Evaluation of Feasibility and Impact of Attacks Against the 6top Protocol in 6TiSCH Networks

Gioele Carignani*, Francesca Righetti*, Carlo Vallati*, Marco Tiloca† and Giuseppe Anastasi *
*Department of Information Engineering, University of Pisa, Pisa, Italy
Email: gioelecarignani@gmail.com, {francesca.righetti, carlo.vallati, giuseppe.anastasi}@ing.unipi.it
†RISE Cybersecurity, RISE Research Institutes of Sweden, Kista, Sweden
Email:marco.tiloca@ri.se

***Abstract*—The 6TiSCH architecture have been gaining attraction as a promising solution to ensure reliability and security for communication in applications for the Industrial Internet of Things (IIoT). While many different aspects of the architecture have been investigated in literature, an in-depth analysis of the security features included in its design is still missing. In this paper, we assess the security vulnerabilities of the 6top protocol, a core component of the 6TiSCH architecture for enabling network nodes to negotiate communication resources. Our analysis highlights two possible attacks against the 6top protocol that can impair network performance and reliability in a significant manner. To prove the feasibility of the attacks in practice, we implemented both of them on the Contiki-NG Operating System and tested their effectiveness on a simple deployment with three Zolertia RE-Mote sensor nodes. Also, we carried out a set of simulations using Cooja in order to assess their impact on larger networks. Our results show that both attacks reduce reliability in the overall network and increase energy consumption of the network nodes.**

***Index Terms*—Security, Industrial Internet of Things, 6TiSCH, 6top, IEEE 802.15.4.**

I. Introduction

Recent advancements in communication technologies have allowed the adoption of wireless solutions also in industrial deployments. Such solutions, are expected to play a crucial role as enabling technology to ensure rapid and flexible deployment in the context of the Industrial Internet of Things (IIoT). At the same time, industrial applications require timely and reliable communication to safely implement critical functionalities such as industrial control and telemetry.

To this end, the "IPv6 over the TSCH mode of IEEE 802.15.4e" - 6TiSCH network architecture [1] has been gaining attention as a solution that can fulfil such requirements and ease the integration of existing IPv6 networks into sensor and actuator networks in industrial systems. The architecture is based on the Time Slotted Channel Hopping (TSCH) mode for the Medium Access Control (MAC) protocol of the standard IEEE 802.15.4 [2]. In particular, TSCH ensures timely data transmission by framing time into deterministic periodic slotframes composed of fixed timeslots, as well as multi-channel communication and resiliency to interference by leveraging multiple frequencies and channel hopping, respectively.

Reliable and timely communication alone is not sufficient to guarantee resiliency of industrial applications. That is, communication security is also required, to ensure confidentiality, integrity and freshness of messages exchanged within the network. For this reason, the 6TiSCH architecture, currently under standardisation within the 6TiSCH Working Group (WG) at the IETF, includes by design a set of security features. In particular, the architecture defines a minimal security framework [3], which ensures secure network joining, contextually with provisioning of key material. Also, it relies on the link-layer security services of the IEEE 802.15.4 standard [4], which provides secure communication through symmetric keys. In practical settings, all network nodes typically use a same shared network key received upon joining the network.

In this paper, we assess some security vulnerabilities of the 6top protocol (6P), a core component of the 6TiSCH architecture which enables network nodes to negotiate communication resources and to agree on a common schedule. Our analysis highlights two possible attacks against 6P, namely the *Traffic Dispersion* attack and the *Overloading* attack, which can significantly impair network performance and reliability.

The rationale of both attacks is to alter the communication schedule adopted by the victim node, through bogus 6P transactions. In particular, both attacks result in a communication schedule that differs between the victim node and its neighbours. As ultimate goal, the Traffic Dispersion attack aims at disrupting the message transmission of the victim node, while the Overloading attack aims at allocating unneeded communication resources and increasing the energy consumption on the victim node.

Even with security at the link-layer, the attacks are possible in practical settings due to the usage of a single network key, shared among all the network nodes as pre-configured or provided upon joining the network. It follows that any node in the network is able to inject or alter a valid 6P message as related to the 6P transaction of two other different nodes. That is, while a 6P negotiation is intended to concern a pair of nodes, it becomes practically exposed to passive monitoring and active manipulation from any other network node.

To demonstrate the feasibility of the attacks in practice, we implemented them on the popular Contiki-NG Operating System (OS) for sensor nodes. We used our implementation to carry out an experimental evaluation of the attacks impact, where the attacks were mounted against real Zolertia RE-Mote sensor nodes. In particular, we proved that both attacks are feasible to perform and effective in achieving their goals.

Furthermore, to assess the attacks impact on larger networks, we carried out a set of simulation experiments on multi-hop topologies by using Cooja. Our results show that both attacks reduce network reliability as a whole and increase energy consumption of network nodes. To the best of our knowledge, no previous work has analysed the security of 6P.

The rest of the paper is organised as follows. In Section III, we introduce the technical background. In Section II we present the related work. In Section IV, we introduce our threat model. In Sections V and VI, we describe the general attack scheme and the practical execution of two specific attacks, respectively. In Section VII, we present our experimental and simulation results, and discuss our results. Finally, in Section VIII we draw our conclusions.

II. Related work

Although many works focus on different aspects of the 6TiSCH architecture, such as resource allocation [5] [6], only a few have focused on the analysis of its security features to highlight potential vulnerabilities. Specifically, they focus on two main security aspects: (i) secure scheduling with respect to selective jamming attacks and (ii) secure time synchronisation.

For instance, in [7] the authors analyse how an external jammer can efficiently and effectively jeopardise a node's communication pattern, by selectively jamming its transmissions and receptions. This is possible by exploiting the periodicity of the channel hopping sequence, and thus getting knowledge of the victim's full schedule. In order to neutralise 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.

Instead, [8] proposes two types of attacks against the time synchronisation of TSCH. That is, the first one is based on the tampering of control messages to induce victim nodes to synchronise with a malicious node, and thus to adopt a different, bogus channel hopping sequence. Instead, the second one exploits the workflow of the routing protocol RPL: by forging routing protocol control messages, the adversary is able to manipulate the topology of the network. The authors propose a set of countermeasures for both the attacks.

III. Technical background

This section overviews the main technical concepts and building blocks referred throughout the paper.

A. TSCH mode of IEEE 802.15.4

Time Slotted Channel Hopping (TSCH) is a mode for the Medium Access Control (MAC) protocol of the standard IEEE 802.15.4 [2]. The main goal of TSCH-based networks is achieving high communication reliability with low power consumption. To this end, TSCH adopts a time slotted access mode. In particular, time is split into equally sized timeslots of fixed length, grouped into periodic slotframes. In order to increase the network capacity, TSCH allows different nodes to concurrently transmit on the same timeslot using different channel offsets (multi-channel communication). Moreover, TSCH exploits the channel hopping technique to be resilient to interference and multi-path fading, by changing the transmission frequency every timeslot.

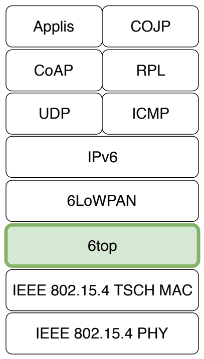

Fig. 1: 6TiSCH Stack.

During a timeslot, a node possibly transmits/receives over one of the available radio frequencies, computed as a function of the Absolute Slotframe Number (ASN) and of a channelOffset value assigned to that node for that timeslot. A *schedule* defines what pairs (timeslot, channel offset) have to be used by a given network node. Such pairs are referred to as *cells*, which are the practical communication resource in the network. The specific building of such schedule is out of the scope of TSCH.

B. 6TiSCH architecture

TSCH targets supervisory control and data acquisition systems based on resource-constrained devices [4], which rely more and more on cloud-based solutions and thus would take great advantage of a standardised integration with the Internet. To facilitate this integration, the IETF has worked on "IPv6 over TSCH mode of IEEE 802.15.4", namely the 6TiSCH architecture [1]. The goal of the 6TiSCH architecture is to enable the Industrial Internet of Things paradigm, by achieving wire-like reliability and low power consumption, which are both already provided by the TSCH mode, together with easy interoperability and integration with the Internet offered by the IPv6 protocol [9].

The 6TiSCH protocol stack is shown in Fig. 1. The Constrained Application Protocol (CoAP) [10] enables lightweight RESTful interactions among network nodes. Routing Protocol for Low-Power and Lossy Networks (RPL) [11] is the routing protocol adopted to ensure multi-hop data delivery. IPv6 over Low-Power Wireless Area Network (6LoWPAN) provides integration and header compression of IPv6 messages over IEEE 802.15.4. The 6TiSCH Operation sublayer (6top) is a sublayer of Logical Link Control (LLC), that provides an abstraction of an IP link over the TSCH-based MAC layer, in order to allow the transmission of IPv6 packets.

In order to dynamically build a suitable schedule, 6TiSCH nodes can run a dedicated *Scheduling Function (SF)*, for instance the 6TiSCH Minimal Scheduling Function (MSF)

[12]. The SF monitors the traffic and determines how many and which cells are needed for the node to correctly operate. Based on such indication from the SF, two nodes can dynamically negotiate a pairwise schedule through the 6top layer.

C. RPL protocol

RPL [11] was designed to implement routing functionalities following the requirements introduced above, by building a Destination Oriented Acyclic Graph (DODAG). It is based on a distance vector that targets collection-based networks, where data are periodically sent by nodes to a central entity known as root node, i.e., the network node where the routing topology described by the DODAG converges to. Each node in the DODAG collects information about its neighbours and select one as preferred parent, as the neighbour to exploit for upward data forwarding.

In order to select the most convenient neighbour, each node has a *rank*, i.e., a number that summarises possible routing metrics associated to that node (e.g., battery level, transmission latency, number of hops to the root node, link quality, etc). An Objective Function (OF) defines the rules to compute a rank value from these metrics and the policy for the selection of the preferred parent.

D. 6top protocol

The 6top protocol (6P) [13] allows neighbours to agree upon a certain schedule by negotiating (adding or deleting) or relocating TSCH cells. The SF triggers 6P transactions with other network nodes, in order to agree on updated pairwise communication schedules.

There are seven types of 6P transactions, namely ADD, DELETE, RELOCATE, COUNT, LIST, SIGNAL and CLEAR. More specifically, a transaction is a complete negotiation between two neighbours that are running the same SF. A transaction can be composed of 2 or 3 steps. In a 2-step transaction, composed by a request message and a response message, the node that sends the request selects the candidate cells to be added, deleted or relocated. Fig. 2 shows an example of a 2-step ADD transaction. Instead, in a 3-step transaction, there is an additional message that concludes the negotiation, i.e., the confirmation message. In this case, the node receiving the request proposes the candidate cells to the transaction initiator, that terminates the negotiation with the confirmation message.

It is important that any two nodes keep their schedule updated and consistent, to avoid misalignment and inconsistencies that would also result in wasting energy and bandwidth. To this end, each pair of neighbour nodes maintains a sequence number (i.e., SeqNum), to ensure that request, response and confirmation messages match with one another, during and throughout 6P transactions. In particular, SeqNum is used to (i) detect and handle duplicate messages, and (ii) detect possible inconsistencies in the schedules.

6P embeds its messages in the Information Elements (IE) [2] of data frames, for which it relies on the security services provided by the link layer and described in Section III-E.

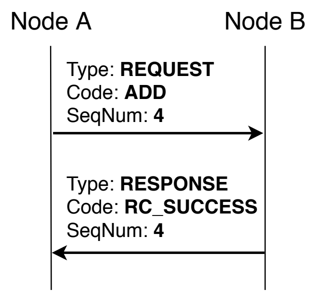

Fig. 2: Successful 6P transaction.

E. 6TiSCH minimal security

In order to securely join a 6TiSCH network, a node performs the steps defined in the 6TiSCH Minimal Security framework [3], as part of the Constrained Join Protocol (CoJP). In particular, three entities are involved: (i) the Join Registrar/Coordinator (JRC); (ii) the pledge, i.e. the new node that wants to join the network; and (iii) the Join Proxy (JP), acting as intermediary between the pledge and the JRC. CoJP is based on the assumption that the JRC and the Pledge pre-share a cryptographic key (PSK) used to mutually authenticate.

Link-layer frames in the 6TiSCH network must be protected, by using the link-layer security services of IEEE 802.15.4. This always provides authentication and integrity protection, optionally together with additional encryption. If encryption is also provided, frames are protected by means of the AES symmetric cipher in CCM* (Counter with CBC-MAC) mode. The encryption key is provided by the JRC upon joining, while the cipher nonce is formed by the ASN and the node identifier.

During the joining process, all CoJP messages exchanged between the pledge and the JRC, possibly through the JP, are not protected at the link layer, since the pledge does not have any knowledge of the key material used in the network yet. Instead, the pledge and the JRC use their shared PSK as Master Secret, in order to derive a security context and secure their CoJP exchanges end-to-end at the application level.

With the except of specific control frames, the messages exchanged in the network are typically encrypted and integrity protected. In either case, message protection at the link-layer is performed by using the key material received from the JRC upon joining, which is used for encrypting, integrity-protecting and verifying the data and ACK frames sent in the network.

Authentication of exchanged data frames is ensured only at a broad network level. That is, a message recipient can assert whether the message sender is any of the legitimate network nodes. However, strict source authentication of data messages is not ensured, i.e. a message recipient cannot assert whether the message sender is in fact the exact network node it claims to be. Of course, this enables attacks based on passive listening, message forgery and injection, as well as on identity impersonation through spoofing of addressing information. Consequently, 6P transactions are vulnerable to security attacks exploiting the commonly shared network key for protecting data frames.

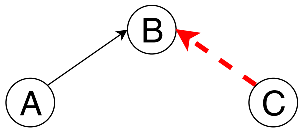

Fig. 3: Network reference topology.

IV. Threat Model

Hereafter, we consider an adversary which is internal, active and on-path, as well as in control of one network node. As an actual member of the network, the compromised node under the adversary's control knows the key material used in the network, especially the symmetric network key shared by all network nodes (see Section III-E). Thus, the adversary is able to read and alter the content of intercepted messages, and to create new cryptographically valid messages. Note that other relevant information, such as the current ASN used to build the cipher nonce, is instead even publicly available.

Also, the adversary is in the transmission/reception range of her intended victim nodes. Thus, she is able to intercept messages they send, and to inject (newly) forged messages for their reception. By possibly spoofing the source MAC address of injected messages, the adversary can practically induce a recipient node A to accept them as authentic and generated from any other impersonated network node B.

In this work, the adversary especially focuses on messages sent during 6P transactions, and is able to carry out a number of attacks against two neighbour nodes and their schedule. In particular, the adversary is able to intercept and forge 6P messages, and thus to carry out bogus 6P transactions between two neighbour nodes. This eventually results in the two nodes enforcing two different schedules with their respective peer, while still believing to have a same common schedule.

As an example, let us consider Fig. 3, where A and B are two legitimate neighbour nodes that communicate according to a consistent and shared schedule. For instance, in the DODAG formed by RPL, B can be the preferred parent of A. Instead, C is a compromised node under the control of the adversary, and in the reception/transmission range of both A and B. Of course, by exploiting the position of node C in the DODAG and in the physical topology, the adversary may be able to target multiple pairs of nodes as attack victims.

V. Attack General Scheme

Consistently with the threat model discussed in Section IV, the adversary builds on the following rationale to mount the two specific security attacks described in Section VI.

That is, node C considers a pair of neighbour nodes A and B, that share a consistent, previously established schedule $\mathcal{S}$. Then, the adversary produces an alternative schedule $\mathcal{S}'$ that can maximise the impact of the intended specific attack. For instance, the alternative schedule $\mathcal{S}'$ can be such that it does not have any cell in common with the original schedule $\mathcal{S}$.

After that, C impersonates A, and performs a bogus 6P transaction with B, in order to induce B installing the new schedule $\mathcal{S}'$. As a result, node A retains the original schedule $\mathcal{S}$, and continues assuming to share it with B. At the same time, B considers the new schedule $\mathcal{S}'$, and assumes it to be a new valid schedule shared with A.

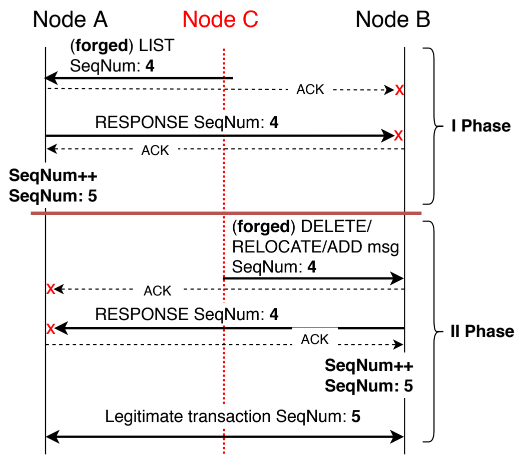

Fig. 4: General scheme for attacks.

Additionally, the adversary has to ensure that A and B not only erroneously *believe* to share a same schedule, but also that they *actually* remain aligned with their 6P SeqNum. This is required in order to prevent A and B from detecting that the attack has occurred in the first place, when they engage in a new 6P transaction. While it is easy for the adversary to learn the current SeqNum for A and B before the attack starts (i.e, by eavesdropping it), it is instead not trivial to have both A and B aligned with their SeqNum at the end of the attack, i.e. once the bogus 6P transaction with B is completed.

In order to successfully mount an attack, the adversary can perform it in two phases, as shown in Fig. 4. We assume that, before the attack starts, nodes A and B have SeqNum equal to 4. Then, the two attack phases are as follows.

In the first phase, node C impersonates node B, and performs a 6P transaction LIST by sending a request to node A. Upon receiving the request from C (impersonating node B), node A: i) sends a link-layer ACK to node B; ii) sends a response addressed to node B, including its schedule $\mathcal{S}$ shared with it; and iii) increments its SeqNum with B to 5, after the ACK sent by node B. Note that node B simply ignores both the ACK and the response received from A, as they do not match any ongoing 6P transaction from its own perspective. Also, node B sends an ACK as a reply to the ignored response from A, which does not impair the attack execution and, most importantly, allows A to increment its SeqNum. Instead, node C intercepts the response from A and thus gets knowledge of the original schedule $\mathcal{S}$.

In the second phase, node C generates an appropriate schedule $\mathcal{S}'$, and performs a bogus 6P transaction with node B. When doing so, C impersonates node A and uses as SeqNum the value 4, i.e., the one eavesdropped by C before the start of the attack. The particular 6P transaction, and hence exchanged messages, depends on the specific attack to mount. For instance, DELETE or RELOCATE can be used to mount the Traffic Dispersion attack described in Section VI-A, while ADD can be used to mount the Overloading attack described in Section VI-B.

Upon receiving the request from C (impersonating node

A), node B: i) sends a link-layer ACK to node A; ii) sends a response addressed to node A, including information consistent with the specific 6P transaction and the received request; and iii) increments its SeqNum with A to 5. Note that B correctly processes the request, as including the SeqNum 4 that node B was expecting for a next 6P transaction with node A. Also, node A simply ignores both the ACK and the response received from node B, as they do not match any ongoing 6P transaction from its own perspective. Besides, node A sends an ACK as a reply to the ignored response from B, which does not impair the attack execution.

Therefore, at the end of the second phase, nodes A and B share the same SeqNum 5. As a consequence, they are not going to experience inconsistencies during following 6P transactions (with other neighbours). At the same time, hereafter node A (B) considers the schedule $\mathcal{S}$ ($\mathcal{S}'$) for communicating with node B (A). As the two schedules do not partially or entirely match by construction, this has a disruptive impact on communications between nodes A and B, and severely impact their performance as well as network reliability as a whole.

A. Discussion on sequence number synchronisation

In the general attack scheme above, we assume that a node A ignores an incoming 6P message from another node B, if this includes a SeqNum that does not match with any ongoing transaction with B. This is consistent with the 6top standard [13], that leaves the possible definition and enforcement of different, more advanced policies to specific implementations.

While our assumption is in fact aligned with current common practices, different 6P implementations might handle the reception of 6P messages with unexpected SeqNum values in a different way. For instance, a node may consider a 6P response that does not match with any ongoing transaction as an inconsistency and take actions to resolve it, e.g. by performing a schedule rollback or a 6P CLEAR transaction.

In this case, the attack scheme requires to be extended, to ensure that it remains undetected to the pair of nodes A and B. Specifically, the adversary has to additionally suppress the ACKs and the 6P responses sent during the attack execution, e.g. by selectively jamming their transmission. This prevents the intended recipient to receive those responses, and hence to detect an inconsistency that can trigger reactions and thus neutralise the attack. In addition, for each suppressed 6P response, the adversary has also to inject a forged ACK message addressed to the sender of that response. This prevents the sender to possibly retransmit the response. With reference to Fig. 4, this applies to: i) the ACK and the response from node A to node B, during the 6P LIST transaction, in the first attack phase; and ii) the ACK and the response from node B to node A, in the second attack phase.

Since all the network nodes have a common synchronised view of slotframes and timeslots, the suppression of 6P responses through jamming can be easily implemented and efficiently carried out. In particular, the cells used for transmitting 6P messages are known to all the network nodes, hence the adversary can infer which frequency jam and during which timeslot, while possibly remaining inactive otherwise.

As a downside from suppressing 6P responses, the adversary is not able anymore to retrieve their content. With particular reference to the 6P LIST transaction in the first attack phase, the adversary cannot gain knowledge of the original schedule between nodes A and B from the response sent by A. Thus, the adversary has to learn the $\mathcal{S}$ schedule in advance and in a different way, e.g. by monitoring 6P transactions or other communications between A and B.

VI. Execution of Specific Attacks

In this section, we present two particular instances of the attack general scheme presented in Section V, namely the *Traffic Dispersion* attack and the *Overloading* attack.

A. Traffic Dispersion attack

The main goal of the Traffic Dispersion attack is to disrupt communications from a victim node A to its neighbour node B, by *dispersing* messages that A sends to B according to their (believed-to-be-)shared schedule. In particular, node B is the preferred parent of node A, in the DODAG built by RPL.

To achieve this goal, the adversary can actually perform two different variants of the Traffic Dispersion attack, based on the specific 6P transaction considered during the second attack phase (see Fig. 4). The first variant relies on a 6P transaction DELETE, and is hereafter referred as *DELETE attack*. The second variant relies on a 6P transaction RELOCATE, and is hereafter referred as *RELOCATE attack*.

Consistently with the attack general scheme presented in Section V, the Traffic Dispersion attack consists of two phases. First, the adversary forges a 6P LIST transaction, and retrieves the original schedule $\mathcal{S}$ between the nodes A and B. Then, in the second phase, the adversary induces B to install an alternative schedule $\mathcal{S}'$, by impersonating A and performing with B a 6P transaction DELETE, in the DELETE attack, or RELOCATE, in the RELOCATE attack.

Fig. 5 shows how the Traffic Dispersion attack is carried out, for the DELETE attack (a) and the RELOCATE attack (b), by a compromised node C under the adversary's control. 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 $\mathcal{S}'$, and is thus not able to receive any transmission from A.

Specifically, Fig. 5(a) shows how, before the DELETE attack starts, A and B share one common cell $c1$, on timeslot 18 and channel offset 3, in transmission and reception, respectively. Due to the bogus 6P DELETE transaction, B is induced to delete such only cell shared with A, and installs an empty alternative schedule $\mathcal{S}'$. Practically, B does not expect further communications from A. On the other hand, A preserves the original schedule $\mathcal{S}$. Finally, the adversary starts being active in receiving (RX) mode, during cell $c1$. 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,

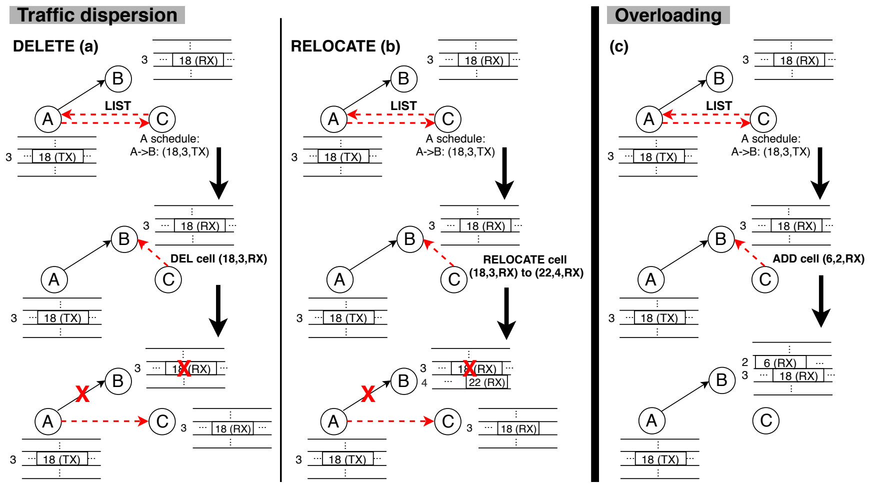

Fig. 5: Schedule manipulation throughout the attack execution

A will erroneously believe communications with B to be successful, and will not suspect of possible anomalies or schedule de-synchronisation that might otherwise result in a new negotiation. However, messages from A to B are not going to be received by B.

Similarly, Fig. 5(b) shows how, before the RELOCATE attack starts, A and B share the same common cell $c1$ as defined above. Due to the bogus 6P RELOCATE transaction, B is induced to replace the only cell shared with A, with the different cell $c2$, on timeslot 22 and channel offset 4, still in RX mode. 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. Also in this case, messages from A to B are not going to be correctly received by B.

B. Overloading attack

The main goal of the Overloading attack is to make a victim node B allocate unneeded cells to increase its energy consumption. This can be done by adding more cells in RX mode to its (believed-to-be-)shared schedule with its neighbour node A. More generally, the adversary can induce node B to add cells to its (believed-to-be-)shared schedule with multiple neighbour nodes. To achieve this goal, the adversary relies on a 6P ADD transaction during the second attack phase (see Fig. 4). As in the Traffic Dispersion attack, node B is the preferred parent of node A, in the DODAG built by RPL.

Specifically, Fig. 5(c) shows how, before the Overloading attack starts, A and B share one common cell $c1$, on timeslot 18 and channel offset 3, in transmission (TX) and RX mode, respectively. Due to the bogus 6P ADD transaction, B is induced to add one more cell $c2$, on timeslot 6 and channel offset 2, shared with A and in RX mode. That is, B installs an alternative schedule $\mathcal{S}'$, including both cell $c1$ and $c2$, and practically expects further communications from A on both cells from then on. On the other hand, 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 to A.

As a consequence, B will regularly be active during cell $c2$ at each slotframe. This has two practical effects. First, B is *wasting* a cell, that it could have rather considered for negotiation with a different neighbour. Second, as regularly listening for potential transmissions during cell $c2$ at each slotframe, B will consume additional energy, while never actually receiving any transmission from A.

VII. Evaluation

In this section, we present the evaluation of the impact on the network performance of the Traffic Dispersion and Overloading attacks. Initially, we tested the feasibility of the attacks by implementing them on real devices (see Section VII-A). The goal of this proof-of-concept implementation composed of only three nodes is to demonstrate the feasibility of implementing the operations required to mount the attacks on real devices, characterised by limited capabilities in terms of memory and computing. After demonstrating that the attacks are feasible in practice, we assessed their effects on a complete network through simulations (see Section VII-B).

A. Proof of concept experiments

In order to demonstrate the feasibility of the attacks and their impact on a simple scenario, we developed an implementation of the attacks on the Contiki-NG OS [1]. To this aim, the 6P implementation available on Contiki-NG was extended to support some missing 6P transactions, i.e., LIST, RELOCATE and CLEAR. The implementation was tested using the Zolertia RE-Mote [14] boards, 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.

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

For the evaluation of the feasibility of the attacks, we deployed the same topology with three nodes as the one reported in Fig. 3. That is, node B is programmed to behave as RPL root node, while node A and C are programmed as regular nodes, which send one packet every 5 seconds, whose payload is 5 bytes in size. The malicious node C, other than carrying out the operations of a regular node, is also programmed to mount the attacks. Each node is configured to use MSF as Scheduling Function [12] for the distributed allocation of cells.

Considering that the attacks are expected to impact the reliability of the packet delivery and the energy consumption of nodes, in all the experiments we measured the following metrics, in order to quantify the effects of the attacks:

- *Packet Delivery Ratio*, defined as the ratio between the total number of packets received at the RPL root node and the total number of packets generated in the network;
- *RX Energy consumption*, defined as the total amount of energy spent by a node when its radio is in RX mode;
- *TX Energy consumption*, defined as the total amount of energy spent by a node when its radio is in TX mode.

All the metrics are evaluated over time with a 60 second time-window granularity, to show the evolution of the network before and after the attacks. We tested both the Traffic Dispersion and the Overloading attacks. For each scenario, 5 different independent replicas of the experiments were run. In order to obtain statistically sound results, the average value over the samples collected from the different replicas are reported with its 95% confidence interval.

To compute energy consumption values, we relied on the Energest tool [2]. Energest is a Contiki-NG module that provides a lightweight, software-based energy estimation of IoT devices running Contiki-NG. Energest makes it possible to keep track of the time spent by the radio in RX and TX mode, and to estimate the overall energy consumption. To derive the energy consumption, we consider the power consumption values 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 [15] [16].

First, we tested the effectiveness of the Traffic Dispersion attack. In this case, the malicious node C is programmed to select node A as victim. Fig. 6 shows the Packet Delivery Ratio as a function of time. We can see that, for the first 180 seconds of the experiment, the root node B receives almost all the packets sent by the victim and the adversary, except for some sporadic losses caused by channel fluctuations and interference. At 180 seconds of the experiment, the adversary triggers the attack and performs a 6P DELETE transaction, impersonating node A, to delete the RX cells allocated by node B. After the transaction is completed, node A still transmits its packets on the TX cells allocated towards B. However, the packets are not received by B, as the RX cells were deleted from its schedule. As a consequence, after the completion of the attack, the percentage of packets received

[2]https://github.com/contiki-ng/contiki-ng/wiki/Documentation:-Energest
[3]http://www.ti.com/lit/ds/symlink/cc2538.pdf

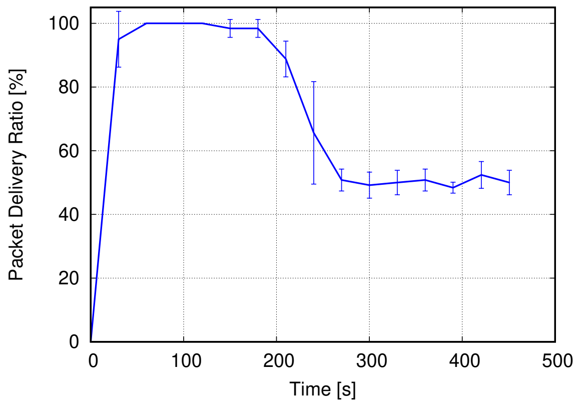

Fig. 6: Packet delivery ratio at the root node. Delete attack

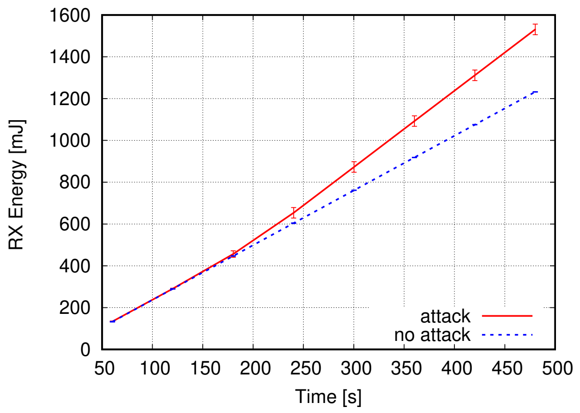

Fig. 7: RX Energy consumption of the root node with and without attack. Overloading attack.

by node B starts to decrease and drops up to 50%, showing that the only packets received by node B are those generated by node C. These results demonstrate that the attack is successful and it can be mounted on the devices considered in our experiments.

Then, we tested the effectiveness of the Overloading attack. In this case, the malicious node C is programmed to select node B, the RPL root node, as victim, to make it allocate unneeded cells towards A and increase its energy consumption. Since the impact of this attack is proportional to the number of the victim's children, we expect to have a small effect on the victim's energy consumption. Nevertheless, we run this experiment to assess its feasibility in practice, and to verify that it can have an impact on the victim's energy consumption, even on a small topology. In Fig. 7, we compare the RX energy consumption of the victim, with and without the attack. The TX energy consumption measurements are not shown here since, as it was expected, the attack did not influence them. At 180 seconds, the adversary starts performing the attack, through which it installs two additional RX cells with node A on the schedule of node B. A, however, remains unaware of those additional scheduled cells. As we can see in Fig. 7, the schedule manipulation results in an increased energy consumption on the victim. This is due to the fact that it turns on the radio during the two additional RX cells to receive possible transmissions from A, which, however, never occur. Thus, the impact of this attack is proportional to the number of additional RX cells that the attacker installs on B.

These results confirm that both the attacks can be mounted on real devices. Regardless of their limited computing capabilities, sensor nodes can timely execute all the functions required to effectively perform the attacks. Also, the results confirm that both the attacks can have an impact on the performance of the victim, even in a small scale scenario.

B. Simulation results

In order to evaluate the effects of the attacks on a larger network, we carried out a set of additional experiments aimed at an extensive performance evaluation. To this end, we used the Cooja network simulator [17], available as part of the Contiki distribution. Cooja allows the execution of the same code used for real devices, thus allowing the simulation of a network in a rapid manner.

For our simulations, we considered a topology with 30 nodes randomly deployed in an area of 200m x 200m. For each simulation the position of nodes is recomputed, except for the RPL root node that is always placed in a fixed position at the centre of the deployment area. The radio model used is the Unit Disk Graph Medium (UDGM) and we used 4 frequency channels. We allocated 3 cells for 6P messages and 1 cell for TSCH/RPL control traffic. An example of topology is shown in Fig. 8. The simulator has been configured to model a network of Cooja motes, i.e., generic sensor nodes equipped with an IEEE 802.15.4 radio. Each non-root node is programmed to generate periodic traffic towards the root node at a rate of one packet every 5 seconds, whose payload is 5 bytes in size.

For each experiment, one malicious node C under the adversary's control is randomly placed in the network. That node is programmed to perform the considered attack at 2200s from the beginning of the experiment. In particular, node C first synchronises with the TSCH network, and waits for the reception of a 6P message from one of its neighbours. Then, as soon as a 6P message is eavesdropped, node C selects the node which originated the 6P message as victim, and carries out the considered attack against it.

Three different scenarios with three different positions for node C are considered. Specifically, three different positions with an increasing distance from the root node are considered: (i) root level area: node C is placed in a circular area with radius 50m from the root node; (ii) middle level area: node C is placed in a circular area with a distance between 50m and 100m from the root node, i.e. at a medium level of the RPL topology; (iii) leaf level area: node C is placed with a distance of at least 100m from the root node, i.e., the victim is one of the leaves in the RPL topology.

For each scenario, we ran 20 independent replicas, in order to obtain statistically sound results. Each simulation lasts 60 minutes. The same metrics collected in the experiments reported in Section VII-A are considered also for the simulations.

1) Traffic Dispersion Attack: First, we analyse the Traffic Dispersion attack, which can be mounted in its two variants by using a 6P DELETE or RELOCATE transaction (see Section VI-A). In our experiments, we observed results displaying the same trend for both the attack variants. Thus, for the sake of brevity, here we report only the results for the variant based on the 6P DELETE transaction, i.e., the DELETE attack.

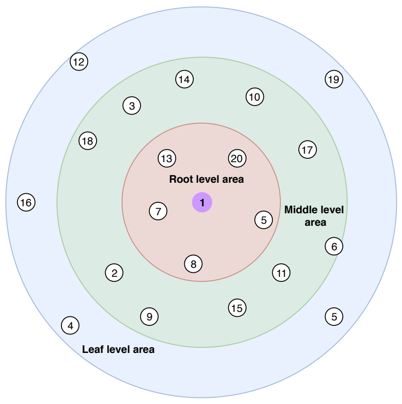

Fig. 8: Example of topology used in simulations.

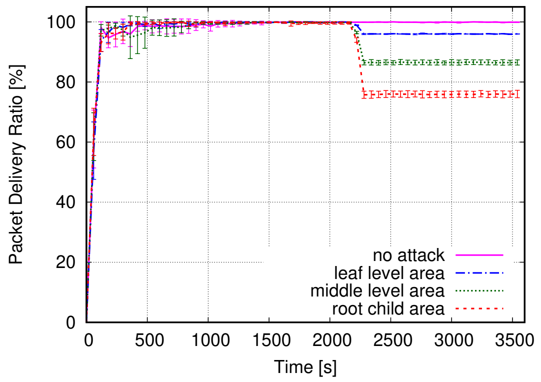

Fig. 9: Packet Delivery Ratio at the root node. DELETE attack.

Fig. 9 shows the Packet Delivery Ratio when a DELETE attack is performed. The solid line represents the Packet Delivery Ratio over time in an attack-free case, while the other lines report the results obtained when the attack is performed, for different positions of the malicious node C. We can see that, right after the execution of the attack, the overall Packet Delivery Ratio reduces, due to the effect of the dispersion of some packets caused by the attack. As expected, the impact of the attack greatly depends on the position of the victim, and consequently of node C, in the network. That is, the closer to the root the victim is, the greater the effects. Specifically, the maximum effect is obtained when node C is placed in proximity of the root node, with a drop in the delivery ratio of approximately 25%, while the lightest effect is obtained when the victim node is a leaf, with a drop of approximately 5%. This can be explained considering that a node forwards all the traffic of the nodes in its sub-tree. Thus, the larger its sub-tree, the higher the amount of traffic that node has to forward. Due to the attack, all the traffic supposed to be forwarded by the victim is dropped. Hence, the larger the sub-tree the more significant the impact. On the contrary, when the victim node is a leaf, the attack only causes the drop of the packets originated by the victim itself.

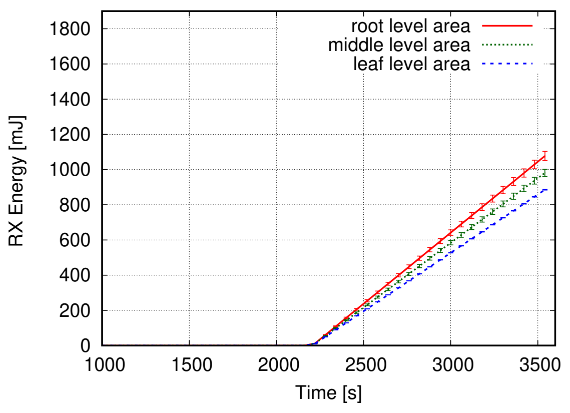

Fig. 10: RX energy consumption. Adversary. DELETE attack.

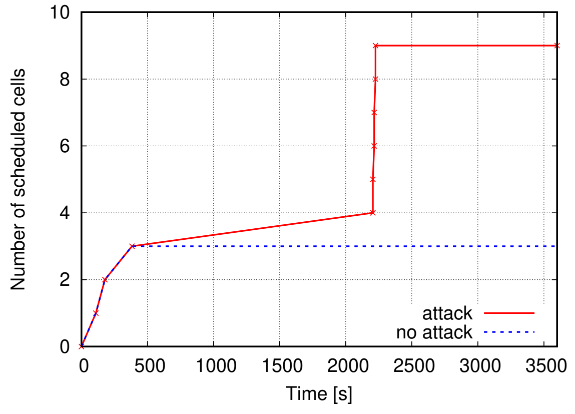

Fig. 12: Cells scheduled on the victim. Overloading attack.

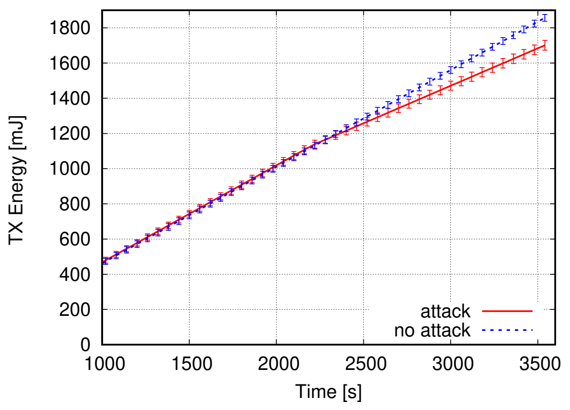

Fig. 11: TX energy consumption. Victim's parent node (root node). DELETE attack.

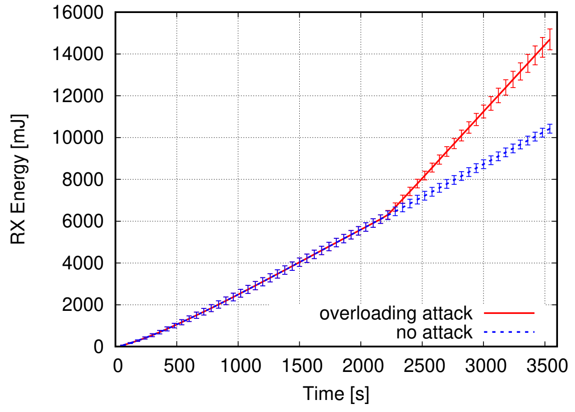

Fig. 13: RX Energy consumption. Victim with and without overloading attack.

Hence, the impact on the Packet Delivery Ratio is limited.

In order to evaluate the cost of mounting the attack for the adversary, in Fig. 10 we report the average RX energy consumption of node C. The graph reports only the additional energy cost due to the attack execution. That is, it shows only the energy that node C additionally consumes to intercept the packets sent by the victim on the deleted RX cells, in order to reply back with ACK frames and ensure the attack remains undetected (see Section VI-A). As expected, also in this case, the closer to the root node, the higher the cost of the attack. This can again be explained with the higher amount of traffic received by node C, that hence spends more time with its radio in RX mode. These results show that performing the Traffic Dispersion attack has a minimal cost in terms of additional energy, and is feasible to mount even when relying on battery powered devices.

To conclude the analysis of the Traffic Dispersion attack, in Fig. 11 we report the average TX energy consumption of the parent of the victim node. For the sake of brevity, we report the measurements obtained in the scenarios with node C placed in the root area and in the attack-free scenario. As expected, when the attack is performed, the parent node of the victim displays a lower energy consumption for its radio in TX mode. This is due to the fact that the victim's parent receives a smaller number of packets, and thus transmits less ACK frames to acknowledge their reception.

2) *Overloading:* In the following, we analyse the effects of the Overloading attack (see Section VI-B). Unlike the Dispersion attack, the impact of this attack is not affected by the position of the victim node in the network topology. For this reason, in each experiment, the malicious node C is always positioned in proximity of a node which has three neighbours and is selected as victim. We focus on analysing the impact on energy consumption, as this attack is expected to mainly affect the energy consumption of the victim.

To get an insight on the dynamics of the the attack, in Fig. 12 we report the number of cells scheduled on the victim node over time, throughout a single simulation run. In the attack-free scenario, as well as before the attack is performed at 2200s, in the attack scenario, the victim has only one cell scheduled per child, i.e., a total of 3 cells. After the attack is performed, the victim has a total of 9 RX cells allocated, 6 of which resulting from the attack execution.

In Fig. 13, we report the average RX energy consumption of the victim node, in the attack-free and in the attack scenario. The attack is performed at 2200s from the beginning of the experiments, and results in the allocation of two additional cells per child on the victim schedule. As expected, after its completion, the attack results in a significant increase in the RX energy consumption of the victim. To better understand the practical implications, we derive the impact on the battery lifetime of a sensor node. To this aim, we consider a device powered by two AA 1.5V batteries, which can provide a total capacity of 25920 J. The victim node consumes 0,004 J per second, while a node under normal operation consumes 0,003 J per seconds. This results in a battery lifetime of 100 days that decreases to 75 days if the attack is executed shortly after installation. This can result in a battery drain that is 25% faster.

C. Attack mitigation and countermeasures

The presented attacks are possible since, in practical settings, 6P transactions are a convenient target for compromised network nodes. In fact, practical settings rely on a single symmetric key - network key - shared by all the network nodes to secure data and acknowledgement frames. Thus, if compromised and under control of an adversary, a network node can perform any of the attacks discussed above. Actually, any (compromised) node in the network, as owning the network key, is able to inject or alter 6P messages. As a consequence, 6P negotiations, as intended to concern a pair of nodes, become practically exposed to passive monitoring and active manipulation from any network node, as a way to alter the schedules of selected victim nodes.

To detect and mitigate these types of attacks, monitoring functionalities can be added to the used Scheduling Function. Such extensions can regularly monitor different network parameters, e.g. RX cell utilisation, and check the consistency between enforced schedules and message exchanges. This would provide relevant information to possibly run countermeasures and mitigate the attack impact. For instance, a Scheduling Function may monitor the actual message reception on scheduled RX cells, and de-allocate them if no (sufficient amount of) traffic is received during those cells. However, even if effects would be mitigated and more limited in time, attacks can still fundamentally impair the network performance until they are actually detected, with consequent eviction of the compromised nodes.

A more effective solution would be using symmetric pairwise keys, each of which shared by a single pair of neighbour nodes. By doing so, strict source authentication of messages is achieved, including for those exchanged during 6P transactions, thus preventing the considered attacks altogether. On the other hand, the distribution, establishment and revocation of of link-layer pairwise keys is a non-trivial, often discouraging task, which practically paves the way to using shared network keys. Approaches based on the establishment and usage of symmetric pairwise keys are left as future work.

VIII. Conclusion

In this paper, we have presented and assessed two attacks against 6P on IIoT networks based on the 6TiSCH architecture, namely the Traffic Dispersion and Overloading attacks. We have proved the feasibility of their execution and evaluated their impact on the network performance, both experimentally on real devices and by means of simulations. Our analysis confirms that both attacks are feasible, and can have a significant impact in terms of reduced reliability and increased energy consumption. To the best of our knowledge, no previous work has analysed the security of 6top. Future work will evaluate the cost and impact of these attacks on large-scale testbeds, and will study the design of possible 6P extensions to mitigate the impact of the attacks. Finally, we plan to analyse the feasibility and cost of approaches for counteracting the presented attacks based on the establishment of pairwise symmetric keys.

Acknowledgments

This work was partially supported by the Italian Ministry of Education and Research (MIUR) in the framework of the CrossLab project (Departments of Excellence), as well as by VINNOVA and the CelticNext project CRITISEC.

References

[1] P. Thubert, “An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4,” Internet Engineering Task Force, Tech. Rep. draft-ietf-6tisch-architecture-28, October 2019, (W.I.P.).

[2] “Ieee standard for low-rate wireless networks,” *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)*, April 2016.

[3] M. Vučinić, J. Simon, K. Pister, and M. Richardson, “Constrained Join Protocol (CoJP) for 6TiSCH,” Internet Engineering Task Force, Tech. Rep. draft-ietf-6tisch-minimal-security-15, Dec. 2019, (W.I:P.).

[4] T. Watteyne, M. Palattella, and L. Grieco, “Using IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) in the Internet of Things (IoT): Problem Statement,” RFC 7554 (Informational), RFC Editor, Fremont, CA, USA, May 2015.

[5] S. Kim, H.-S. Kim, and C. Kim, “Alice: Autonomous link-based cell scheduling for tsch,” in *Proceedings of the 18th International Conference on Information Processing in Sensor Networks*. ACM, 2019.

[6] F. Righetti, C. Vallati, G. Anastasi, and S. K. Das, “Analysis and improvement of the on-the-fly bandwidth reservation algorithm for 6tisch,” in *2018 IEEE 19th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*.

[7] M. Tiloca, D. D. Guglielmo, G. Dini, G. Anastasi, and S. K. Das, “DISH: DIstributed SHuffling Against Selective Jamming Attack in IEEE 802.15.4e TSCH Networks,” *ACM Transactions on Senor Networks*, Dec. 2018.

[8] W. Yang, Q. Wang, Y. Wan, and J. He, “Security vulnerabilities and countermeasures for time synchronization in ieee802.15.4e networks,” in *2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud)*.

[9] X. Vilajosana, K. Pister, and T. Watteyne, “Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration,” RFC 8180 (Best Current Practice), RFC Editor, Fremont, CA, USA, May 2017.

[10] Z. Shelby, K. Hartke, and C. Bormann, “The Constrained Application Protocol (CoAP),” RFC 7252 (Proposed Standard), RFC Editor, Fremont, CA, USA, Jun. 2014, updated by RFC 7959.

[11] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, “RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks,” RFC 6550 (Proposed Standard), RFC Editor, Fremont, CA, USA, Mar. 2012.

[12] T. Chang, M. Vučinić, X. Vilajosana, S. Duquennoy, and D. Dujovne, “6TiSCH Minimal Scheduling Function (MSF),” Internet Engineering Task Force, Tech. Rep. draft-ietf-6tisch-msf-10, Dec. 2019, (W.I.P.

[13] Q. Wang, X. Vilajosana, and T. Watteyne, “6TiSCH Operation Sublayer (6top) Protocol (6P),” RFC 8480 (Proposed Standard), Fremont, CA, USA, Nov. 2018.

[14] “Zolertia re-mote,” https://bit.ly/37rgqZT, accessed: 2020-01-25.

[15] A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He, “Software-based online energy estimation for sensor nodes,” in *Proceedings of the 4th Workshop on Embedded Networked Sensors*. ACM, 2007.

[16] X. Fafoutis, A. Elsts, A. Vafeas, G. Oikonomou, and R. Piechocki, “On predicting the battery lifetime of iot devices: Experiences from the sphere deployments,” in *Proceedings of the 7th International Workshop on Real-World Embedded Wireless Systems and Networks*. ACM, 2018.

[17] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, “Cross-level sensor network simulation with cooja,” in *31st IEEE Conference on Local Computer Networks*, 2006.