryman, S. Cooley, Data outlier detection using the Chebyshev theorem, in: 2005 IEEE Aerospace Conf., 0000, pp. 3814–3819.
[48] D. Fogel, An introduction to simulated evolutionary optimization, IEEE Trans. Neural Netw. 5 (1) (1994) 3–14.
[49] A. Likas, N. Vlassis, J. Verbeek, The global k-means clustering algorithm, Pattern Recognit. 36 (2) (2003) 451–461.
[50] M. Sharir, A strong-connectivity algorithm and its applications in data flow analysis, Comput. Math. with Appl. 7 (1) (1981) 67–72.
[51] T. Cormen, et al., Introduction to Algorithms, The MIT Press, 2009.
[52] A. Marback, et al., A threat model-based approach to security testing, Softw. Pract. Exper. J. 43 (2) (2013) 241–258.
[53] R. Gupta, et al., Machine learning models for secure data analytics: A taxonomy and threat model, Comput. Commun. 153 (2020).
[54] S. Marano, V. Matta, L. Tong, Distributed detection in the presence of Byzantine attacks, IEEE Trans. Signal Process. 57 (1) (2008) 16–29.
[55] P. Nandhini, B. Mehtre, Directed acyclic graph inherited attacks and mitigation methods in RPL: a review, in: Int. Conf. on Sustain. Commun. Netw. and Appl., Springer, 2019, pp. 242–252.
[56] B. Zarpelão, et al., A survey of intrusion detection in Internet of Things, J. Netw. Comput. Appl. 84 (2017) 25–37.
[57] H. Bostani, M. Sheikhan, Hybrid of anomaly-based and specification-based IDS for internet of things using unsupervised OPF based on mapreduce approach, Comput. Commun. (2016) 52–71.
[58] H. Sedjelmaci, S. Senouci, T. Taleb, An accurate security game for low-resource IoT devices, IEEE Trans. Veh. Technol. 66 (10) (2017) 9381–9393.
[59] J. Kaur, An ultimate approach of mitigating attacks in RPL based low power lossy networks, in: Proc. of 17th Int. Conf. on Secur. and Manage. (SAM), 2019.
[60] T. Wauters, et al., Federation of internet experimentation facilities: architecture and implementation, in: European Conf. on Netw. and Commun. (EuCNC) 2014, IEEE, 0000, pp. 1–5.
[61] M. Berman, et al., GENI: A federated testbed for innovative network experiments, Comput. Netw. 61 (2014) 5–23.
[62] P. Valsamas, et al., Multi-PoP network slice deployment: A feasibility study, in: 2019 IEEE 8th Int, Conf. on Cloud Netw. (CloudNet), 0000, pp. 1–6.
[63] P.D. Maciel, et al., A marketplace-based approach to cloud network slice composition across multiple domains, in: 2019 IEEE Conf. on Netw. Softw. (NetSoft), 0000, pp. 480–488.
[64] P. Valsamas, et al., A multi-domain experimentation environment for 5G media verticals, in: IEEE 2019 Conf. on Comput. Commun. Workshops, 0000, pp. 461–466.
[65] F.R. Group, The university of utah, 2020, https://www.flux.utah.edu/index.
[66] S. Skaperas, L. Mamatas, A. Chorti, Real-time video content popularity detection based on mean change point analysis, IEEE Access 7 (2019) 142246–142260.
[67] S. Skaperas, L. Mamatas, A. Chorti, Real-time algorithms for the detection of changes in the variance of video content popularity, IEEE Access 8 (2020) 30445–30457.
[68] T. Theodorou, L. Mamatas, A versatile out-of-band software-defined networking solution for the internet of things, IEEE Access 8 (2020) 103710–103733.
[69] T. Theodorou, L. Mamatas, SD-MIoT: A software-defined networking solution for mobile internet of things, IEEE Internet Things J. (2020) 1.

George Violettas earned his Ph.D. in **Network Control and Security for the Internet of Things** from the University of Macedonia, Thessaloniki, Greece. He holds an M.Sc. Degree in Applied Informatics from the same University, and a 4-yrs Bachelor in Computer Science from the Hellenic Open University. He has worked as a senior researcher in EU founded projects (Horizon 2020): **NECOS H2020** (Novel Enablers for Cloud Slicing), **UNIC** (Unikernel-based CDNs for 5G Networks, FED4FIRE+ Open Call 4, H2020), **MEC** (Multi-homing with Ephemeral Clouds on the Move in MONROE Open Call 2, H2020) and **CORAL** (Cross-Layer Control of Data Flows, WiSHFUL Open Call 2, H2020). He has hands-on experience with experimentation facilities and test-beds (Fed4fire, Emulab, Monroe).

George Simoglou received his B.Sc. degree in Applied Informatics from the University of Macedonia, Thessaloniki, Greece. His B.Sc. thesis was on the **Security issues of the RPL routing protocol**, presented on Feb. 2020. He is currently working as a Web and software developer. His research interests include **Internet of Things, network protocols and security**.

Sophia Petridou is an **Assistant Professor** in the Department of Applied Informatics, University of Macedonia. She received her Ph.D. from the Department of Informatics, Aristotle University of Thessaloniki, Greece in 2008. Her main research interests are in the areas of ***Internet of Things, Wireless and Optical networks' protocols, formal verifications and probabilistic model checking of protocols, protocols' security***. She has been involved in international **research projects** of: **NECOS H2020** (Novel Enablers for Cloud Slicing) , **UNIC** (Unikernel-based CDNs for 5G Networks, FED4FIRE+ Open Call 4, H2020), **MEC** (Multi-homing with Ephemeral Clouds on the Move, MONROE Open Call 2, H2020) and **CORAL** (Cross-Layer Control of Data Flows, WiSHFUL Open Call 2, H2020). She has more than 40 publications in journals and conferences. She is a **Member of the IEEE Computer Society** and serves as an **Associate Editor** of the *International Journal of Communication Systems*.

Lefteris Mamatas is an **Associate Professor** in the Department of Applied Informatics, University of Macedonia, Greece. He leads the **Softwarized & Wireless Networks Research Group** (http://swn.uom.gr) in the same University. He worked as a researcher at the University College London (UK), Space Internetworking Center/Democritus University of Thrace (Greece), and DoCoMo Eurolabs (Germany). His research interests lie in the areas of **Software-Defined Networks, Internet of Things, 5G Networks, and Multi-Access Edge Computing**. He participated in many international **research projects**, such as NECOS (H2020), FED4FIRE+ OC4 (H2020), WiSHFUL OC2 (H2020), MONROE OC2 (H2020), Dolfin (FP7), UniverSELF (FP7), and Extending Internet into Space (ESA). He has published more than **60 papers** in international journals and conferences. He served as a General Chair for the WWIC 2016 conference and the INFOCOM SWFAN 2016 workshop, as a TPC Chair for the INFOCOM SWFAN 2017, E-DTN 2009, IFIP WWIC 2012 conferences/workshops and as a Guest Editor for the Elsevier Ad Hoc Networks Journal.