Vulnerabilities of the 6P Protocol for the Industrial Internet of Things: Impact Analysis and Mitigation[⋆]

Francesca Righetti[a,*], Carlo Vallati[a], Marco Tiloca[b], Giuseppe Anastasi[a]

[a]*Department of Information Engineering, University of Pisa, Via Girolamo Caruso 16, 56126, Pisa, Italy*
[b]*RISE Cybersecurity, RISE Research Institutes of Sweden, Kista, Sweden*

Abstract

The 6TiSCH architecture defined by the IETF provides a standard solution for extending the *Internet of Things (IoT)* paradigm to industrial applications with stringent reliability and timeliness requirements. In this context, communication security is another crucial requirement, which is currently less investigated in the literature. In this article, we present a deep assessment of the security vulnerabilities of 6P, the protocol used for resource negotiation at the core of the 6TiSCH architecture. Specifically, we highlight two possible attacks against 6P, namely the *Traffic Dispersion* and the *Overloading* attacks. These two attacks effectively and stealthy alter the communication schedule of victim nodes and severely thwart network basic functionalities and efficiency, by specifically impacting network availability and energy consumption of victim nodes. To assess the impact of the attacks two analytical models have been defined, while, to demonstrate their feasibility, they have been implemented in Contiki-NG. The implementation has been used to quantitatively evaluate the impact of the two attacks by both simulations and measurements in a real testbed. Our results show that the impact of both attacks may be very significant. The impact, however, strongly depends on the position of the victim node(s) in the network and it is highly influenced by the dynamics of the routing protocol. We have investigated mitigation strategies to alleviate this impact and proposed an extended version of the Minimal Scheduling Function (MSF), i.e., the reference scheduling algorithm for 6TiSCH. This allows network nodes to early detect anomalies in their schedules possibly due to an Overloading attack, and thus curb the attack impact by appropriately revising their schedule.

Keywords: Industrial Internet of Things, Security, 6TiSCH, 6P, MSF, 6P Vulnerabilities, Availability

[⋆]A preliminary version of this article has been published as G. Carignani, F. Righetti, C. Vallati, M. Tiloca and G. Anastasi, "Evaluation of Feasibility and Impact of Attacks Against the 6top Protocol in 6TiSCH Networks," 2020 IEEE 21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM), 2020, pp. 68-77, doi: 10.1109/WoWMoM49955.2020.00027.

[*]Corresponding author

Email addresses: `francesca.righetti@ing.unipi.it` (Francesca Righetti), `carlo.vallati@unipi.it` (Carlo Vallati), `marco.tiloca@ri.se` (Marco Tiloca), `giuseppe.anastasi@unipi.it` (Giuseppe Anastasi)

1. Introduction

Wireless technologies are currently re-shaping the way industrial systems are designed and managed, as they allow to interconnect cyber-physical systems while guaranteeing rapid deployment and flexibility. In the industrial field, reliable, secure and timed communication is essential in many application domains to enable functionalities such as remote monitoring and control. Also, since industrial applications increasingly rely on, and interact with, cloud-based solutions, they would greatly benefit from a full-fledged, standardized integration with the Internet.

To facilitate this process, the Internet Engineering Task Force (IETF) has defined the 6TiSCH architecture [1] to enable the *Industrial Internet of Things (IIoT)*, i.e., the extension of the traditional IoT paradigm to industrial environments.

The 6TiSCH architecture provides the abstraction of IPv6 link over TSCH, with an industrial grade of service. To this end, it includes a *Scheduling Function (SF)* and the *6top negotiation protocol (6P)* for managing communication resources. The SF is used, at each node, to dynamically derive the amount of communication resources required by the application, depending on the current operating conditions. Instead, the 6P protocol is used to negotiate the allocation of these communication resources with neighbor nodes. The 6TiSCH specifications also include a *Minimal Scheduling Function (MSF)* [2].

Since 6P has a key role in providing the Quality of Service (QoS) required by the application, its correct behavior is of vital importance, especially in industrial contexts.

However, reliable and timely communication alone is not sufficient in industrial environments. Communication security is a non functional requirement of paramount importance in industrial systems, as it ensures to fulfill the security objectives of confidentiality, integrity and freshness of messages exchanged within the network. To this aim, 6TiSCH includes a set of basic security features, i.e., the *Minimal Security Framework* [3]. This defines the procedures through which a node securely joins the network, contextually obtaining a network wide key to perform secure communication by protecting exchanged data and control messages, including 6P messages.

Furthermore, even when relying on secure communication through message protection, ensuring reliable and timely communication strictly relates to fulfilling the security objective *availability*, which the National Institute of Standards and Technology (NIST) defines as "ensuring timely and reliable access to and use of information" [4]. Consistent with a loss of availability being "the disruption of access to or use of information or an information system", NIST classifies availability to be of "HIGH" potential impact when sensor data are involved (e.g., in industrial systems). That is, a loss of availability in industrial systems would have a "severe or catastrophic adverse effect on organizational operations, organizational assets, or individuals".

In this article, we show that, even when secure communication is used, the 6P protocol exhibits a number of security vulnerabilities, which may compromise its correct behavior and severely impact the network performance as well network reliability and availability. In particular, we describe two different attacks that can be performed by a compromised node during a 6P transaction, namely the *Traffic Dispersion* attack and the *Overloading* attack. We show that these attacks can significantly impair the network behavior, by thwarting network basic functionalities and efficiency, and specifically impacting network availability and energy consumption of victim nodes.

The rationale of both attacks is to alter the communication schedule adopted by the victim node, through bogus 6P messages. In particular, both attacks result in a communication schedule that differs between the victim node and its neighbors. Therefore, both attacks result in an undesired, distorted use of network resources on the victim node, which in turn does not reflect and accommodate the application and traffic requirements according to expectations. That is, both attacks undermine the timely and reliable access to information otherwise exchanged within the network in the attack-free case, hence greatly jeopardizing availability.

As ultimate goal, the Traffic Dispersion attack aims at disrupting the message transmissions of the victim node. Hence, it especially results in jeopardizing the propagation of information to the root node, which acts as the data sink node for final information processing and release. This in turn yields the disruption of access to and use of information collected in the network.

Instead, the Overloading attack aims at allocating unneeded communication resources at the victim node, thus increasing its energy consumption. Hence, it especially results in the induced additional allocation of unneeded resources on the victim nodes, thus preventing a more appropriate allocation and use of such resources. In turn, this yields additional and unnecessary energy consumption, which severely reduces the operational capability and lifetime of the victim nodes.

Even when the Minimal Security framework is used, the two attacks are still possible in practical settings. This is because practical settings do rely on a single network key, shared by all the nodes, either pre-configured or provided upon joining the network. It follows that any (compromised) node is able to inject or alter valid 6P messages exchanged by other nodes. That is, although intended to concern only a particular pair of nodes, a 6P negotiation becomes practically exposed to passive monitoring and active manipulation from any other (compromised) node.

In order to analyze the effects of both attacks we first develop two analytical models. Then, to extend our study, we carry out an extensive analysis, through simulation and measurements on a real testbed. Our results show that the effects of the considered attacks may be very relevant and depend on the position of the victim nodes within the network. Specifically, we observed a reduction in the overall Packet Delivery Ratio of up to 65%, due to a Traffic Dispersion attack mounted against three victim nodes. Also, the Overloading attack can increase the energy consumption of a victim node of up to 55%.

In order to alleviate the impact of the considered attacks, we have investigated possible mitigation strategies. In particular, we propose an extension of the standard MSF (referred to as MSF-M), by introducing a monitoring mechanism running on the nodes, that allows an early detection of Overloading attacks. Also, we evaluate through simulation the benefits of using MSF-M, instead of the original SF.

To the best of authors' knowledge, this is the first paper to consider potential attacks to 6P, in addition to our original conference contribution [5], which is extended in this work. In particular in this work we extensively analyze the impact of the attacks analytically and via an extensive performance evaluation that includes both simulations and real experiments. Our experiments are the first to highlight how those attacks can be used to impair the normal resource allocation procedures, which might lead to reduced data delivery or an increase in the energy consumption of nodes. As highlighted in Section 3, the majority of the works from the literature that investigated security aspects of 6TiSCH mainly focused on other aspects such as time synchronization, routing and jamming attacks. The only previous work focusing on 6P attacks is our previous

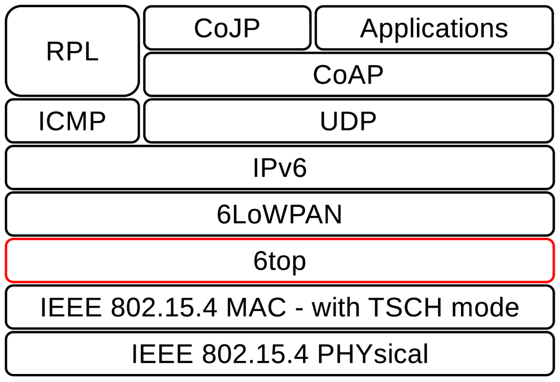


Figure 1: Overview of the typical 6TiSCH protocol stack.

conference paper [5], which has been considerably extended in this paper.

The remainder of this article is structured as follows. Section 2 provides an overview of the technical background, while in Section 3 we discuss the related work. In Section 4 we introduce the considered threat model. In Section 5 we present the general attack rationale and scheme, while in Section 6 and Section 7 we describe the specific execution of the Traffic Dispersion attack and the Overloading attack, respectively. In Section 8 we introduce the analytical models to analyze the effects of both attacks. In Section 9 we present our simulation results, while in Section 10 we describe the testbed used for our experimental analysis. In Section 11 we discuss attack mitigation mechanisms and propose a modified version of the standard MSF. Finally, in Section 12 we draw our conclusions.

2. The 6TiSCH Architecture

In this section, we describe the 6TiSCH architecture [1] defined by the Internet Engineering Task Force (IETF) to enable the IIoT paradigm.

Figure 1 shows the complete protocol stack. The Physical and Data Link layers correspond to the IEEE 802.15.4 standard, a wireless technology for low-rate and short-range wireless communication. Specifically, the *Time Slotted Channel Hopping (TSCH)* mode of 802.15.4 is used as Medium Access Control (MAC) protocol to provide high reliability, guaranteed bandwidth and predictable latency, while ensuring low energy consumption. On top of the Data Link layer, the *6top Operation* sublayer [6] implements the abstraction of IP link over TSCH, allowing the transmission of IPv6 datagrams with industrial grade service, while the 6LoWPAN protocol [7] provides integration and header compression of IPv6 datagrams over TSCH frames. Multi-hop delivery of IPv6 datagrams is accomplished through the *Routing Protocol for Low-Power and Lossy Networks (RPL)* [8]. At the application layer, the *Constrained Application Protocol (CoAP)* [9] uses the underlying transport service provided by the UDP to enable lightweight, RESTful interactions among IoT nodes. CoAP is used not only to exchange messages generated by the application, but also to transport messages related to the *Constrained Join Protocol (CoJP)* [3]. The latter protocol is part of the 6TiSCH *Minimal Security* framework and is intended for ensuring secure admission of IoT nodes.

In the following subsections, we describe some of the above-mentioned protocols. Specifically, we focus on those protocols that are later considered in the analysis of the 6P vulnerabilities.

2.1. *Time Slotted Channel Hopping*

Time Slotted Channel Hopping (TSCH) is one of the access modes defined in the IEEE 802.15.4 standard [10]. It allows for short-range wireless communication with guaranteed bandwidth, bounded latency, high reliability and energy efficiency. To this end, it leverages *time-slotted* access, *multi-channel communication*, and *frequency hopping*.

In TSCH, time is divided into *timeslots* of fixed length, which are in turn grouped into equally sized and periodic *slotframes*. Timeslots are long enough to enable the transmission of a maximum-size data frame (i.e., 127 bytes) and the corresponding acknowledgment. Timeslots are identified through the Absolute Slotframe Number (ASN), an integer that counts the number of timeslots elapsed since a reference point (e.g., the start of the network).

To increase the network capacity, TSCH allows different nodes to concurrently transmit/ receive on the same timeslot, using different channels (multi-channel communication). There are 16 different channels available, each identified by a channel offset, i.e., an integer value in the range 0-15. Furthermore, TSCH leverages channel hopping to provide resilience to interference and multi-path fading. Each node changes the transmission frequency used at each timeslot, according to a predefined hopping sequence. Specifically, at timeslot T, the frequency F is computed as a function of the ASN and the channel offset.

In TSCH, each element in the two-dimensional slotframe, namely a *cell*, is identified through a timeslot and a channel offset. A cell can be either *dedicated* or *shared*. Dedicated cells are assigned to a couple of nodes for data transmission/reception and are guaranteed to be contention-free. Instead, shared cells are allocated to all (or many) nodes and are accessed on a contention basis.

2.2. *6TiSCH Scheduling*

While TSCH allows nodes to allocate and deallocate cells for communication, depending on their traffic needs, the standard does not specify how nodes can allocate them. This is under the responsibility of the 6top sublayer, which implements the abstraction of an IP link over TSCH (see Figure 1) and represents the core of the 6TiSCH architecture.

The 6top sublayer includes a *Scheduling Function (SF)* and the *6top negotiation protocol (6P)* [6]. The SF is used to compute the number of cells required by a node to meet the application requirements, while the 6P protocol is used to negotiate the allocation and deallocation of cells with neighbor nodes.

While 6TiSCH allows for using different SFs to meet the requirements of specific IoT applications, the 6TiSCH specifications include a reference SF, namely the *Minimal Scheduling Function (MSF)* [2]. MSF allows nodes to adapt to time-varying traffic conditions by dynamically allocating and deallocating cells through the 6P negotiation protocol, as described in Algorithm 1. Each node periodically (i.e., every *MAX_NUMCELLS*) checks the utilization of the cells it has allocated in transmission and increases/decreases their number in such a way to keep their utilization within two pre-defined thresholds, namely *LIM_NUMCELLSUSED_HIGH* and *LIM_NUMCELLSUSED_LOW*. Specifically, if the number of transmission cells used is higher than *LIM_NUMCELLSUSED_HIGH*, one more cell is negotiated. Instead, if the cells used are below *LIM_NUMCELLSUSED_LOW*, one cell is deallocated.

The MSF relies on 6P to negotiate the dynamic allocation and deallocation of cells among couples of neighboring nodes. Each node can execute only one 6P transaction at a time, and

Algorithm 1: MSF Algorithm

Input: NCE = Number of allocated cells elapsed from the last time MSF was run
$MAX\_NUMCELLS$ = Number of allocated cells elapsed to trigger next MSF run
NCU_{tx} = Number of allocated cells used for transmission
$LIM\_NUMCELLSUSED\_HIGH$ = Threshold to add a transmission cell
$LIM\_NUMCELLSUSED\_LOW$ = Threshold to delete a transmission cell

Output: ADD/DEL one transmission cell

```
1 while allocated cell then{
2     if $NCE >$ MAX_NUMCELLS then{
3         if $NCU_{tx} >$ LIM_NUMCELLSUSED_HIGH then{
4             trigger 6P to ADD one cell}
5         if $NCU_{tx} <$ LIM_NUMCELLSUSED_LOW then{
6             trigger 6P to DEL one cell}
7         $NCE = 0$
8     }else{$NCE = NCE + 1$}
9 }
```

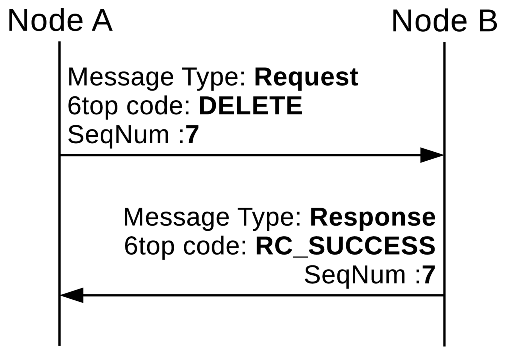


Figure 2: Example of successful 6P transaction (2-step DELETE).

6P messages are embedded in the Information Elements (IE) [10] of TSCH data frames, whose protection relies on the security services of the IEEE 802.15.4 standard (see Section 2.4).

A 6P transaction among two nodes follows a Request/Response paradigm and may consist of 2 or 3 steps. In a 2-step transaction, the cells to be added/removed/relocated are selected by the initiator node and notified to the destination node through a Request message. The destination node replies with a Response message. An example of a 2-step ADD transaction is shown in Figure 2. In a 3-step transaction, the cells to be added/removed/relocated are selected by the destination node and a final Confirmation message is sent by the initiator node to conclude the transaction. In both cases, the Request message includes a code specifying the command to be executed by the destination node, namely ADD, DELETE, RELOCATE, COUNT, LIST, SIGNAL and CLEAR. Both the Response and Confirmation messages also include a return code to notify a successful (RC_SUCCESS) or failed transaction.

In order to prevent inconsistencies and a misalignment that would also yield inefficient resource utilization, it is important that any two nodes keep their pairwise schedule updated and consistent over time. To ensure this, each pair of neighbor nodes maintains a sequence number (SeqNum) and uses it to check that Request, Response and Confirmation messages are consistent and correctly match with one another, during and throughout 6P transactions. Specifically, SeqNum is used to (i) detect and handle duplicate 6P messages; (ii) detect possible out-of-order negotiations; and (iii) detect possible inconsistencies in the schedules.

2.3. The RPL Protocol

The *IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL)* [8] ensures multi-hop delivery of data messages. RPL takes a distance-vector approach and assumes that the majority of traffic is upward (i.e., generated by nodes and directed to a central node). Downward traffic (i.e., from the central node to the other nodes) is assumed to be sporadic, while node-to-node interactions are considered rare. For this reasons, RPL builds and maintains a *Destination Oriented Directed Acyclic Graph (DODAG)* rooted on the central node.

Within the DODAG, each node derives its *parent set*, i.e., the set of candidate neighbors for upstream data delivery, and selects the *preferred parent* as the most convenient neighbor in the parent set. To this end, each node computes a *rank* for each of its neighbors, i.e., a value that summarizes the routing-relevant metrics associated with that neighbor (e.g., the number of hops to the root node, the link quality, the transmission latency, the current battery level). In particular, the *Objective Function (OF)* defines the specific rules to compute a rank value from the available metrics indicators, as well as the policy for the selection of the preferred parent, based on the computed ranks. A commonly used OF, the Minimum Rank with Hysteresis Objective Function [11], focuses on minimizing the expected number of transmissions (ETX) on the path from the packet source to the DODAG root [12].

2.4. The 6TiSCH Minimal Security Framework

The 6TiSCH *Minimal Security* framework [3] allows nodes to securely join a 6TiSCH network through the Constrained Join Protocol (CoJP). This process involves the following three entities: (i) the *pledge*, i.e., the node that wishes to join the network; (ii) the *Join Registrar/ Coordinator (JRC)*; and (iii) the *Join Proxy (JP)*, which acts as an intermediary between the pledge and the JRC. CoJP assumes that the pledge and the JRC own a cryptographic Pre-Shared Key (PSK), which they use to mutually authenticate.

While performing the joining process, CoJP messages exchanged between the pledge and the JRC (possibly through the JP) are not protected at the link layer. In fact, at that point in time, the pledge does not own yet the key material used to protect messages. To ensure the security of the joining process, the pledge and the JRC rather use their PSK as Master Secret, in order to derive a dedicated security context. Then, they rely on such security context to protect the exchanged CoJP messages end-to-end at the application level, by using the standard security protocol OSCORE [13][14]. At the end of the joining process, the pledge receives from the JRC the key material to use for securely communicating with other nodes in the network.

When considering the actual traffic within the 6TiSCH network, TSCH frames are protected by means of the link-layer security services of IEEE 802.15.4. These services provide integrity protection and authentication of transmitted messages, which may be additionally encrypted to achieve data confidentiality. In the latter case, frames are encrypted by using the AES symmetric cipher in CCM* (Counter with CBC-MAC) mode. The specific encryption key to use is provided to a node by the JRC, upon joining the network as a pledge. On the other hand, the cipher nonce used to encrypt a specific message is computed from the node identifier of the sender node and the ASN at the current timeslot. Finally, the encrypted message and the Message Integrity Code (MIC) are produced by providing as input to AES the cipher nonce, the encryption key, the payload and the information elements to protect.

Messages exchanged within the network are usually both integrity protected and encrypted, with the exception of some specific control frames. These include Enhanced Beacons (EBs) [10], i.e., broadcast control messages implementing several functionalities, which are integrity-protected but not encrypted. This is intentional and allows also joining nodes to read the message content, and thus to learn about the network configuration and to synchronize themselves with the current slotframe.

In practical settings, the key material received from the JRC consists of two symmetric network keys, which are commonly shared by all the nodes in the network. That is, the first key is used only to integrity-protect and verify the EBs. Instead, the second key is used only to perform encryption and integrity-protection, as well as to verify, the received data and acknowledgments in the network.

In the light of the key material typically used as described above, authentication of exchanged data frames is actually ensured only at a broad network level. In fact, a message recipient is able to determine whether the message sender is any of the other nodes in the same network. However, it is not possible to ensure strict source authentication of data messages sent in the network. That is, a message recipient is not able to assert whether the message sender is actually the exact network node it claims to be, e.g., based on the message source address. Evidently, this makes it possible to perform a number of attacks relying on passive message listening, active message forgery and injection, as well as on identity impersonation (e.g., by spoofing addressing information). Therefore, as further discussed in Sections 5-7, 6P transactions are inherently vulnerable to security attacks that exploit the commonly shared network key and their usage to protect data frames.

3. Related Work

Although many works have considered different aspects of the 6TiSCH architecture, such as resource allocation [15, 16], nodes' selfishness [17], network formation dynamics [18, 19] and routing protocol performance [20], only few have focused on the analysis of its security features to highlight potential vulnerabilities. Specifically, those have addressed four main security aspects: (i) secure routing; (ii) secure time synchronization; (iii) secure scheduling with respect to selective jamming attacks; and (iv) secure use of 6P.

Attacks against routing protocols in IoT networks can easily thwart the operations and safety of the whole network. The impact of internal routing attacks was studied in [21] and [22]. In [21] the impact of the attack exploiting the DODAG Information Solicitation (DIS) message during network formation is evaluated. The DIS attack targets the service availability, and it causes a noticeably increase of the network joining time and of the energy consumption for the involved nodes. Moreover, in [22] it has been shown that the RPL protocol is vulnerable to attacks aimed at forging rank values, which represent nodes' position with respect to the root node in the DODAG built by RPL. In order to protect RPL from rank-related attacks, a forged rank and routing metric detector (FORCE) is proposed. In particular, every node analyzes the RPL messages received from its neighbors, in order to detect possible rank-attacks and react against those.

Another crucial aspect in IoT networks based on the 6TiSCH architecture is time synchronization. That is, nodes must be precisely synchronized in turning their radio interface on for transmitting and receiving packets during their assigned TSCH timeslots. Indeed, attacks against time synchronization in TSCH are possible, and the work in [23] describes two types of attacks. The first one is based on tampering with control messages, in order to induce victim nodes to synchronize with a malicious node, thus adopting a different, bogus channel hopping sequence. Instead, the second one exploits the workflow of the RPL protocol. That is, by forging RPL control messages, the attacker is able to manipulate the topology of the network. The goals of the attacker are to lower the communication reliability and to fragment the network. The authors also propose a set of countermeasures for both the attack types.

The channel hopping mechanism allows nodes in 6TiSCH networks to be resilient to interference. However, they can still be vulnerable to selective jamming attacks. For instance, in [24] the authors analyze how an external jammer can efficiently and effectively jeopardize a node's communication pattern, by selectively and stealthy jamming its transmissions and receptions. This is possible by exploiting the periodicity and predictability of the channel hopping sequence, thus getting knowledge of the victim's full schedule. In order to neutralize the attack, the authors proposed DISH, a preventive countermeasure that allows network nodes to pseudo-randomly alter their communication pattern at every slotframe in a way which is unpredictable to the adversary, while still preserving a consistent, collision-free schedule and without requiring any additional communication.

When considering the 6TiSCH architecture, 6P plays the key role of enabling resource negotiation among neighbor nodes, for building and maintaining their TSCH schedule. Thus, a successful execution of attacks against 6P can disrupt communications among network nodes and greatly impair network performance as a whole. To the best of our knowledge, the only contribution that addresses possible security threats to 6P is our previous work in [5]. In particular, [5] evaluated the feasibility of performing two different attacks against 6P, i.e., Traffic Dispersion attack and Overloading attack, as carried out by internal compromised nodes under adversarial control.

This article extends our previous work in [5], by providing an extended performance evaluation over a broader set of network configurations and attack cases. In particular, it considers the presence of multiple compromised nodes, as well as victim nodes positioned in different network areas and with different configuration settings. The impact of the two attacks has been extensively evaluated through simulation experiments and measurements on a real IoT testbed [25]. In this article, we also define possible mitigation strategies for the attacks and propose an extension of the standard SF for 6TiSCH, namely MSF.

4. Threat Model

The rest of this article considers a threat model where the adversary is internal, both active and on-path, and has control of one or multiple nodes in the network. In particular, multiple nodes under the adversary's control can be exploited in order to achieve a greater attack impact against the network and application scenario. In fact, this enables the adversary to overall target multiple sets of victim nodes at the same time, e.g., one victim pair through one of the compromised nodes. As a consequence, this can further result not only in direct effects such as increased packet loss or energy consumption, but also in indirect effects as a byproduct, such as sinkholes and network partitioning yielding a more severe service disruption.

Being the adversary internal, the compromised nodes under her control are active members of the network, and thus know the currently used key material. The latter especially includes the symmetric network key shared by all the nodes in the network, and used to protect data and acknowledgment frames (see Section 2.4), among which 6P messages. As this is indeed the case in practical settings, every node under the adversary's control is thus able to access and possibly modify the content of any intercepted message, as well as to create new messages to inject in the network as cryptographically valid.

In other words, the use of a commonly shared network key makes both the *confidentiality* and *integrity* security objectives [4] possible to ensure only at a whole network level, but not for any different pair of network nodes as to their pairwise communications. In particular, source authentication of messages cannot be achieved, i.e., a recipient can securely determine that a received message has been sent by *another* node in the network, but not strictly by the alleged sender (based, e.g., on the message source address). As explained in the following sections elaborating the general attack execution pattern (see Section 5) and its two specific incarnations Traffic Dispersion attack (see Section 6) and Overloading attack (see Section 7) presented in this article, this can be non-trivially exploited by an internal adversary in control of one or more compromised nodes, in order to perform the two attacks above.

Evidently, this would not be possible if each two neighbor nodes were sharing a dedicated symmetric pairwise key and using it to protect the messages they exchange with one another. This would ensure strict source authentication of messages, thus preventing altogether any undetected message tampering from other nodes in the network. Instead, as previously mentioned, practical settings do rely on a single, commonly shared network key. While this prevents from achieving strict source authentication, it is also a typical choice as it yields an overall easy and feasible key management process. Instead, the distribution, establishment and revocation of link-layer pairwise keys is a non-trivial, often discouraging task, which practically paves the way to simply rely on a shared network key. Further related considerations as well as the definition of suitable and seamless procedures for establishing pairwise keys, are out of the scope of this work.

On the other hand, note that some relevant pieces of information are rather transmitted as plain content in the first place. These especially include the current ASN used to build the cipher nonce (see Section 2.4), as part of Enhanced Beacon frames that are authenticated and integrity-protected, but not encrypted.

Furthermore, we assume that every node under the adversary's control is within the transmission/reception range of the network nodes targeted as its intended victims. Hence, through each of the compromised nodes, the adversary is able to intercept messages sent by the neighbor legitimate nodes, as well as to possibly inject (newly) forged messages. Finally, by spoofing the source MAC address of newly crafted or altered injected messages, the adversary is able to induce a recipient node A to accept them not only as authentic and cryptographically valid, but also as generated by any other impersonated sender node B.

The specific adversary considered in this article is especially interested in leveraging 6P messages, i.e., messages exchanged as part of 6P transactions. This is the starting point as well as

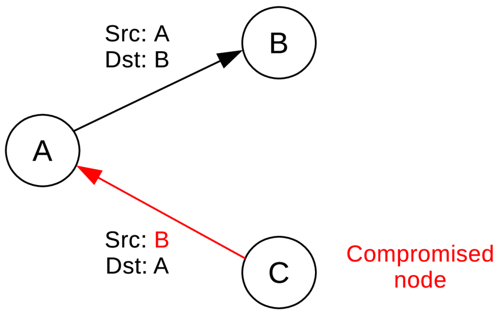


Figure 3: Example of victim pair and adversarial node.

the means to practically perform concrete attacks against two neighbor nodes and especially their 6TiSCH schedule. Specifically, building on the threat model discussed above, the adversary relies on the compromised nodes under her control to intercept, forge and inject valid 6P messages. This is exploited in order to have the victim node pair performing bogus and inconsistent, yet complete 6P transactions. As a result, each of the two nodes in the victim pair ends up enforcing a different schedule with its respective peer, although still believing to indeed share a common schedule, and most important unaware that any manipulative attack has happened in the first place.

To fix ideas, we refer to the example in Figure 3, where A and B are two legitimate neighbor nodes, and communicate as per a consistent, currently shared 6TiSCH schedule. The considered example can well reflect the case where, in the DODAG built by RPL, node A has chosen node B as its own preferred parent. On the other hand, node C is one of the compromised nodes under the adversary's control, and is located within the reception/transmission range of both node A and node B. Note that, depending on the position of the compromised node C in the network topology, it might be possible for the adversary to effectively use such a node to perform attacks against multiple, different pairs of nodes.

5. Attack Rationale and Execution Pattern

This section builds on the threat model in Section 4, and overviews the rationale and general pattern that the adversary can rely on for building up concrete attack instances to mount, such as the attacks presented in Section 6 and Section 7 and specifically targeted in this article.

For simplicity, but with no loss of generality, the following refers to a single compromised node C under the adversary's control, targeting a single pair of nodes A and B. Nevertheless, consistently with the threat model in Section 4, the adversary may be in control of more compromised nodes, each of which may be able to target multiple pairs of nodes within its transmission/reception range.

5.1. Attack Rationale

Let us consider a compromised node C and a pair of neighbor nodes A and B as its target. We assume that nodes A and B share a consistent, previously established 6TiSCH schedule $\mathcal{S}$.

As a preliminary step, the adversary locally produces an alternative 6TiSCH schedule $\mathcal{S}' \neq \mathcal{S}$, which may additionally aim at maximizing the impact of the intended specific attack. For example, the alternative schedule $\mathcal{S}'$ can be built in such a way that it does not specify any of the cells in the original schedule $\mathcal{S}$.

Then, node C impersonates node A and performs a bogus 6P transaction with node B. The main goal is to make B install the new schedule $\mathcal{S}'$, as allegedly proposed by A. It follows that, after the attack is completed, node A has still the original schedule $\mathcal{S}$, and will continue referring to it as the schedule to use for communicating with B. On the other hand, node B has installed the new alternative schedule $\mathcal{S}'$ and erroneously believes it to be a new valid schedule shared with A. That is, from now on, B will refer to $\mathcal{S}'$ as the schedule to use for communicating with A.

5.1.1. Synchronization of SeqNum Values

When performing the bogus 6P transaction described above, the adversary has to specifically take care of a caveat that would otherwise prevent the attack from being successful. That is, other than ensuring that A and B erroneously *believe* to share the same schedule, the adversary must also keep A and B *actually* aligned with respect to their 6P SeqNum (see Section 2.2). This is necessary in order to hide from A and B that an attack has occurred in the first place, which they would be able to understand when later on performing a legitimate 6P transaction and detecting a mismatch of their respective SeqNum values.

The adversary can easily learn the SeqNum value used by A and B before the attack execution starts, by simply eavesdropping a legitimate 6P transaction between A and B, and then reading the SeqNum value conveyed in the exchanged 6P messages. However, since the bogus 6P transaction actually engages only with B (see above), it is neither easy nor obvious to have A and B aligned on a same SeqNum value *after* the attack execution ends, i.e., once B has installed the new alternative schedule $\mathcal{S}'$.

5.2. Attack Execution

According to the attack rationale in Section 5.1, the adversary can successfully perform an attack in two phases, as detailed below and as depicted in Figure 4. The following assumes that nodes A and B have their SeqNum equal to 4 before the adversary starts performing the attack.

Consistently with the threat model presented in Section 4, the compromised node under adversarial control relies on its legitimate knowledge of the commonly shared network key as main attack enabler, exploiting the fact that confidentiality, integrity and source authentication of exchanged messages are ensured only at a whole network level. Thus, when performing the attack steps presented in Sections 5.2.1 and 5.2.2, the compromised node is able to read in clear text the 6top messages exchanged by nodes A and B, as well as to craft and inject cryptographically valid, yet bogus, 6top messages addressed to A (B) as if they were plausibly sent by B (A).

5.2.1. Attack Phase 1

In the first phase of the attack, node C impersonates node B, and starts a 6P LIST transaction with node A, by sending a forged request message addressed to A. After receiving the request from C (impersonating node B), node A performs the following actions.

First, node A sends a link-layer ACK frame to node B. After that, A sends a response message addressed to node B, which includes its current schedule $\mathcal{S}$ shared with B, consistent with the received request message and as per a 6P LIST transaction. Finally, after receiving the link-layer ACK frame sent by B and matching with the previous response message, A increments

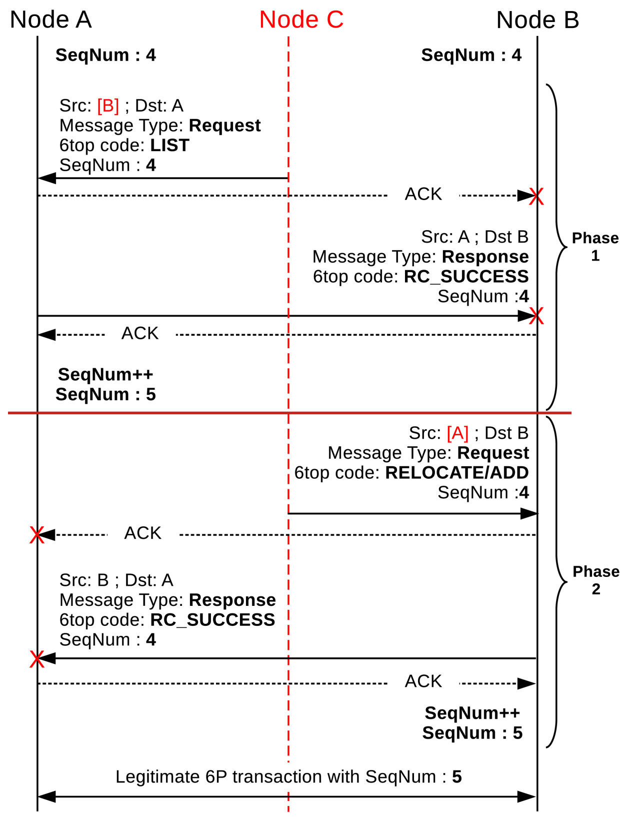


Figure 4: General attack execution.

its SeqNum value associated with B, which thus takes value 5. Note that node B is instead still considering 4 as value of its SeqNum associated with node A.

During this first attack phase, node C intercepts the response message from A and hence gets knowledge of the conveyed original schedule $\mathcal{S}$. On the other hand, node B simply ignores both the ACK frame and the response message sent by A, as they do not match against any ongoing 6P transaction from its own point of view. Also, B sends an ACK as a reply to the ignored response message from A. This does not prevent the attack execution from being successful, and fundamentally results in A incrementing the value of its SeqNum associated with B.

5.2.2. Attack Phase 2

In the second phase of the attack, node C generates an appropriate new schedule $\mathcal{S}' \neq \mathcal{S}$, and performs a bogus 6P transaction with node B. When doing so, C impersonates node A and uses 4 as the value of SeqNum conveyed in the request message to B, i.e., the value learned during the first attack phase (see Section 5.2.1).

The particular messages exchanged with node B, hence the particular 6P bogus transaction to perform, depends on the specific attack instance to mount. The following sections focus on two particular attack instances, i.e.,: i) the *Traffic Dispersion* attack (see Section 6), which can be performed through a 6P DELETE or RELOCATE transaction with node B; and ii) the *Overloading* attack (see Section 7), which can be performed through a 6P ADD transaction with B.

After having received the request message from C (impersonating node A), node B performs the following actions.

First, node B sends a link-layer ACK frame to node A. After that, B sends a response message addressed to A, which includes information consistent with the specific 6P transaction triggered by the received request message. Finally, B increments the value of its SeqNum associated with A, which thus takes value 5 and becomes aligned with the value at A installed during the first phase of the attack (see Section 5.2.1).

Note that node B correctly processes the received request message, since this includes the SeqNum value 4 that B was expecting to use in the next 6P transaction with node A. In addition, node A simply ignores both the ACK frame and the response message sent by B, since they do not match against any ongoing 6P transaction from its own point of view. Also, node A sends an ACK frame as a reply to the ignored response message from B, which does not prevent the successful attack execution.

5.2.3. Discussion

As a result, after the end of the second attack phase, nodes A and B share the same SeqNum value 5. Thus, they are not going to experience inconsistencies during possible 6P transactions that they perform with one another later on. That is, the two nodes seamlessly continue to operate, and would not perceive any anomaly or detect that the attack has occurred in the first place.

Most important, from now on node A (B) refers to the schedule $\mathcal{S}$ ($\mathcal{S}'$) to communicate with node B (A). Since the two schedules $\mathcal{S}$ and $\mathcal{S}'$ do not partially or entirely match by construction, this results in a disruptive impact against communications between the two nodes, and severely affects their performance as well as the reliability of the 6TiSCH network as a whole.

That is, a successful execution of the attack results in an undesired, distorted use of network resources on the victim node, which in turn does not reflect and accommodate the application and traffic requirements according to expectations. With the exact impact depending on its specific incarnation (see Sections 6 and 7), the attack undermines the timely and reliable access to information otherwise exchanged within the network in the attack-free case, hence greatly jeopardizing availability.

Note that, from the adversary's point of view, the attack described above is particularly convenient and preferable to more invasive alternatives such as, e.g., systematically jamming the cells used by nodes A and B. That is, if the adversary decided to jam the traffic of the victim pair altogether, this would have to be performed *repeatedly* and *regularly*, i.e., for all the cells in the schedule $\mathcal{S}$ and at every slotframe. This in turn has the following negative implications for the compromised node C.

First, node C would consume a much greater amount of resources, especially in terms of energy. Second, by acting as a recurring jammer, node C would become much more exposed, hence increasing the risk to be detected and consequently removed from the network, thus neutralizing the desired attack impact to a great extent. Third, node C would have in fact to repeatedly jam the target cells in $\mathcal{S}$, rather than possibly use them for its own traffic, i.e., on channel offsets allocated as per its own schedules associated with other neighbor nodes.

On the contrary, the type of attack presented in this section is performed only once on the victim pair, of which the two nodes are induced to unwittingly adopt different schedules. After that, node C can simply return to its own legitimate traffic as per its schedules, and possibly even negotiate with other peers some of the cells that have become effectively unavailable to use for the victim pair.

5.3. Advanced Handling of SeqNum Synchronization

When presenting the general attack rationale and execution earlier in Section 5, we have assumed that node A simply ignores an incoming 6P message allegedly from node B, if this conveys a SeqNum which: i) does not match against any ongoing 6P transaction with B; or ii) is not the right value for the next expected 6P transaction with B. This is consistent with the 6top standard [6], which leaves to specific implementations the possibility to define and enforce alternative, more advanced policies to handle such inconsistencies and other erroneous situations.

Although our assumption does in fact reflect current common practices, different implementations of the 6top standard might enforce a different error handling, upon receiving 6P messages that convey unexpected or inconsistent SeqNum values. For example, a node might consider a 6P response message that does not match with any ongoing transaction as a fatal inconsistency and explicitly react, e.g., by performing a 6P CLEAR transaction with the other peer or a schedule rollback.

In such a case, the attack execution requires to be extended with one additional action, in order to still ensure that the victim pair remains unable to detect it. In particular, the adversary must additionally suppress the 6P response messages sent by either node in the victim pair during the attack execution, e.g., by selectively jamming their transmission. With reference to Figure 4, this applies to: i) the response from node A to node B, during the 6P LIST transaction, in the first attack phase; and ii) the response from node B to node A, in the second attack phase.

By doing so, the adversary prevents the intended recipient from receiving those responses, and hence to detect their inconsistent SeqNum value that can trigger reactions and thus neutralize the attack. Furthermore, for each of those suppressed 6P responses, the adversary must also inject a forged ACK message addressed to the sender of that response. This prevents that sender node to possibly retransmit the response.

As all the nodes in the network have a synchronized shared view of slotframes and timeslots, it is also easy for the compromised node C to suppress those 6P responses through jamming. Moreover, the exact cells used to exchange 6P messages are well known to all the nodes in the network. Therefore, node C also has a full knowledge of the exact frequency to jam during any of such timeslots.

Evidently, suppressing those 6P responses during their transmission makes the adversary also unable to retrieve their content in the first place. Especially with reference to the 6P LIST transaction in the first attack phase, the adversary cannot rely on the response sent by node A to learn the original schedule $\mathcal{S}$ shared between A and B. Therefore, in such a case the adversary has to gain knowledge of the schedule $\mathcal{S}$ in a different way and before starting to perform the attack, e.g., by passively monitoring 6P transactions or other communications between A and B.

6. Traffic Dispersion Attack

This section presents the first specific incarnation of the general attack execution discussed in Section 5, namely the *Traffic Dispersion* attack.

The goal of the Traffic Dispersion attack is to disrupt communications from a victim node A to its neighbor node B, by *dispersing* messages that A sends to B according to the schedule that

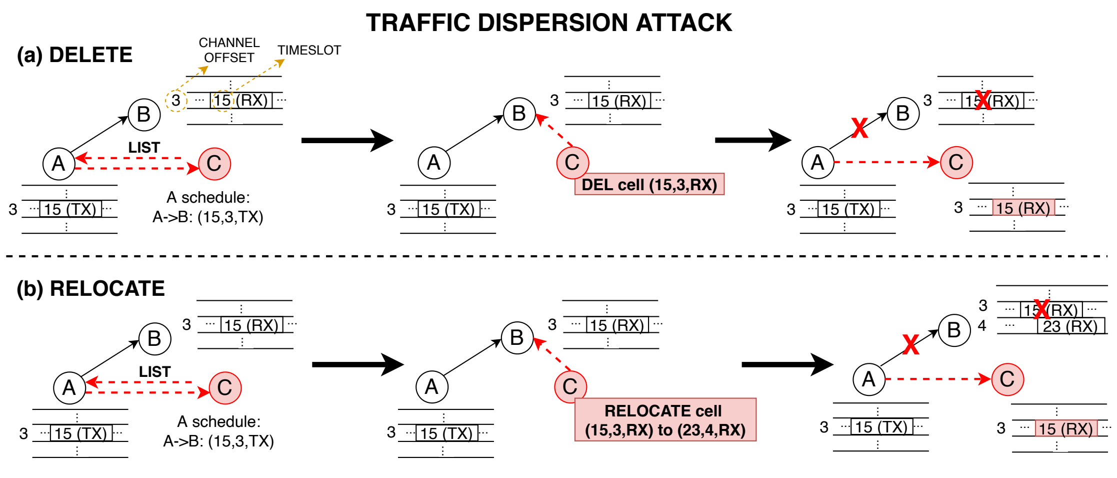


Figure 5: Schedule manipulation through the Traffic Dispersion attack.

the two nodes (believe to) consistently share. In the DODAG built by RPL, node A has node B as preferred parent. The attack execution results in thwarting the propagation of information from the nodes to the root node, which acts as the data sink node for final information processing and release. In turn, this yields the disruption of access to and use of information collected in the network, hence greatly jeopardizing availability.

The adversary can achieve the attack goal by performing two different variants of the Traffic Dispersion attack, depending on the specific 6P transaction considered during the second attack phase (see Figure 4). In particular, the first attack variant relies on a 6P transaction DELETE, and is hereafter referred to as *DELETE attack*. Instead, the second attack variant relies on a 6P transaction RELOCATE, and is hereafter referred to as *RELOCATE attack*.

Consistently with the attack general scheme presented in Section 5, the Traffic Dispersion attack is composed of two phases. In the first phase, the adversary forges a 6P LIST transaction with node A, and exploits it for retrieving the original schedule $\mathcal{S}$ currently used by nodes A and B. Then, in the second phase, the adversary induces B to install an alternative schedule $\mathcal{S}'$. To this end, the adversary impersonates node A and performs with node B either a 6P transaction DELETE, in the DELETE attack, or a 6P transaction RELOCATE, in the RELOCATE attack.

Figure 5 shows how a compromised node C under the adversary's control practically performs the Traffic Dispersion attack, i.e., the variant DELETE attack in Figure 5a or the variant RELOCATE attack in Figure 5b. In particular, the figure highlights how node B shifts from the original schedule $\mathcal{S}$ to the alternative schedule $\mathcal{S}'$ installed during the attack. After that, node B considers the schedule $\mathcal{S}'$ for communicating with node A. Since the schedule $\mathcal{S}'$ has no cells in common with the schedule $\mathcal{S}$, node B is not able to receive any transmission from A.

More specifically, Figure 5a shows that, before the DELETE attack is performed, nodes A and B share a schedule $\mathcal{S}$ composed of one common cell $c1$. The cell is associated with timeslot 15 and channel offset 3, and is used in transmission mode (TX) by A as well as in reception mode (RX) by B. Then, due to the bogus 6P DELETE transaction, B is induced to delete such only cell shared with A. That is, B installs the alternative empty schedule $\mathcal{S}'$ as believed to be shared with A. Practically, B does not expect to engage in any communication with A. On the other hand, A preserves the original schedule $\mathcal{S}$. Finally, the adversary starts being active in RX during cell $c1$.

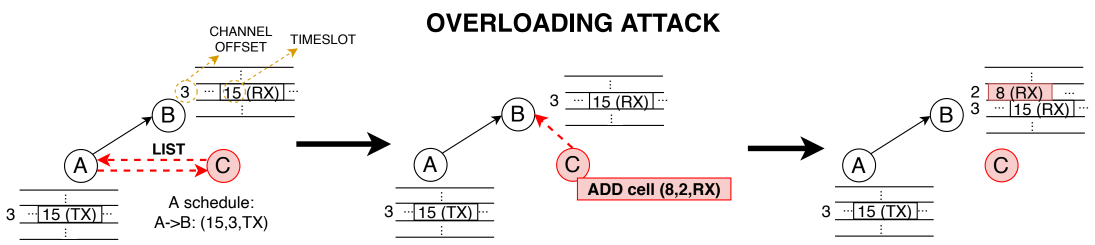


Figure 6: Schedule manipulation through the Overloading attack.

More generally, C allocates in its own schedule the same cells that have been removed from the schedule of B. From then on, upon receiving messages from A and addressed to B during those cells, C transmits an ACK to A. As a consequence, A will erroneously believe communications with B to be successful, without suspecting possible anomalies or schedule de-synchronization which might otherwise result in a new negotiation. As ultimately intended by the adversary, all messages from A to B are not going to be received by B.

Similarly, Figure 5b shows that, before the RELOCATE attack is performed, nodes A and B share the same schedule $\mathcal{S}$ composed of one common cell $c1$, to be used as defined above. Then, due to the bogus 6P RELOCATE transaction, B is induced to replace the only cell shared with A, with a different cell $c2$. The cell is associated with timeslot 23 and channel offset 4, and is still intended to be used for transmitting by A as well as for receiving by B. Thus, B installs an alternative schedule $\mathcal{S}'$ including only the cell $c2$, and practically expects further communications from A on that cell from then on. On the other hand, A preserves the original schedule $\mathcal{S}$. After that, C starts listening on cell $c1$, and performs as described above for the DELETE attack. It follows that, also in this case, messages from A to B are not going to be received by B.

7. Overloading Attack

This section presents the second specific incarnation of the the general attack execution discussed in Section 5, namely the *Overloading* attack.

The goal of the Overloading attack is to make a victim node B allocate unneeded cells to be used in reception mode (RX), with the ultimate intent to increase the victim's energy consumption. Like in Section 6 for the Traffic Dispersion attack, the DODAG built by RPL is such that node A has node B as preferred parent. The attack execution results in inducing the additional allocation of unneeded communication resources (6TiSCH cells) on the victim nodes, thus preventing a more appropriate allocation and energy-aware use of resources. This yields additional and unnecessary energy consumption, which severely reduces the operational capability and lifetime of the victim nodes, hence greatly jeopardizing availability.

The adversary can achieve this attack goal by adding more cells in reception mode (RX) to the schedule that the victim node B believes to consistently share with its neighbor node A. More generally, the adversary induces node B to add cells to its (believed-to-be-)shared schedule with multiple neighbor nodes. In order to achieve this goal, the adversary relies on a 6P ADD transaction during the second attack phase (see Figure 4).

More specifically, Figure 6 shows how, before the Overloading attack is performed, nodes A and B share a schedule $\mathcal{S}$ composed of one common cell $c1$. The cell is associated with timeslot 15 and channel offset 3, and is used in transmission mode (TX) by A as well as in reception mode (RX) by B. Then, due to the bogus 6P ADD transaction, B is induced to add to the schedule with A one more cell $c2$, to be used in RX on timeslot 8 and channel offset 2. That is, B installs an alternative schedule $\mathcal{S}'$ as believed to be shared with A, including both cells $c1$ and $c2$, and practically expects further communications from A on both cells from then on. Instead, A preserves the original schedule $\mathcal{S}$. Note that, unlike in the Traffic Dispersion attack, C does not require to perform any further actions such as the injection of ACK messages addressed to A.

As a consequence, B will regularly be active in RX also during cell $c2$, at every slotframe. This in turn has two practical effects. First, as regularly listening for potential incoming messages during cell $c2$ at every slotframe, B will regularly consume additional energy to no avail, i.e., while never receiving any transmission from A during such cell. Second, B would be effectively *wasting* a cell, that could rather have been considered for negotiating a schedule with a different neighbor.

8. Analytical Model

In order to assess the influence of the attacks on the network performance, we developed an analytical model to estimate the impact of each attack.

This model can be used by the attacker to configure the specific attack parameters or select the proper area to target in the overall network, in order to attain the desired attack impact. At the same time, this model can be used by the network designer and operator, in order to estimate the possible maximum impact to expect, and accordingly judge whether that is acceptable to (temporarily) endure or countermeasures and mitigations are to be adopted (e.g., see Section 11).

In the following we present two analytical models, one for the Traffic Dispersion attack and one for the Overloading attack, respectively. By construction, the analytical models are simplified and cannot grasp the complexity of the network as a whole. For this reason, a detailed assessment of the impact of both the attacks is carried out via simulation in Section 9.

8.1. Traffic Dispersion

The Traffic Dispersion attack concerns any packet traversing the victim node, i.e., including those generated by its children and forwarded upstream. Consequently the attack has an impact on the overall packet delivery experienced in the network. In order to estimate the impact of the attack on the whole network, in the following we model the network Packet Delivery Probability (PDP) when the attack is performed.

Let us consider a network composed of N nodes, where traffic is delivered towards a root node via multi-hop delivery, driven by the RPL routing protocol. RPL builds a logical tree topology that can be represented with an oriented graph $G = (V, E)$, where $V = \{n_0, n_1, n_2, ...n_{N-1}\}$ is the set of nodes and n_0 is the root node, while any n_i, with $1 \leq i \leq N-1$, is the i-th node of the network, and $|V| = N$ is the number of nodes in the network. For each node n_i let us define p_i as its preferred parent. Each node n_i, is connected to its preferred parent p_i with a link, $(n_i, p_i) \in E \subset V \times V$, therefore $E = \{(n_1, p_1), (n_2, p_2), ..., (n_{N-1}, p_{N-1})\}$ is the set of links between nodes in the network, and thus $|E| = N - 1$. Let us define $F = \{F_0, F_1, F_2, ...F_{N-1}\}$ as the set of all the traffic flows in the network. Every i-th node with $1 \leq i \leq N-1$ originates a flow

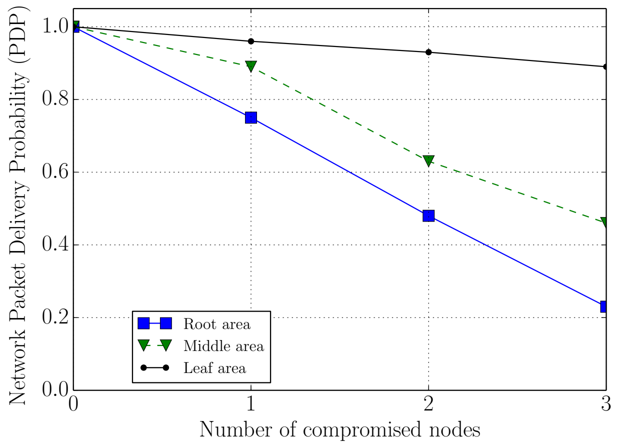


Figure 7: Analytical Network PDP for the Traffic Dispersion attack.

to send its data towards the root node n_0 by following the path $F_i = \{(n_i, p_i), ...(n_x, p_x)\} \in F$, where $p_x = n_0$, selected according to the RPL protocol.

When considering the Traffic Dispersion attack, the victim node A: (i) sends the data packets of the flow it generates; and (ii) forwards the data packets of the flows traversing itself to its preferred parent B. Node A has a schedule $\mathcal{S}_A$ where $N_{\mathcal{S}_A}$ TX cells are allocated for the transmission of data to B. On the other end, node B has an initial schedule $\mathcal{S}_B$, where $N_{\mathcal{S}_B}$ RX cells are allocated for the reception of data from A. It is worth noting that $N_{\mathcal{S}_A}$ TX cells and $N_{\mathcal{S}_B}$ RX cells are allocated on the same timeslot and channel offset. Due to the attack execution, an alternative schedule $\mathcal{S}'_B$ is installed on node B, where some or all of the RX cells allocated to receive data from node A are relocated or deleted. Let us define $N'_{\mathcal{S}_B}$ as the number of cells on $\mathcal{S}'_B$ that have a corresponding TX cell in $\mathcal{S}_A$, in terms of timeslot and channel offset. That is, $N'_{\mathcal{S}_B}$ is the number of cells where successful data transfer can still be performed from node A to node B after the attack execution.

The Packet Delivery Probability in the hop between A and B (PDP_{AB}), i.e., the probability that any packet that A sends to B is correctly received by B, can be defined as follows:

$$PDP_{AB} = Plink_{AB} \frac{N'_{\mathcal{S}_B}}{N_{\mathcal{S}_A}} \tag{1}$$

where $Plink_{AB}$ is the (A,B) link loss probability due to the channel quality (as influenced, e.g., by interference, distance, noise, etc.), while the term $(N'_{\mathcal{S}_B}/N_{\mathcal{S}_A})$ models the additional loss related to the attack execution.

Therefore, an estimate of the network PDP can be obtained as follows:

$$PDP = \frac{\sum_{F_i \in F} PDP_{F_i}}{|F|}. \tag{2}$$

where

$$PDP_{F_i} = \prod_{\forall (n_j, p_j) \in F_i} PDP_{n_i p_j}. \tag{3}$$

Specifically, the network PDP estimate is computed as the average of the PDPs of all the flows in the network, where PDP_{F_i} denotes the PDP of the flow F_i. In turn, PDP_{F_i} is computed as the combined probability that a packet is successfully transmitted on each link of the flow F_i. Specifically, PDP_{F_i} is obtained by multiplying the PDPs $PDP_{n_i p_j}$ of each link (n_i,p_j) in the flow $F_i \in F$, i.e., from a node n_i towards its parent p_j.

By relying on the presented analytical formulas, we now analyze how the impact of the attack is affected by the number of compromised and victim nodes and by the position of the victim nodes in the network. Specifically, we vary the number of compromised nodes in the range [1, 3], and we consider three areas in which the victim nodes can be positioned, i.e., one hop from the root node (i.e., the root area), more than one hop from the root node (i.e., the middle area), and up to the periphery of the network considering only nodes with no children (i.e., the leaf area). In the following, we assume that there is no source of external interference, i.e., $Plink_{AB} = 1$. Thus, when no attack is performed, all the packets sent by the network nodes are correctly received by the sink node.

Figure 7, shows the results obtained when considering a network topology composed of 30 nodes, where each non root node generates one traffic flow. The values reported in the graph represent the expected PDP of the whole network, when the attack is not performed (i.e., there are no compromised nodes) or when one or more compromised nodes are present and perform the Traffic Dispersion attack, for different network areas. As shown in the graph, the impact of the attack is bigger when the number of compromised nodes increases, and when the victim nodes are positioned in the root area or in the middle area. When considering three compromised nodes and the victim nodes as leaf nodes, our model shows that the reduction of PDP is at most of about 15%, while it reaches a reduction of almost the 80% when considering the same number of compromised nodes, rather positioned in the root area.

8.2. Overloading

The Overloading attack main impact is the increase of the energy consumption on the victim nodes. For this reason, in order to estimate its impact, we model the additional energy consumption due to the attack execution.

Let us consider a victim node B, with an initial schedule $\mathcal{S}_B$. The node B has a set of child nodes $C = \{c_0, c_1, ..., c_{N_c-1}\}$ where $|C| = N_c$ is the number of child nodes, i.e., the nodes that selected B as preferred parent. For each $c_i \in C$, node B has a certain number of RX cells $N_{\mathcal{S}_B c_i}$ allocated to receive its traffic. Due to the attack execution, an alternative schedule $\mathcal{S}'_B$ is installed on node B, where a certain number of RX cells $N'_{\mathcal{S}_B c_i} > N_{\mathcal{S}_B c_i}$ for each child $c_i \in C$ is allocated. This number of additionally allocated cells results in an increase in the energy consumption of node B. It is worth highlighting that, while these cells yield additional energy consumption, node B never receives any transmission during them from any of its child node $c_i \in C$. Compared to the attack-free case, the *extra* RX energy consumption experienced by node B in a single slotframe can be estimated as follows:

$$E_B = \sum_{c_i \in C} (N'_{\mathcal{S}_B c_i} - N_{\mathcal{S}_B c_i}) \cdot E_{cell} \tag{4}$$

In particular, $E_{cell} = W_{RX} \cdot T_{RX}$ is the RX energy consumption for a single RX cell of node B, where W_{RX} is the power consumption in RX state and T_{RX} is the time that node B spends in RX

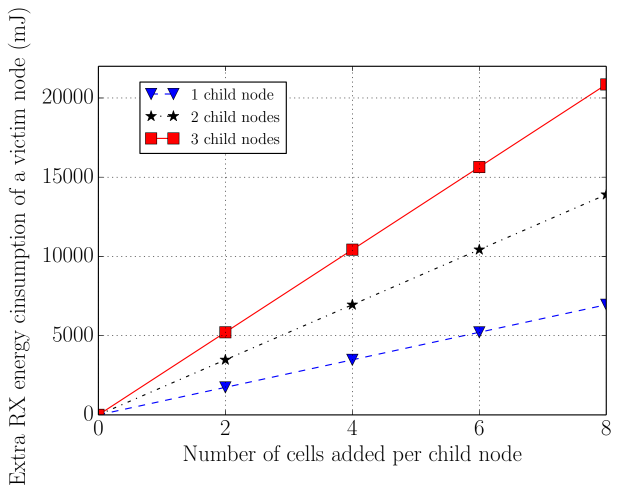


Figure 8: Analytical extra RX energy consumption of a victim node for the Overloading attack.

state during an RX cell. With particular reference to the additional RX cells allocated due to the Overloading attack, T_{RX} is actually the amount of time the node B waits for potential incoming transmissions during an RX cell, before turning off its radio in case none is detected. Practically, as per [10], $T_{RX} = 3.320ms$.

In order to compute the extra energy consumption experienced by node B, we consider a current of $20mA$ and a Voltage of $3.3V$. These are the reference values for the Zolertia RE-Mote board [26], which will be the board used for the experimental evaluation in Section 10 as well. Thus the instantaneous power consumption of node B in RX state is equal to $W_{RX} = 20mA \cdot 3.3V = 66mW$. It follows that the energy consumption for each RX cell added as a consequence of the attack is $E_{cell} = 66mW \cdot 3.320ms = 0.22mJ$.

Figure 8 shows the increase in energy consumption experienced by the victim node B over 20 minutes, after the Overloading attack is performed at time 0. In order to compute the energy consumption increase, we consider a slotframe length of 29 timeslots, i.e., the slotframes repeat over time every 290 ms, and we vary the number of the victim's child nodes in the range [1, 3]. Thus, when 2/4/6/8 RX cells *per child* are added due to the attack execution, the new schedule $\mathcal{S}'_B$ installed on node B by the compromised node has 2/4/6, 4/8/12, 6/12/18 or 8/16/24 additional and unneeded RX cells, respectively. As shown in Figure 8, when considering three child nodes, the maximum increase in the energy consumption of the victim node B is about 21000 mJ, while the minimum increase, when considering only one child node, is 1737 mJ.

9. Performance Evaluation of the Attacks

In order to assess the impact of the Traffic Dispersion and Overloading attacks on the network performance and reliability in a realistic scenario, we have performed an extensive performance evaluation, based on both simulation and real experiments. Simulations allow us to swiftly consider a large set of scenarios, characterized by different number of compromised nodes, different location of victim nodes, and different network configurations. Instead, measurements on a real IoT testbed allow us to demonstrate the feasibility of the attacks in practice and to validate the simulation results and evaluate the impact of the attacks in a real deployment with a real channel model.

In this section, we present the simulation setup and discuss the results obtained in the various considered scenarios. The experimental results obtained through measurements in the real testbed are presented later in Section 10.

9.1. Simulation Setup

In particular, we simulated a network composed of 30 nodes, randomly deployed in an area of 200m x 200m. For each node, we used the Cooja mote type, i.e., the simulator was configured to model a network of Cooja motes (generic IoT nodes equipped with an IEEE 802.15.4 radio), and used the Unit Disk Graph Medium (UDGM) as radio model. This channel model, which provides 100% packet delivery under the radio reception range of a node, was adopted in order to better assess the effects of the attacks, the Traffic Dispersion in particular. A set of experiments with a real testbed is carried out as well, in order to confirm the results of simulations in a real environment.

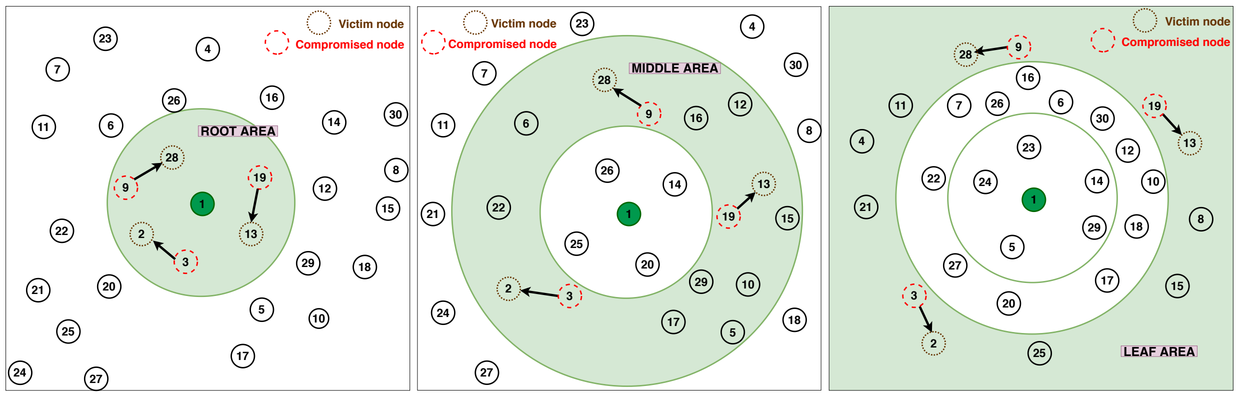
For our analysis, we implemented the execution of both attacks in the Contiki-NG Operating System (OS)[1], a popular OS for IoT devices. One significant advantage of Contiki-NG is the Cooja [27] network simulator, which is available as part of the Contiki distribution. Cooja allows the execution of the same code running in real devices, thus reproducing a realistic environment.

In our experiments, we considered a periodic data reporting application, e.g., for environmental monitoring, following the convergecast communication paradigm [28]. Specifically, we considered an IoT network where nodes periodically report their data to the root node of the DODAG, behaving as traffic sink. Each (non-root) node is programmed to generate periodic traffic towards the root node, at a rate of one 5-byte message every 2 seconds. The UDP protocol is used at the transport layer.

TSCH was configured with a slotframe composed of 29 timeslots and 16 different channels. At bootstrap, the 6top sublayer allocates 3 shared cells for 6P messages and 1 more shared cell for TSCH/RPL control traffic. The latter cell can be used also for data traffic, if no other cells are available. The remaining cells are managed by MSF to ensures the cell allocation according to the traffic conditions. Since the evaluated attacks are expected to affect the communication reliability and the resource consumption of the victim node, we considered the following metrics in order to quantify the attack impact.

• *Packet Delivery Ratio (PDR)* over time in percentage, defined as the ratio between the total number of packets received at the root node and the total number of packets generated by network nodes, calculated as follows: $PDR = \frac{N_{PR}}{N_{PS}} \cdot 100$, where N_{PS} is the number of packets sent by all the nodes, while N_{PR} is the number of packets received by the root node in the considered time period. The metric is calculated in consecutive time windows of 60s throughout the experiment in order to show the value over time before and after the activation of the attack. This metric provides a direct measurement of the overall network communication reliability over time.
• *RX Energy Consumption*, defined as the total amount of energy spent by a node in RX state, i.e., when its radio is in RX mode during the overall experiment. It measures the total energy

[1]https://github.com/contiki-ng/contiki-ng



(a) Root area. (b) Middle area. (c) Leaf area.

Figure 9: Traffic Dispersion attack with different positions of the victim nodes.

consumed by a node for RX cells.

• *Cell Consumption*, defined as the ratio between the total number of (shared, RX and TX) cells allocated by a node and the slotframe length, reported in percentage. This metric provides a direct measurement of the communication resources consumed by a node (in cells) with respect to the overall bandwidth potentially available.

In order to collect and analyze energy consumption of each node, we relied on the Energest tool[2]. Energest is a Contiki-NG module that provides a lightweight, software-based energy consumption estimation of Contiki-NG devices, by keeping track of the time they spend with their radio on, in RX or TX mode. To derive precise energy consumption values, we considered the power consumption reported in the hardware specifications of the cc2538 radio chip[3]. While values extracted through Energest are an estimate, they are proven to be reliable [29][30].

In our analysis, we considered different scenarios, in which we varied the number of compromised nodes and the location of the victim nodes within the network. With reference to Figure 9 and Figure 16, we consider the node with ID = 1 as DODAG root, and we define three areas in which victim nodes can be positioned, namely *root* area, *middle* area and *leaf* area. The root area includes all the nodes that are in the communication range of the root node. Instead, nodes located in the middle area are nodes that (i) are, at least, two hops away from the root node; and (ii) have, at least, one child node. Finally, nodes in the leaf area are at least two hops away from the root node, but they do not have any children, as they are located at the periphery of the network.

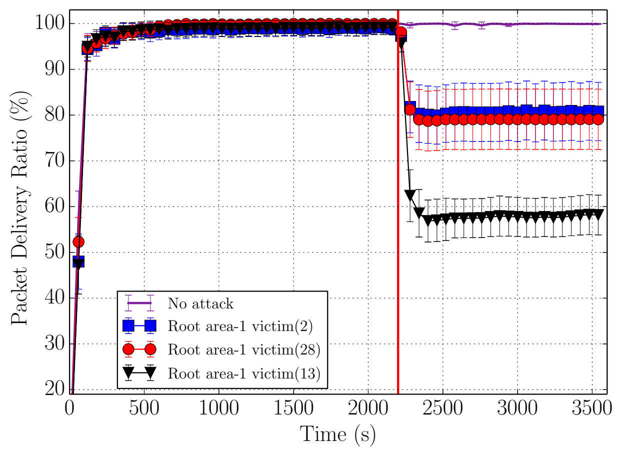
In order to obtain statistically sound results, for each experiment we ran 30 independent replicas of 1-hour duration each. The results presented below are averaged over all the replicas. We also show confidence intervals, obtained with a 95% confidence level.

9.2. Impact of the Traffic Dispersion Attack

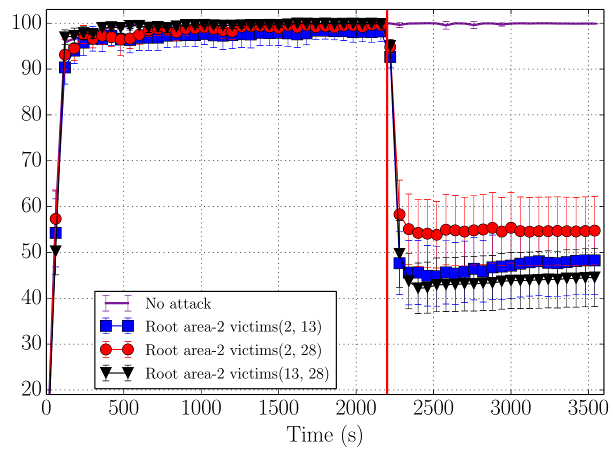
In this section, we investigate the impact of the Traffic Dispersion attack. Figure 9 shows the three different cases that we consider when evaluating this attack, where the victim nodes are positioned in different areas of the network. Specifically, we will analyze the impact in the

[2]https://github.com/contiki-ng/contiki-ng/wiki/Documentation:-Energest

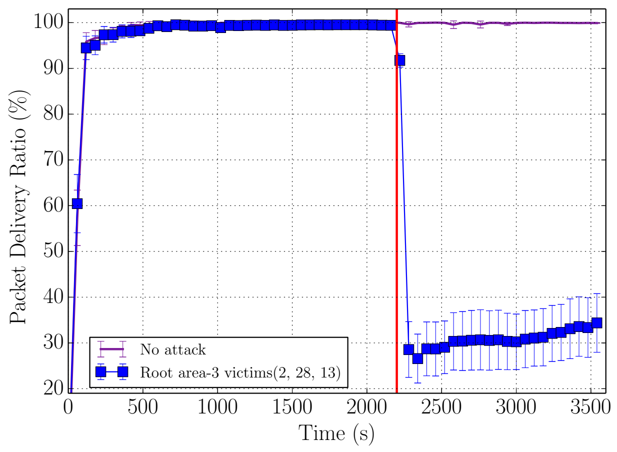
[3]http://www.ti.com/lit/ds/symlink/cc2538.pdf



(a) One victim node.



(b) Two victim nodes.



(c) Three victim nodes.

Figure 10: PDR with Traffic Dispersion attack. Root area.

presence of one, two or three victim nodes (and as many compromised nodes) in the root area (see Figure 9a), in the middle area (see Figure 9b) and in the leaf area (see Figure 9c).

We consider one victim node per compromised node, and assume that each compromised node targets exactly a single victim node. In the presence of multiple victim nodes,we also evaluate how the attack mounted by different compromised nodes can mutually amplify one another and results in an overall wider impact. In Figure 9, we also report the victim nodes that have ID 2, 13 and 28, while the corresponding compromised nodes have ID 3, 19 and 9.

For the root area and the middle area, we consider different positions of each victim node and analyze different combinations for the cases with more than one victim node. Specifically, with reference to Figure 9a and Figure 9b, in the case with one victim node we consider, as victim, first node 2, then node 28 and, finally, node 13. Instead, in the case with two victim nodes, we consider the following victim pairs: nodes 2 and 13, nodes 2 and 28, and, finally, nodes 13 and 28.

The rationale behind this is that a victim node in the root area or in the middle area could have a different number of children, which could affect the impact of the attack. Instead, if the victim node is positioned in the leaf area, it does not have any child. Therefore, the attack will only affect the performance of that node, and hence the overall impact on the network does not depend on the specific victim node.

9.2.1. Root Area

We start our analysis by considering the scenario where the victim and compromised nodes are located in the root area (see Figure 9a), and hence they have a direct link with the root node. Figure 10 compares the impact of the attack in terms of PDR, with one (Figure 10a), two (Figure 10b) or three (Figure 10c) victim nodes. When the attack is performed by a single node, we separately evaluate its impact for three different positions of the victim node (i.e., node 2, node 28 and node 13). As shown in Figure 10a, if the attack is performed by node 3 against node 2, or by node 9 against node 28, we observe a similar impact on the PDR, which decreases by approximately 20%, when the attack is activated. Instead, if the attack is performed by node 19 against node 13, the impact on the PDR is more highlighted, as the attack yields at least a 40% PDR reduction. This is because node 13 has several children in its subtree, from which it receives a significant amount of traffic (see Figure 9a). Since the cells shared by node 13 with its preferred parent (i.e., the root node) are deleted as part of the Traffic Dispersion attack, all the packets from the subtree of node 13 are not going to be successfully delivered to the root node.

Figure 10b shows the PDR over time, showing its reduction when the Traffic Dispersion attack is performed by two compromised nodes simultaneously, e.g., nodes 3 and 9 against nodes 2 and 28, respectively (red line with circles). As expected, the impact is now more relevant than in the previous case. As above, also in the case of two compromised nodes, when node 13 is among the victims the impact on the PDR is the highest, as shown by the black and blue lines. In these two cases, the PDR decreases by, at least, 50%.

Finally, Figure 10c shows the impact with all the three victim nodes in the root area. In this configuration, the attack reduces the PDR by at least 65%. By looking at network topology (see Figure 9a) and considering that the victim nodes are the only nodes that forward data packets to the root node (the compromised nodes do not join the DODAG and do not forward packets), we would expect a 100% reduction of the PDR from the attack. The reason for the lower reduction of PDR caused by the attack is due to the availability of the shared cells scheduled by all the nodes for control traffic. As mentioned in Section 9.1, these shared cells can be used also for transmitting data packets, when no other cells are available. In our case, this results in about 30% of data packets to be delivered, even when the attack is performed, since these cells are not affected by the attack.

It is worth highlighting that the width of the confidence intervals reflects a non-negligible variability of the presented results. We investigated this behavior and found that the dynamics of the RPL protocol have a considerable effect on the impact of the attack, i.e., they can amplify or reduce its effectiveness. This happens because the victim nodes in the root area have several neighbors, and, due to the RPL protocol [31], the network topology may change considerably across different replicas and even within the same replica. Even slight changes in the network conditions (e.g., a temporary increase in the packet loss), may have an impact on the link quality estimation performed by RPL, and hence on the network topology.

Even though an in-depth analysis of the reasons why RPL builds different topologies for the same network is out of scope for this work, it is worth having an insight on the different

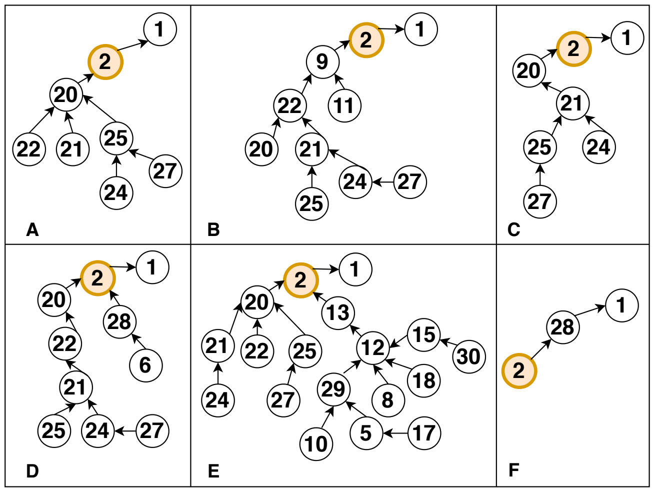


Figure 11: Most frequent RPL topologies built for the same scenario, with one compromised node (ID = 3) and one victim node (ID = 2).

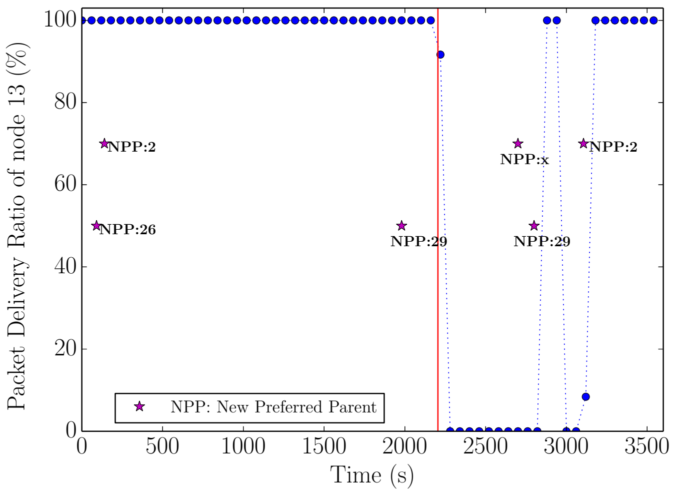


Figure 12: Example of changes of preferred parent before and after the Traffic Dispersion attack.

topologies that are built over different replicas. For simplicity, we analyze only the scenario with a single compromised node (node 3) and one victim node (node 2). However, we observed the same behavior also for all the other considered scenarios with the victim nodes located in the root area.

Figure 11 shows the six different topologies that occur most frequently, across all the simulation replicas for the same scenario. It can be seen that the number of nodes in the subtree of the victim node (node 2) varies significantly. In the topology shown in Figure 11E, 17 nodes are affected by the attack, i.e., all the nodes in the victim's subtree, plus the victim itself. Instead, in the topology in Figure 11F, only the victim node is affected. Consequently, when the attack is performed in the same scenario with the topology in Figure 11E, its impact will be very high; instead, when performed with the topology in Figure 11F, its impact will be minimum.

Another event triggered by the RPL protocol, i.e., the preferred parent change, can influence

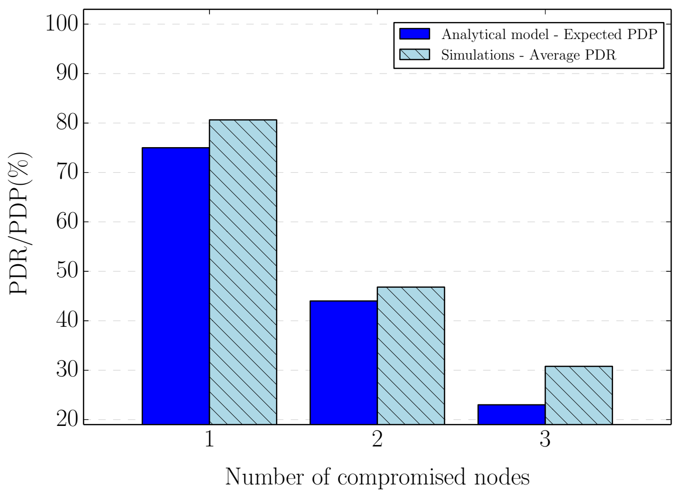
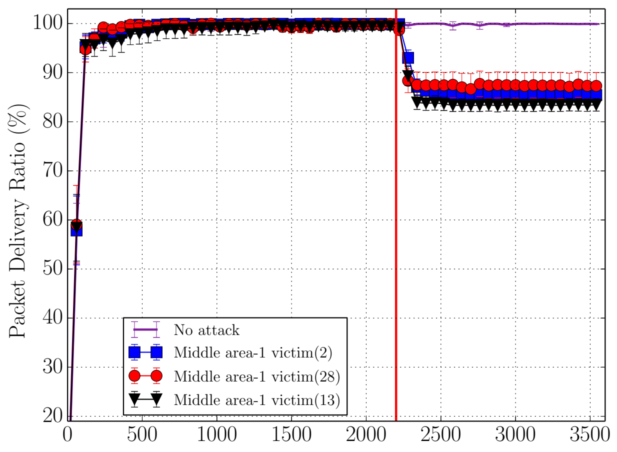


Figure 13: Analytical model and simulation comparison for the Traffic Dispersion attack - Root area.

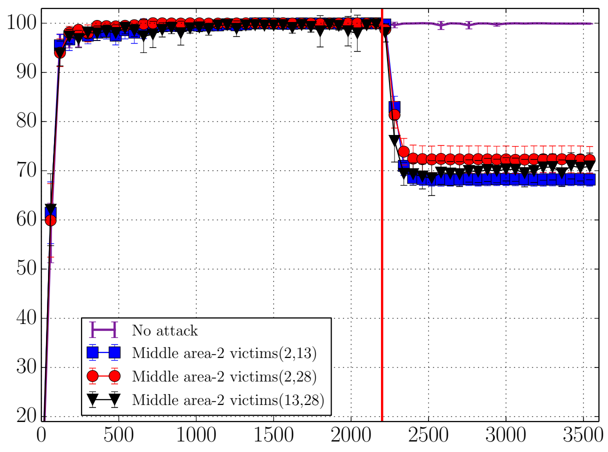
the impact of the attack, if it takes place after the attack is activated. In order to show this, we consider a simulation run with a single compromised node (node 19) and one victim node (node 13). As shown in Figure 12, the victim node has a PDR equal to 100% before the attack, while it is goes to the minimum (i.e., 0%) just after. At time 2700 seconds, i.e., 500 seconds after the attack execution, the victim node decides to change its preferred parent (Next Preferred Parent (NPP):x in Figure 12) due to changed network conditions (e.g., unresponsiveness of the preferred parent or packet collisions). However, RPL is not able to find a new preferred parent that guarantees a better rank to the victim. Hence, shortly after, the victim node selects again node 29 as its preferred parent. This change results in both the victim node and its preferred parent clearing their schedule and negotiating new cells from scratch, by performing 6P transactions. With the victim having cleared its (bogus) schedule, the originally mounted attack becomes ineffective, and hence the effects of the attacks are canceled, i.e., the PDR goes back to 100%. In the considered simulation run, after some minutes the PDR of the victim node starts to decrease again (due to bad network conditions), and the node changes again its preferred parent to node 2, which yields 100% PDR.

The example above shows that a change of preferred parent, when performed after the execution of the attack, makes the attack ineffective from then on. Since such an event may occur frequently in IoT networks due to time-varying network conditions, this can indirectly reduce the impact of the Traffic Dispersion attack.

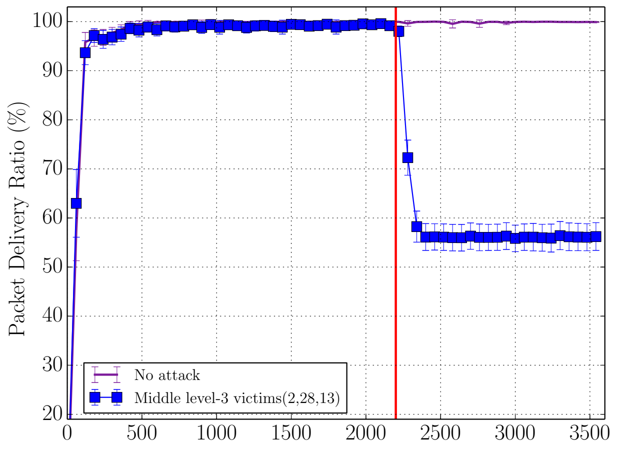
Figure 13 shows the comparison of the analytical and the simulation results, when considering different numbers of compromised nodes. In order to evaluate the model, we consider the topology in Figure 9a, and use a DODAG built by RPL after the network formation phase. This means that the topology will be fixed for the analytical model, while it can vary over time in a simulation run. This is the reason for the difference in terms of expected PDP and Average PDR, shown in Figure 13. Here it is worth highlighting that the expected PDP can be translated in the PDR, since it measure the probability for each packet to be correctly delivered to its destination. Nevertheless the difference, in terms of PDR, for the analytical model and the simulation can be around 30% in the case of 3 compromised nodes, this can be considered an effect of the dynamism of the RPL protocol that is not captured in the model. Hence, the analytical model



(a) One victim node.



(b) Two victim nodes.



(c) Three victim nodes.

Figure 14: PDR with Traffic Dispersion attack. Middle area.

provides the maximum impact that the Traffic Dispersion attack can have on the reduction of the network PDR, that strictly depends also on network conditions. We only show the comparison for the root area, since the results have the same trend for the other considered areas.

9.2.2. Middle Area

In order to analyze the impact of the Traffic Dispersion attack in the middle area, we consider the case depicted in Figure 9b. In this scenario, the victim nodes are more than two hops away from the root node. As shown in Figure 14, the impact of the attack still depends on the considered number of compromised nodes and victim nodes. However, it is generally lower than in the previous case where the victim nodes are in the root area (see Section 9.2.1). Specifically, now the PDR decreases by (10-18)% in the presence of one compromised node (see Figure 14a), (28-38)% with two compromised nodes (see Figure 14b), and (40-45)% with three compromised nodes (see Figure 14c). This happens because victim nodes in the middle area have less children, and consequently less packets are lost due to the attack. In addition, we can observe that there is much less variability in the obtained results. This is because the victim nodes are less impacted by the variability introduced by the RPL protocol, due to the lower number of their neighbors.

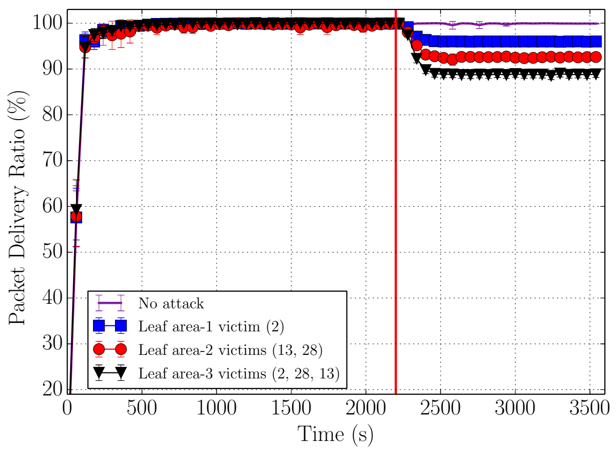


Figure 15: PDR with Traffic Dispersion attack. Leaf area.

9.2.3. Leaf Area

Finally, we consider the case where the victim nodes and the compromised nodes are positioned in the leaf area. In this scenario, we did not run simulations for different victim pairs, since it is not relevant for this case. In fact, nodes in the leaf area do not root any subtree (see Figure 9c), and thus each of such nodes only sends its own self-generated traffic towards the root node.

The results in Figure 15 show that the impact of the attack is dramatically reduced in this scenario, i.e., the PDR reduction is at most 12%. This happens because the victim node does not have any children, and hence it marginally contributes to handle the overall network traffic. In addition, we can notice that the variability is now very limited, because the victim nodes are positioned in the leaf area (i.e., at the network periphery), and thus are not significantly affected by the dynamism of the RPL protocol.

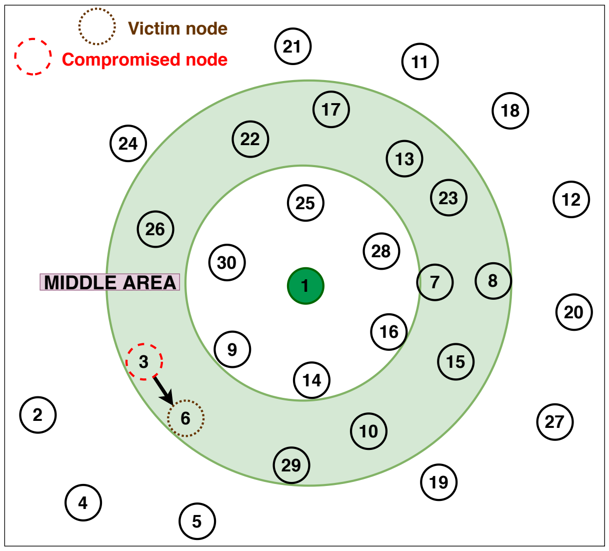
9.3. Impact of the Overloading Attack

In this section, we analyze the impact of the Overloading attack. As above, we have performed an in depth analysis by considering two different network topologies, shown in Figure 16a.

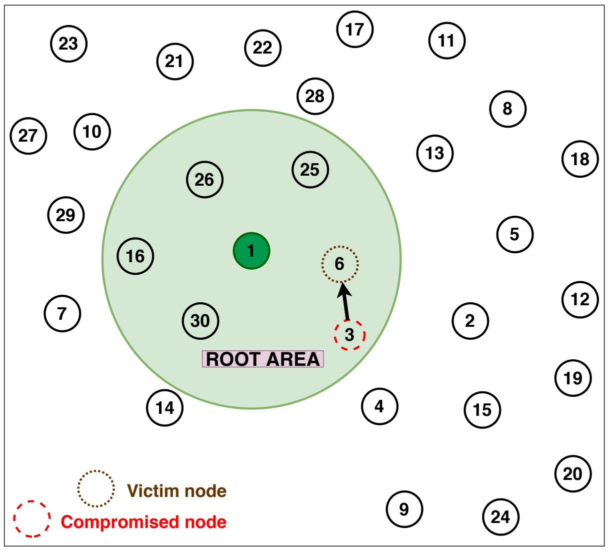
We recall that, in the Overloading attack, the victim node is forced to add a number of RX cells in its schedule. Those RX cells are meant to make the victim node listen to the medium in vain, since there are no corresponding allocated TX cells at the other peer. Therefore, the victim node is actually not going to receive any packet during such additional RX cells, but yet to consume energy.

It is worth to mention that from the energy consumption point of view, scheduling additional unneeded TX cells would not have the same effect of scheduling unneeded RX cells. In fact, while a node always switches from idle to RX when using RX cells, it does not switch to TX mode unless there is actual data to be sent.

Figure 16a shows the case where the victim nodes and the compromised nodes are positioned in the middle area, while Figure 16b refers to the case where the victim nodes are in the root area. For the Overloading attack, we do not consider the case where the victim node is positioned in



(a) Middle area.



(b) Root area.

Figure 16: Overloading attack with different locations of the victim nodes.

the leaf area, as this attack is not effective when the victim is a leaf node in the DODAG. In fact, if a node has no children, it is going to schedule only TX cells, but no RX cells. Since adding TX cells does not affect the energy consumption of the victim, we will not consider this case in our analysis.

9.3.1. Middle area

We start our analysis considering the victim node (i.e., node 6) in the middle area. Figure 17 shows the RX energy consumed by the victim node when the attack is not performed (purple thick line) and when the compromised node adds an increasing numbers of RX cells to the schedule of the victim node. The traffic is assumed to be periodic and packet generation period is equal to 2s.

If we consider the overall RX energy consumption (i.e., the value at the end of the simulation experiment of 1-hour duration), there is a considerable increase when 2 cells are added. Specifically, the additional energy consumed by the victim node due to the attack is in the order of 6.7 J, i.e., 32% more than in the attack-free scenario. When adding more RX cells, the energy wastage is higher, i.e, in the order of 11.8 J (+ 45%) with 4 cells, and of 18.2 J (+55%) with 6 or more cells. It is worthwhile observing that this additional energy is consumed in the time interval between the attack execution (at time 2200s) and the end of the experiment (at time 3600s), corresponding to about 23 minutes. Also, we want to emphasize that, once the unneeded RX cells have been installed in the victim's schedule, the number of such cells does not affect the energy consumption of the compromised node.

The results in Figure 17 show that there is no further significant increase in the energy wastage of the victim node when more than 6 cells are added. This can be explained by considering that, in our experiments, we assumed a TSCH slotframe length equal to 29 timeslots. Hence, a node can use at most 29 cells per slotframe. When the victim node has 3 children, adding 6 (8) RX cells per child results in 18 (24) additional RX cells. Considering that there are other legitimately scheduled cells (at least 4 in each node for control traffic), adding 6 or more RX cells per child

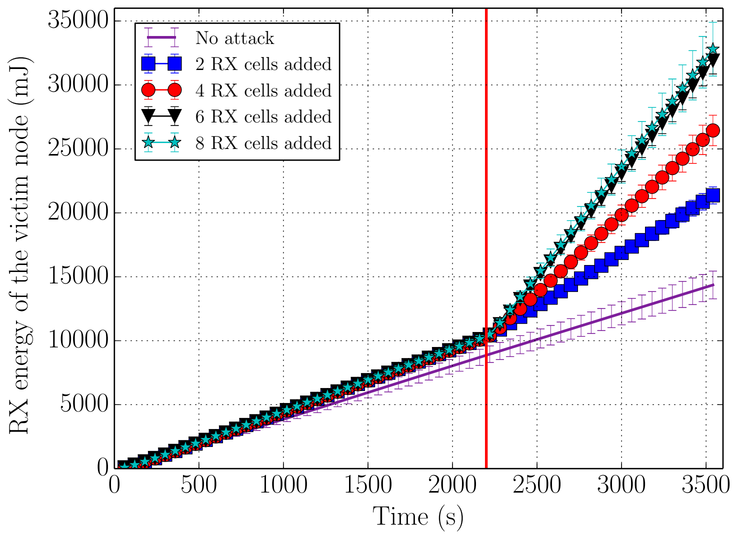


Figure 17: RX energy of the victim node (6) with Overloading attack. Middle area.

may results in using (almost) all the cells that can be allocated to the victim node.

This conclusion is confirmed by Figure 18, which shows the number of RX cells allocated by each node, averaged throughout all the simulation replicas. When 8 unneeded RX cells are added, the victim node (node 6) allocates, on average, 20 RX cells. While this value is below the maximum value, even when taking also into account control cells, we need to consider that this is the average value. In addition, even when the number of cells allocated by the victim node is lower than the maximum value, the allocation of some cells is denied if the cells proposed by the compromised node are already busy.

Figure 19 shows the cell consumption of each node, i.e., the percentage of cells allocated to a node, with respect to the maximum number of cells available (i.e., 29 in our experiments). The cell consumption takes into account both the TX/RX cells allocated for data traffic and the shared cells used for control traffic. Of course, the cell consumption depends on the amount of traffic managed by a node. However, all the nodes have a minimum cell consumption equal to 14%, due to the 4 shared control cells. For the victim node (node 6), the Overloading attack results in a cell consumption equal to 45% and 87% (on average), when 2 or 8 RX cells are added by the compromised node, respectively. Another relevant aspect that emerges from Figure 17 is that, in the presence of an Overloading attack, the energy consumption of the victim node is *always* greater than that in the attack-free case, even before the attack starts. This apparently counter intuitive result can be explained by recalling that a compromised node is not involved in network operation (i.e., it does not cooperate in forwarding messages), and by looking at the number of allocated RX cells (Figure 18) and cell consumption (Figure 19). In the attack-free scenario, node 3 is not compromised, and hence it is involved in the DODAG. Specifically, it can be selected as preferred parent and cooperate in forwarding messages to the root node. This reduces the number of children of node 6, and hence the number of cells allocated by node 6. When node 3 is compromised, it does not join the DODAG, and hence node 6 must allocate more RX cells than before. This justifies its higher RX energy consumption before the attack starts.

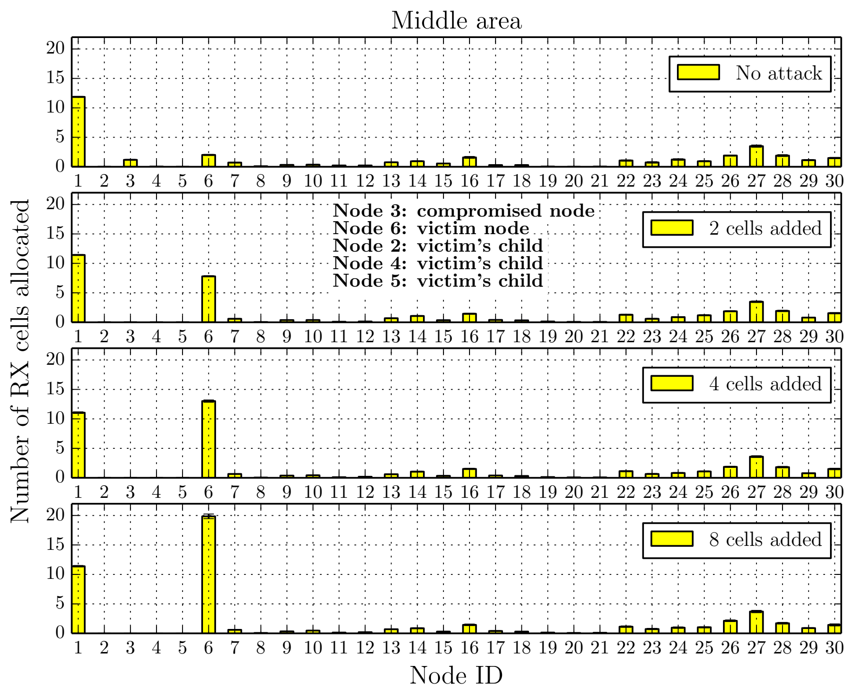


Figure 18: Number of RX cells allocated by all the nodes. Overloading attack.

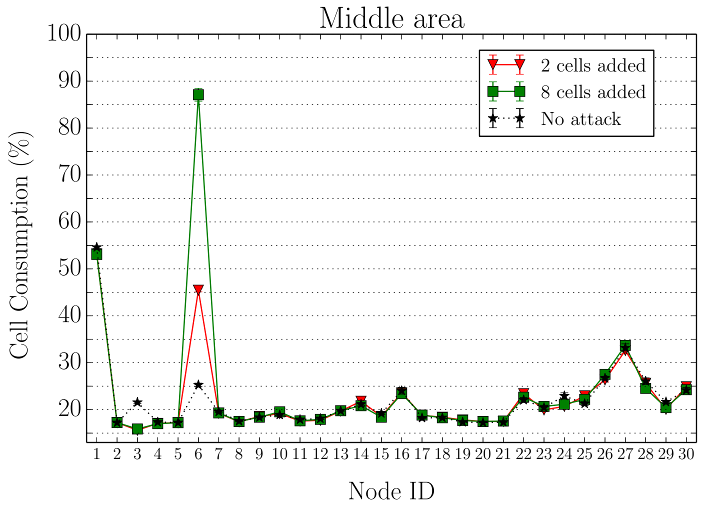


Figure 19: Cell consumption for all the nodes with the Overloading attack.

Figure 20 shows the RX energy consumption increase after 23 minutes from the execution of the Overloading attack both computed analytically and derived from our simulation results. From the graph it can be noticed that the additional energy consumption due to the attack is comparable between the analytical model and the simulations, except for the case when 8 RX cells are added. This can be explained considering that in the analytical model, if a node has 3 child nodes, we consider to add $8 \cdot 3 = 24$ cells, and compute the additional RX energy consumed on those cells. On the other hand, during simulation, many other cells (e.g., for control traffic, parent communication, etc) could be already scheduled by the node before the attack, thus reducing the number of free cells that can be scheduled by the compromised node performing the Overloading attack, also considering that the slotframe length is 29 timeslots, a subset of the 24 cells will be allocated, reducing the increase in the RX energy consumption. We only show the results of the comparison for the middle area, since the results have the same trend for the root area.

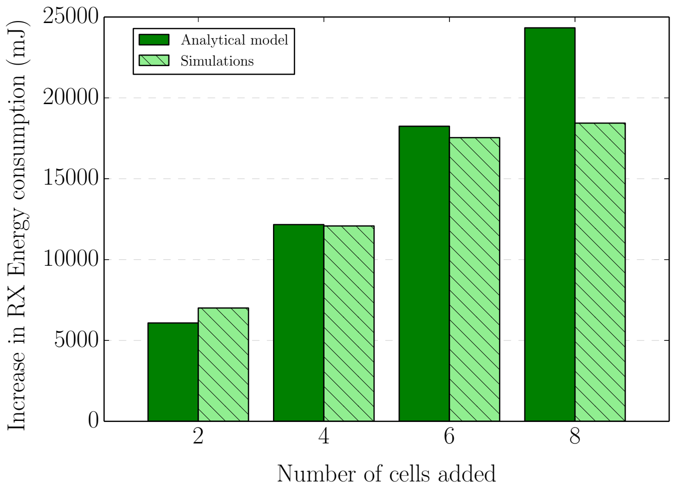


Figure 20: Analytical model and simulation comparison for the Overloading attack - Middle area.

9.3.2. Root area

Figure 21 shows the impact of the Overloading attack when the victim node is positioned in the root area, as per Figure 16b. The attack is effective also in this case. If we compare these results with those obtained for the middle area (see Figure 17), we can observe that the total energy consumption at the end of the simulation experiment is approximately the same. However, in this scenario there is a higher variability, as shown by the larger confidence intervals. As highlighted in the previous sections, this is due to the fact that the root area is characterized by a high node density, and thus the nodes have a high probability of changing their preferred parent during the simulation experiment. Specifically, we observed that the average number of parent changes experienced by a node during the simulation experiment is about 2, if the node is located in the root area, and 1.3 if the node is located in the middle area.

A more detailed breakdown is reported in Figure 22 that shows the number of RX cells and the corresponding energy consumed by the victim node during a simulation run. In the considered experiment, the victim node (node 6) is the preferred parent of two children. Before the execution of the attack, the victim node schedules legitimate RX cells with its children. Then, at time 2200s, the compromised node successfully adds to the victim node 4 RX cells per victim's child. Hence, the RX energy consumed by the victim node starts increasing at a faster pace. However, right before 3000s, one of the two victim's children changes preferred parent, thus freeing all the RX cells that it had scheduled with the victim node (including those added as a result of the attack execution). From that point on, the RX energy consumption grows at a lower pace again, since the victim node is using a lower number of RX cells.

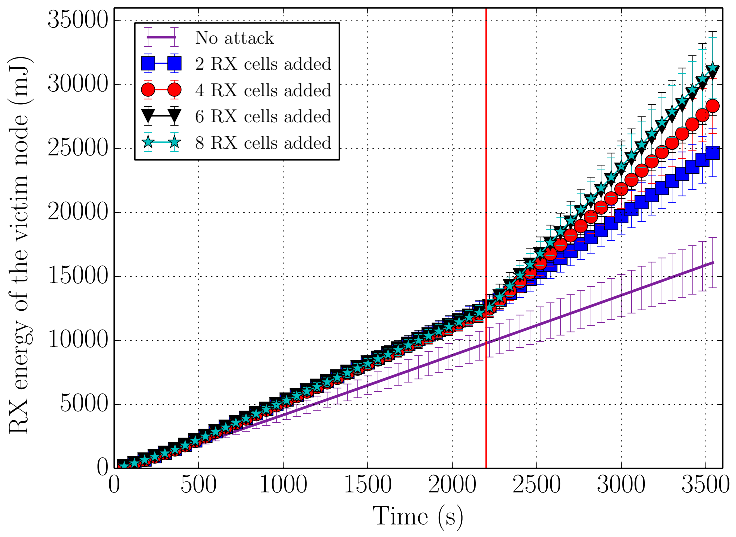


Figure 21: RX energy of the victim node (ID = 6) with Overloading attack. Root area.

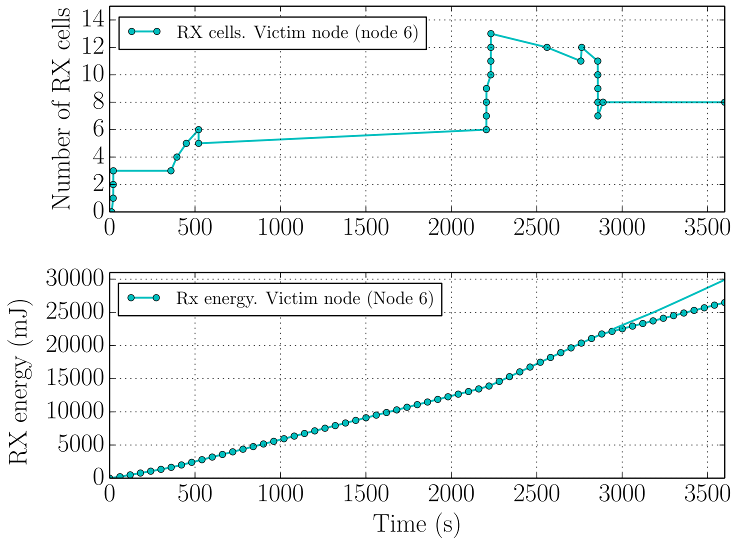


Figure 22: Number of allocated RX cells (top) and RX energy consumption in mJ (bottom) over time of the victim node with Overloading attack. Root area.

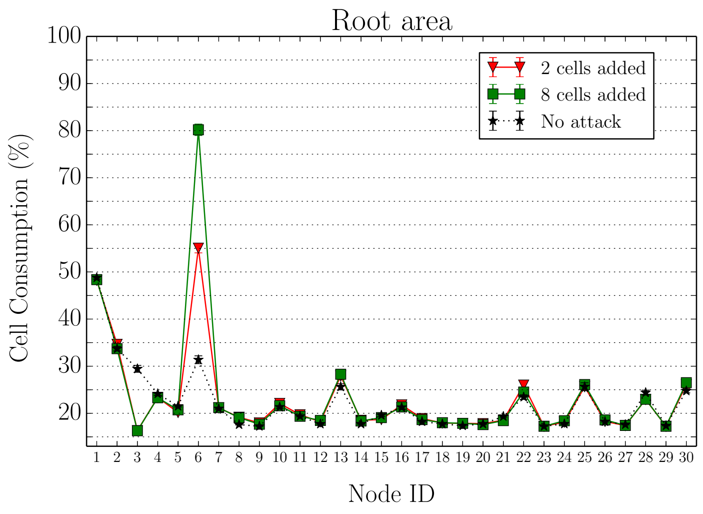


Figure 23: Cell consumption for all the nodes with the Overloading attack.

As shown in Figure 23, the more dynamic behavior of the nodes in the root area also reflects on the cell consumption. Specifically, the cell consumption of the victim node is now 80%, on average, while it was 87% for the victim node located in the middle area (see Figure 19). Since the nodes have more neighbors, there is a wider set of nodes from which they can select the preferred parent. For this reason, node 6 has a lower average number of children in the root area, and hence its energy consumption under attack increases at a lower rate, in comparison with the previous results for the middle area. Specifically, when adding 8 unneeded RX cells, the increase in the cell consumption due to the attack is 49% in the root area and 55% in the middle area.

To conclude the analysis of the Overloading attack, and in order to give an idea of how the increase in the energy consumption of the victim node impacts its lifetime, we consider a device powered by two AA 1.5V batteries, which can provide a total capacity of 25,920 J $(2 * (2.4Ah * 1.5V * 3600s))$. In the attack-free scenario, node 6 consumes 0.004 J per second, thus its lifetime is estimated to be about 75 days. If the compromised node adds 8 RX cells per victim's child, node 6 consumes 0.009 J per second and its lifetime is estimated to be around 33 days. Hence, the battery drains 56% faster than in the attack-free scenario. If the compromised node only adds 2 RX cells per victim's child, node 6 consumes 0.007 J per second and its lifetime is estimated to be of 43 days, thus its battery will drain 43% faster than in the attack free scenario.

10. Experimental Results

Since simulation experiments might not take into account all the contributing factors that can occur in a real environment, we also performed a set of measurements on a real testbed. The purpose of this experimental analysis is twofold: (i) validating, and thus confirming, the previous simulation results, and (ii) evaluating the impact of the two considered attacks in a real environment.

For our experimental measurements, we used the Pisa IoT Testbed (PINT) [25], a multi-purpose platform for validation and performance evaluation of IoT protocols and applications.

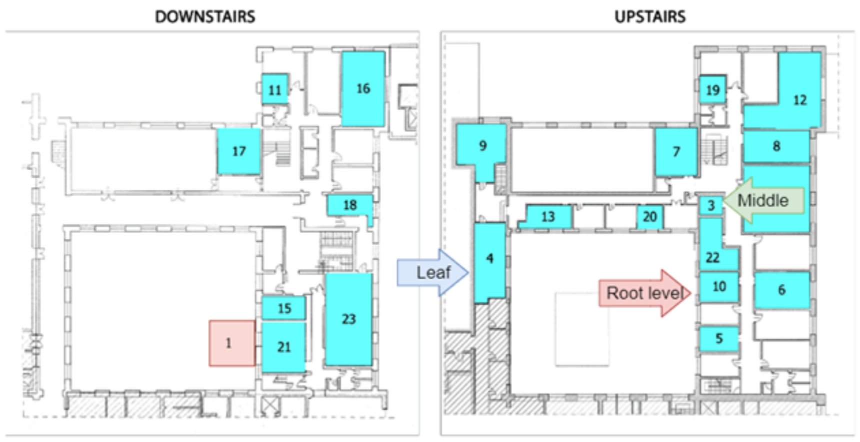


Figure 24: Testbed deployment map with different areas highlighted.

The testbed consists of 23 nodes deployed on a two-floor office environment at the Department of Information Engineering of the University of Pisa. Nodes are installed in different rooms, as shown in Figure 24. Each node is composed of a Zolertia RE-Mote board [26], a rapid prototyping solution for many IoT applications. The boards are equipped with an ARM Cortex-M3 microcontroller, with 512 KB of ROM and 32 KB of RAM, as well as with a radio interface implementing the IEEE 802.15.4 standard at 2.4 GHz and sub 1 GHz.

Each node is connected to a Raspberry Pi embedded system, which is installed to control the execution of the experiments and collect logs from the node. The Raspberry Pi is a popular board widely adopted for IoT rapid prototyping. The system comes with an ARM Cortex-A7 CPU, which can support the execution of different Linux distributions. The USB connection with the Raspberry Pi provides power and serial communication to the Re-Mote. For statistical collection and debug purposes, the output of each node can be collected during the execution of the experiments, through a management service running on the Raspberry Pi that can collect the output of each Re-Mote.

For our experimental analysis, we replicated only a subset of the previous simulation experiments. Specifically, for the Traffic Dispersion attack, we considered one compromised and victim node. Similarly, for the Overloading attack we considered only one victim node with two children; in addition, during the attack only one RX cell was added by the compromised node for each child of the victim node.

In order to increase the accuracy of our experimental results, for each experiment we performed 6 different replicas, each spanning 20 minutes. The results presented below are averaged over all the replicas. In the figures, we also show the 95% confidence intervals.

In the testbed, we considered a movable compromised node, with node ID equal to 2. This allows us to position the compromised node in different areas of the network and evaluate the impact of the attack for different scenarios, as we did in the simulation study in Section 9. As shown in Figure 24, for the Traffic Dispersion attack the victims are node 10 in the root area, node 3 in the middle area, and node 4 in the leaf area. For the Overloading attack, the victim is node 13, located in the middle area.

It is worth highlighting that the RPL DODAG formed by the real nodes varies between different experiment replicas, due to interference and fluctuations in the channel conditions. This

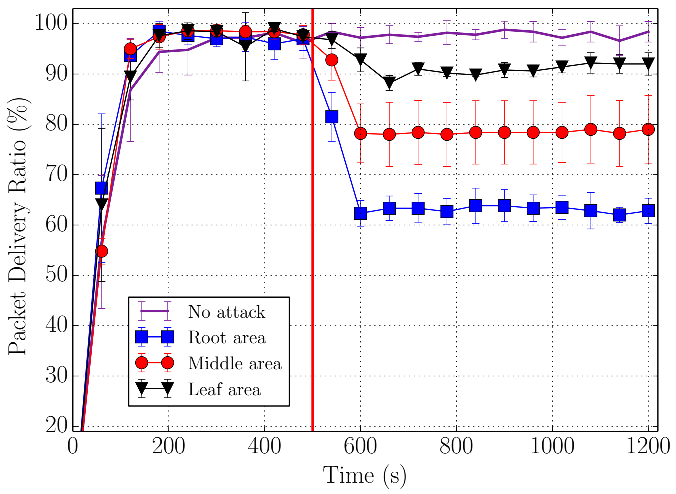


Figure 25: PDR with Traffic Dispersion attack in the PINT testbed.

happens because the experiments were run during working hours, when the building is populated and many devices that generate interference are turned on. Nevertheless, nodes 10, 3, 4 and 13 still end up positioned in the topology areas mentioned above.

Figure 25 shows the impact of the Traffic Dispersion attack, in terms of reduced PDR. As a preliminary remark, we observe that the PDR is lower than 100%, even in the absence of attack. This is due to the unreliability of the wireless medium (in the simulation experiments, we assumed an ideal communication channel to better emphasize the impact of the Traffic Dispersion attack).

The experimental results in Figure 25 confirm those obtained through simulations (see Section 9.2). The impact of the attack greatly depends on the compromised node's and victim node's position in the DODAG. Specifically, after the attack is executed (i.e., at time 500s), the PDR decreases with respect to the attack-free case by 5-10% if the victim node is in the leaf area, by approximately 20% if it is in the middle area, and by about 40% if it is in the root area.

As already highlighted, the RPL DODAG in a real testbed is subject to frequent changes, even during a single experiment. Therefore, in order to obtain consistent results between different replicas of the same experiment, the Overloading attack was performed on a victim node, i.e., node 13, that always has two children, i.e., nodes 4 and 9. In addition, the attack consists in adding only two unneeded RX cells to the victim node's schedule, i.e., one per victim's child node.

Figure 26 shows the overall RX energy consumed by the victim node over time, with and without attack. When the attack is executed (i.e., at time 500s), two additional RX cells are added to the victim node's schedule. As expected, this yields an increase in the growth rate of the energy consumption. The total RX energy consumed by the radio chip at the end of the experiment is 4.8 J, i.e., 20% more than in the attack-free case.

11. Attack mitigation

In this section, we discuss some possible actions than can be implemented to mitigate the impact of the Traffic Dispersion and Overloading attacks on availability. For instance, if a victim

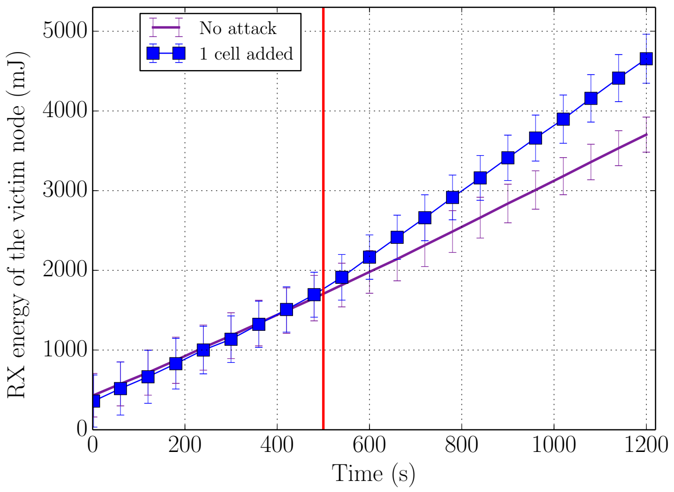


Figure 26: RX energy of the victim node with Overloading attack in the PINT testbed.

pair could timely detect an anomaly in its network activities, it would be possible to execute specific actions as attack mitigation.

Unfortunately, a victim pair cannot notice anomalies due to a Traffic Dispersion attack. This is because, during and after the attack execution, neither the compromised node performing the attack displays any suspicious behavior nor the victim node perceives any anomalies in its network activities. Therefore, the attack itself does not yield any suspicious deviation in the regular network operations performed by the victim node. It is worth recalling that, when performing the Traffic Dispersion attack, the compromised node deletes/relocates all (or part of) the RX cells that the victim node has scheduled with its preferred parent. Then, the compromised node adds the deleted/relocated cells to its own schedule so that it can acknowledge data packets sent by the victim node (see Section 6), hence the victim node not being able to notice any anomaly thereafter. On the other hand, the preferred parent of the victim node can fairly assume that those RX cells were deleted due to the victim node experiencing lower traffic condition, or having selected a new preferred parent (in case of complete cancellation). Since those events are totally possible to occur, they do not generate any suspect on either node of the victim pair.

Instead, a victim pair can notice anomalies due to an Overloading attack. This can rely on a node-based, local monitoring process, which can be incorporated in the used SF at each network node. Specifically, on each network node, the SF used to trigger the allocation/deallocation of cells can locally monitor whether the (new added) RX cells are or are not actually used to receive data packets. Even assuming some tolerance in the short term, not receiving data packets at a given cell for some time can be assumed as a symptom of a schedule anomaly yielding inefficiency. In fact, the allocation of unused resources should be released also in order to save energy.

In order to implement this mitigation approach, we have modified the MSF by introducing an additional check on the utilization of RX cells. Of course, this check can be implemented on any SF. The modified version of MSF, hereafter referred to as MSF-M, is shown in Algorithm 2, where the differences with respect to the original MSF are highlighted in blue. Each node periodically checks the utilization of each of its allocated RX cells. That is, it checks how many

Algorithm 2: MSF-M Algorithm

```
Input: $NCE$ = Number of allocated cells elapsed from the last time MSF was run
       $MAX\_NUMCELLS$ = Number of allocated cells elapsed to trigger next MSF run
       $NCU_{tx}$ = Number of allocated cells used for transmission
       $LIM\_NUMCELLSUSED\_HIGH$ = Threshold to add a transmission cell
       $LIM\_NUMCELLSUSED\_LOW$ = Threshold to delete a transmission cell
       $RXU[N]$ = Utilization values of the N allocated reception cells
       $MIN\_RX\_UTILIZATION$ = Threshold to remove unused reception cells
Output: ADD/DEL one transmission cell
        DEL unused reception cells
1  while allocated cell then{
2      if $NCE > MAX\_NUMCELLS$ then{
3          if $NCU_{tx} > LIM\_NUMCELLSUSED\_HIGH$ then{
4              trigger 6P to ADD one cell }
5          if $NCU_{tx} < LIM\_NUMCELLSUSED\_LOW$ then{
6              trigger 6P to DEL one cell }
7          $NCE = 0$
8          for each reception cell i do {
9              if $RXU[i] < MIN\_RX\_UTILIZATION$ for more than two monitoring periods
10                 DEL reception cell i}
11     }else{ $NCE = NCE + 1$ }
12 }
```

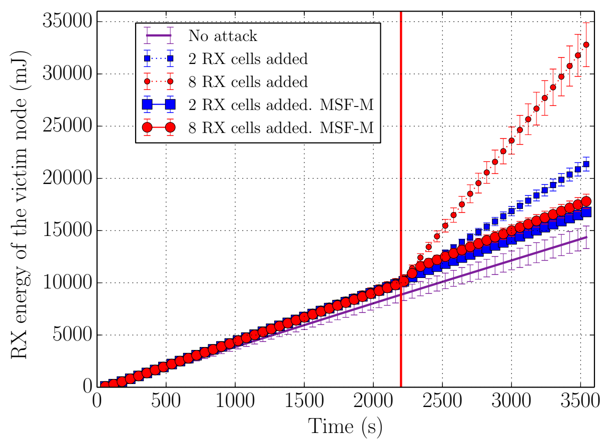
data packets it has received for each of its allocated RX cells, during the last monitoring period. If, for two consecutive monitoring periods, the utilization of an RX cell is below a pre-defined threshold (e.g., 10%), the node deletes that specific cell from its schedule.

We have configured the monitoring process to wait for two consecutive periods, in order to avoid deleting newly and legitimately scheduled RX cells, such as those that are scheduled right before the start of the monitoring period. The choice of an appropriate duration for the monitoring period is a trade-off between, on one hand, the risk of deleting (to-be) used cells and, on the other hand, the futile energy consumption experienced until unused RX cells are eventually deleted from the current schedule.

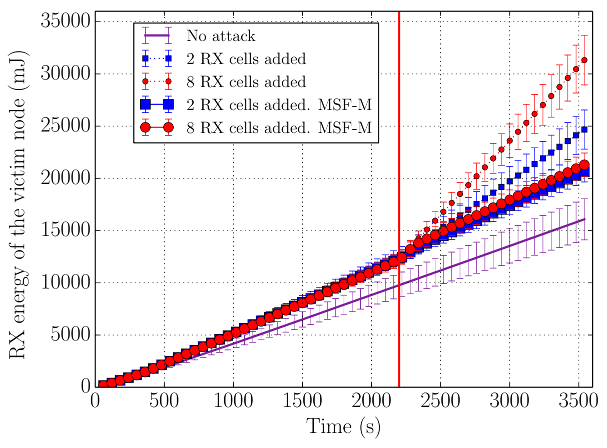
Note that we cannot rely on a 6P DELETE transaction to remove the unused RX cells, because the transaction would terminate with an error. Specifically, when performing a 6P DELETE transaction, the cells that need to be removed must be already scheduled between the two nodes, otherwise the node that receives the 6P DELETE request replies with an error and no cells will be effectively freed up.

11.1. Results

In the following, we discuss the effectiveness of the MSF-M in mitigating the impact of Overloading attacks on the energy consumption of the victim node and the availability of the network as a whole. Figure 27a and Figure 27b show the RX energy consumed by the victim node, when it is positioned in the middle area and in the root area, respectively. When using MSF-M, the additional energy consumption due to the Overloading attack is dramatically reduced, as



(a) Middle area.



(b) Root area.

Figure 27: RX energy of the victim node with Overloading attack and attack mitigation.

the anomaly due to the attack is detected very early by the victim node. For the same reason, the impact of the attack with 2 or 8 added cells is not so noticeable as before. With respect to the case without mitigation, the reduction of the energy consumption in the middle area is 21% when 2 cells are added, and 46% when 8 cells are added. Instead, in the root area, the reduction is of 16% when 2 cells are added, and 32% when 8 cells are added.

Of course, the victim node still consumes more RX energy than in the attack-free case. This is because the MSF-M takes some time to detect the anomaly due to the attack, while in the meantime the victim node still switches its radio interface to RX during the unneeded RX cells added by the compromised node. However, these results show that the MSF-M effectively and considerably mitigates the impact of the Overloading attack.

We would like to point out that the mitigation presented above is *not* a security solution that prevents the Overloading attack altogether. That is, while it substantially reduces the impact of an Overloading attack on availability, it does not make the attack intrinsically harder to perform or less severe. In other words, while this mitigation is clearly beneficial in counteracting the effects of the Overloading attack on the victim nodes' energy consumption and on network availability, it does not play any direct role on the threat model or in terms of security enforcement, neither for better or for worse.

At the same time, due to its effectiveness, the proposed mitigation should be considered as a good practice to adopt for managing network resources, and specifically cells in 6TiSCH schedules, also in an attack-free scenario, as generally resulting in reducing the long-term allocation of lengthily unused cells. These indeed include the cells allocated as a result of an executed Overloading attack, with consequent reduction of its impact on the energy consumption of the victim nodes in particular, and on the availability in the network as a whole.

12. Conclusion

In this article, two security vulnerabilities of the 6P protocol are presented. These two attacks are capable to easily and stealthily alter the communication schedule of the victim nodes, resulting in a reduced reliability of data transmission throughout the network or an increased energy consumption on the victim nodes. The potential impact of the attacks has been assessed initially via an analytical model and subsequently via an extended performance evaluation both via simulation and real experiments.

Our results have shown that the considered attacks can significantly impair network performance and threaten network basic functionalities and efficiency, by specifically impacting network availability and energy consumption of victim nodes. Moreover, the position of the victim nodes within the network greatly affects the magnitude of the attack impact. That is, when performing the Traffic Dispersion attack against three victim nodes, the overall reduction of the Packet Delivery Ratio ranges from 12%, when the victim nodes are positioned in the peripheral area of the network, to 65% when they are close to the root node. Instead, when performing the Overloading attack, the energy consumption of the victim node may increase of 55% and 49%, when that node is positioned in the middle area or in the root area of the network, respectively.

Our analysis has also pointed out that the impact of the considered attacks can be magnified or reduced by the dynamics of the network introduced by the RPL protocol. For instance, if the victim node changes its preferred parent after the attack execution, the attack becomes ineffective from then on. In order to alleviate the impact of the considered attacks, we have investigated possible mitigation actions. While no mitigation is possible against the Traffic Dispersion attack, we have modified the Minimal Scheduling Function (MSF) to include a node-based, local monitoring mechanism for early detection of schedule anomalies possibly due to an Overloading attacks. The modified MSF has shown itself to be indeed effective in considerably alleviating the impact of the Overloading attack.

Acknowledgments

The authors sincerely thank the anonymous reviewers and the Associate Editor for their insightful comments and suggestions, that have helped to improve the technical and editorial quality of the manuscript. The authors would also like to thank Gioele Carignani for his invaluable help in the implementation of the attacks on the Contiki-NG OS. This work was partially supported by the Italian Ministry of Education and Research (MIUR) in the framework of the CrossLab project (Departments of Excellence); by VINNOVA and the CelticNext project CRITISEC; by the H2020 project SIFIS-Home (Grant agreement 952652); and by the SSF project SEC4Factory (grant RIT17-0032).

References

[1] P. Thubert, An Architecture for IPv6 over the Time-Slotted Channel Hopping Mode of IEEE 802.15.4 (6TiSCH), RFC 9030 (Informational) (May 2021).

[2] T. Chang, M. Vučinić, X. Vilajosana, S. Duquennoy, D. R. Dujovne, 6TiSCH Minimal Scheduling Function (MSF), RFC 9033 (Proposed Standard) (May 2021).

[3] M. Vučinić, J. Simon, K. Pister and M. Richardson, Constrained Join Protocol (CoJP) for 6TiSCH, RFC 9031 (Proposed Standard) (Month 2021).

[4] NIST, Standards for Security Categorization of Federal Information and Information Systems. Technical Report FIPS-199 (February 2004).

[5] G. Carignani, F. Righetti, C. Vallati, M. Tiloca, G. Anastasi, Evaluation of Feasibility and Impact of Attacks Against the 6top Protocol in 6TiSCH Networks, in: 21st International Symposium on ”A World of Wireless, Mobile and Multimedia Networks” (WoWMoM), IEEE, New York (NY, USA), 2020, pp. 68–77.

[6] Q. Wang, X. Vilajosana, T. Watteyne, 6TiSCH Operation Sublayer (6top) Protocol (6P), RFC 8480 (Proposed Standard) (November 2018).

[7] G. Montenegro, N. Kushalnagar, J. Hui, D. Culler, Transmission of IPv6 Packets over IEEE 802.15.4 Networks, RFC 4944 (Proposed Standard) (September 2007).

[8] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, R. Alexander, RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks, RFC 6550 (Proposed Standard) (March 2012).

[9] Z. Shelby, K. Hartke, C. Bormann, The Constrained Application Protocol (CoAP), RFC 7252 (Proposed Standard) (June 2014).

[10] IEEE, IEEE 802.15.4-2020 - IEEE Standard for Low-Rate Wireless Networks, IEEE, New York (NY, USA) (July 2020).

[11] O. Gnawali, P. Levis, The Minimum Rank with Hysteresis Objective Function, RFC 6719 (Proposed Standard) (September 2012).

[12] D. S. J. De Couto, D. Aguayo, J. Bicket, R. Morris, A High-Throughput Path Metric for Multi-Hop Wireless Routing, in: 9th Annual International Conference on Mobile Computing and Networking, MobiCom ’03, ACM, New York (NY, USA), 2003, p. 419–434.

[13] G. Selander, J. Mattsson, F. Palombini, L. Seitz, Object Security for Constrained RESTful Environments (OSCORE), RFC 8613 (Proposed Standard) (July 2019).

[14] M. Gunnarsson, J. Brorsson, F. Palombini, L. Seitz, M. Tiloca, Evaluating the performance of the OSCORE security protocol in constrained IoT environments, Internet of Things 13 (2021) 1–16.

[15] S. Kim, H.-S. Kim, C. Kim, ALICE: Autonomous Link-Based Cell Scheduling for TSCH, in: 18th International Conference on Information Processing in Sensor Networks, ACM, New York (NY, USA), 2019, pp. 121–132.

[16] F. Righetti, C. Vallati, S. K. Das, G. Anastasi, An Evaluation of the 6TiSCH Distributed Resource Management Mode, ACM Transactions on Internet of Things 1 (4) (2020) 1–31.

[17] Y. Boufenneche, R. Zitouni, L. George, N. Gharbi, Selfishness in secure internet of things networks: 6TiSCH case study, Wireless Networks 27 (6) (2021) 3927–3946.

[18] C. Vallati, S. Brienza, G. Anastasi, S. K. Das, Improving Network Formation in 6TiSCH Networks, IEEE Transactions on Mobile Computing 18 (1) (2019) 98–110.

[19] D. Fanucchi, B. Staehle, R. Knorr, Network Formation for Industrial IoT: Evaluation, Limits and Recommendations, in: 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), IEEE, New York (NY, USA), 2018, pp. 227–234.

[20] D. Fanucchi, F. Righetti, C. Vallati, B. Staehle, G. Anastasi, Improving Link Quality Estimation Accuracy in 6TiSCH Networks, in: 6th International Conference on Internet of Things: Systems, Management and Security (IOTSMS), IEEE, New York (NY, USA), 2019, pp. 243–250.

[21] A. Kalita, A. Brighente, M. Khatua, M. Conti, Effect of DIS Attack on 6TiSCH Network Formation, IEEE Communications Letters 26 (5) (2022) 1190–1193.

[22] A. Althubaity, T. Gong, K.-K. Raymond, M. Nixon, R. Ammar, S. Han, Specification-based Distributed Detection of Rank-related Attacks in RPL-based Resource-Constrained Real-Time Wireless Networks, in: 2020 IEEE Conference on Industrial Cyberphysical Systems (ICPS), Vol. 1, IEEE, New York (NY, USA), 2020, pp. 168–175.

[23] W. Yang and Q. Wang and Y. Wan and J. He, Security Vulnerabilities and Countermeasures for Time Synchronization in IEEE 802.15.4e Networks, in: 3rd International Conference on Cyber Security and Cloud Computing (CSCloud), IEEE, New York (NY, USA), 2016, pp. 102–107.

[24] M. Tiloca, D. D. Guglielmo, G. Dini, G. Anastasi, S. K. Das, DISH: DIstributed SHuffling Against Selective Jamming Attack in IEEE 802.15.4e TSCH Networks, ACM Transactions on Sensor Networks 15 (1) (2019) 1–28.

[25] C. Vallati, E. Ancillotti, R. Bruno, E. Mingozzi, G. Anastasi, Interplay of Link Quality Estimation and RPL Performance: An Experimental Study, in: 13th Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks, PE-WASUN '16, ACM, New York (NY, USA), 2016, p. 83–90.

[26] Zolertia Re-Mote, `https://github.com/Zolertia/Resources/wiki/RE-Mote`, Accessed: 2022-05-13 (November 2016).

[27] F. Osterlind and A. Dunkels and J. Eriksson and N. Finne and T. Voigt, Cross-Level Sensor Network Simulation with COOJA, in: 31st IEEE Conference on Local Computer Networks, IEEE, New York (NY, USA), 2006, pp. 641–648.

[28] O. Durmaz Incel, A. Ghosh, B. Krishnamachari, K. Chintalapudi, Fast Data Collection in Tree-Based Wireless Sensor Networks, IEEE Transactions on Mobile Computing 11 (1) (2012) 86–99.

[29] A. Dunkels, F. Osterlind, N. Tsiftes, Z. He, Software-Based on-Line Energy Estimation for Sensor Nodes, in: 4th Workshop on Embedded Networked Sensors, ACM, New York (NY, USA), 2007, p. 28–32.

[30] X. Fafoutis, A. Elsts, A. Vafeas, G. Oikonomou, R. Piechocki, On Predicting the Battery Lifetime of IoT Devices: Experiences from the SPHERE Deployments, in: 7th International Workshop on Real-World Embedded Wireless Systems and Networks, ACM, New York (NY, USA), 2018, p. 7–12.

[31] F. Righetti, C. Vallati, G. Anastasi, S. Das, Performance Evaluation the 6top Protocol and Analysis of its Interplay with Routing, in: 2017 IEEE International Conference on Smart Computing (SMARTCOMP), 2017, pp. 1–6.

Appendix A. Symbols and Abbreviations

In the following, Table A.1 and Table A.2 provide a list of the symbols and abbreviations used throughout this article, respectively.

Table A.1: List of symbols

Symbol	**Description**
C	Set of child nodes
E	Set of links between nodes
E_B	Extra RX energy consumption experienced by a node B
E_{cell}	RX energy consumption for a RX cell
F	Frequency used during a TSCH timeslot
G	Oriented Graph
N	Number of nodes in the network
N_c	Number of child nodes
$N_{\mathcal{S}}$	Number of cells scheduled
$N'_{\mathcal{S}}$	Number of respectively allocated cells between a couple of node in a child-parent relationship
n_i	i^{th} node in the network
NPP	Node's Preferred Parent
p_i	Parent of the i^{th} node
Φ	Set of traffic flows
$Plink$	Link loss probability due to the channel quality
$\mathcal{S}$	Original 6TiSCH schedule shared between two network nodes
$\mathcal{S}_A$	Original 6TiSCH schedule installed on a network node A
$\mathcal{S}'$	Alternative 6TiSCH schedule installed on a victim node by the adversary
$\mathcal{S}'_B$	Alternative 6TiSCH schedule installed on a victim node B by the adversary
SeqNum	Sequence Number used by two 6TiSCH neighbors for their 6P transactions
T	Identifier of a TSCH timeslot
T_{RX}	Time that a node spends in RX state
V	Set of nodes
W_{RX}	Power consumption of a node in RX state

Table A.2: List of abbreviations

Abbreviation	**Description**
6P	6top Protocol
6TiSCH	IPv6 over the Time-Slotted Channel Hopping Mode of IEEE 802.15.4
AES	Advanced Encryption Standard
ASN	Absolute Slotframe Number
CCM	Counter Mode with Cipher Block Chaining (CBC) Message Authentication Code (MAC)
CoAP	Constrained Application Protocol
CoJP	Constrained Join Protocol
DIS	DODAG Information Solicitation
DISH	DIstributed SHuffling
DODAG	Destination Oriented Directed Acyclic Graph
EB	Enhanced Beacon
ETX	Expected Number of Transmissions
FIPS	Federal Information Processing Standards
FORCE	Forged rank and routing metric detector
IE	Information Element
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IoT	Internet of Things
IIoT	Industrial Internet of Things
IP	Internet Protocol
JP	Join Proxy
JRC	Join Registrar/Coordinator
MAC	Medium Access Control
MIC	Message Integrity Code
MSF	Minimal Scheduling Function
MSF-M	Minimal Scheduling Function with Mitigation
NIST	National Institute of Standards and Technology
OF	Objective Function
OSCORE	Object Security for Constrained RESTful Environments
PDP	Packet Delivery Probability
PDR	Packet Delivery Ratio
PINT	Pisa IoT Testbed
PSK	Pre-Shared Key
QoS	Quality of Service
RAM	Random Access Memory
ROM	Read Only Memory
REST	REpresentational State Transfer
RPL	Routing Protocol for Low-Power and Lossy Networks
RX	Reception Mode

Abbreviation	**Description**
SF	Scheduling Function
TSCH	Time Slotted Channel Hopping
TX	Transmission Mode
UDGM	Unit Disk Graph Medium
UDP	User Datagram Protocol
USB	Universal Serial Bus