Mitigating Energy Depletion Attacks in IoT via Random Time-Slotted Channel Access

Savio Sciancalepore*, Pietro Tedeschi†, Usman Riasat†, Roberto Di Pietro†
*Eindhoven University of Technology, Eindhoven, Netherlands
s.sciancalepore@tue.nl
†Division of Information and Computing Technology (ICT)
College of Science and Engineering (CSE), Hamad Bin Khalifa University (HBKU), Doha, Qatar
{ptedeschi, uriasat, rdipietro}@hbku.edu.qa

***Abstract*—Energy depletion attacks represent a challenging threat towards the secure and reliable deployment of low-power Internet of Things (IoT) networks. Indeed, by simply transmitting canning standard-compliant packets to a target IoT device, an adversary can quickly exhaust target devices' available energy and reduce network lifetime, leading to extensive Denial-of-Service (DoS). Current solutions to tackle energy depletion attacks mainly rely on *ex-post* detection of the attack and the adoption of follow-up countermeasures. Still, the cited approaches cannot prevent external adversaries from sending wireless packets to target devices and draining down their energy budget.**

In this paper, we present RTSCA, a novel countermeasure to energy depletion attacks in IoT networks, that leverages Random Time-Slotted Channel Access. RTSCA randomizes channel access operations executed by a couple of directly-connected IoT devices operating through the IEEE 802.15.4 MAC, significantly reducing the time window of opportunity for the attacker, with little-to-none energy cost on legitimate IoT devices. RTSCA also includes a detection mechanism targeted to the recently-introduced Truncate-after-Preamble (TaP) energy depletion attacks, that leverages the observation of error patterns in the received packets. We carried out an extensive performance assessment campaign on real Openmote-b IoT nodes, showing that RTSCA forces the adversary to behave as a (sub-optimal) reactive jammer to achieve energy depletion attacks. In such a setting, the adversary has to spend between 42.5% and 55% more energy to carry out the attack, while at the same time having no deterministic chances of success.

***Index Terms*—IoT Security; Energy Depletion Attacks; IEEE 802.15.4; Random Time-Slotted Channel Access.**

I. Introduction

The weak security and privacy protection mechanisms offered by the first generation of Internet of Things (IoT) products and platforms motivated extensive scientific literature, as well as the design of innovative security solutions tailored to the hardware and software constraints of IoT devices [1].

Although security protocols and suites for IoT have nowadays significantly evolved, *energy depletion* attacks remain a hard-to-face threat, especially in Low Power Networks (LPW) based on the latest IEEE 802.15.4 specification [2]. Energy depletion attacks aim at exhausting IoT devices' batteries, by maliciously triggering the target nodes to execute standard-compliant but energy-consuming operations [3]. Despite such attacks can be contextualized to any layer of the protocol stack, preventing their launch at the Physical (PHY) and Medium Access Control (MAC) layer is particularly challenging. Indeed, the adversary simply needs to deliver protocol-abiding packets to the target node and let it process (and eventually discard) such packets, according to standard protocol operations. Given that radio and security procedures are the most energy-consuming activities for an IoT device, causing their regular execution leads to energy drain, rapidly exhausting devices' battery and shortening network lifetime [4].

As demonstrated by recent scientific contributions, such as the *Ghost* attack presented in [5] and the Truncate-after-Preamble (TaP) attack introduced by [6], an adversary can launch energy depletion attacks stealthily and efficiently (i.e., incurring a low energy cost), by manipulating the security fields or the PHY header of an IEEE 802.15.4 packet.

Current solutions conceived to mitigate general-purpose and technology-agnostic energy depletion attacks are all based on attack detection. Indeed, energy depletion attacks require several rounds to attain a noticeable level of energy exhaustion, and their timely detection by the target can limit the energy drain. However, current solutions cannot prevent an external adversary from launching continuous energy depletion attacks. Indeed, they all expose to the adversary the radio access patterns of the legitimate devices, an information that can be exploited by the adversary to mount a sustained energy depletion attack.

Contribution. In this paper, we present RTSCA, a novel and efficient solution to prevent energy depletion attacks in LPW networks based on the IEEE 802.15.4 PHY/MAC. RTSCA requires little-to-none additional energy cost on legitimate IoT devices. At the same time, RTSCA prevents energy depletion attacks launched by external adversaries by randomizing the time when a device listens to the wireless channel for incoming packets, so not exposing to the adversary the radio access patterns. Furthermore, RTSCA provides a novel pure-software strategy that detects and mitigates the impact of recently-disclosed TaP energy depletion attacks, based on the analysis of error patterns within received packets.

Our experimental performance assessment carried out on real Openmote-b IoT devices shows that RTSCA reduces the effectiveness of current energy depletion attacks to at least 10.8% of their nominal effectiveness. Such value can be improved at will by the IoT devices by increasing the time duration of the IEEE 802.15.4 slot, trading off such gain with a slight reduction of the network throughput. We also show that, to have chances of depleting the devices' energy, the adversary has to deploy a (sub-optimal) reactive jammer, increasing the required energy consumption by a $42.5 - 55\%$ range, in broadcast and unicast scenarios, respectively, hence incurring a significantly reduction in efficiency, stealthiness, and effectiveness.

To the best of our knowledge, we are the first ones to point out and to demonstrate that solutions like RTSCA, traditionally conceived to avoid random spot jamming, can be adapted and re-used to efficiently and effectively prevent PHY/MAC energy depletion attacks, with little-to-none additional cost on energy-constrained IoT devices.

Roadmap. The rest of the paper is organized as follows. Section II reviews related work, Section III introduces the system and the adversary models, Section IV introduces the attacks tackled in our paper, Section V illustrates RTSCA, Section VI provides the assessment of the proposed approach and, finally, Section VII tightens conclusions.

II. Related Work

A few works recently investigated energy depletion attacks and related countermeasures in IoT networks based on the IEEE 802.15.4-2015 PHY/MAC (see [2] for a comprehensive overview). In this context, the *Ghost* attack presented in [5] showed that an adversary could transmit replayed secure packets to a target, in a way to induce it into executing energy-consuming AES decryption operations, quickly exhausting its energy. In the same paper, the authors highlighted a few potential countermeasures, based on the localization of the adversary via the Cluster Heads (CHs) of the network and the spread of such information through dedicated protocols to all the nodes in the network. However, the cited countermeasures do not prevent energy depletion, but only detect the attack after suffering the injecting of a few bogus packets.

The authors in [7] focused on the Coordinated Sampled Listening (CSL) operation mode of the IEEE 802.15.4 MAC and Denial-of-Sleep attacks, specific to this operation mode. This class of attacks leverages optional security of the wake-up phase of the CSL to let the target spend more time in the energy-consuming RX mode, thus draining its energy quicker. The countermeasures deployed to face such attacks are specific to the CSL mode of IEEE 802.15.4, and hardly generalize to any MAC based on IEEE 802.15.4.

Some flavours of jamming attacks are generally considered a form of energy depletion for LPW [8]. Specifically, deceptive and reactive jamming attacks lead a target to receive and process a standard-compliant packet [9], emerging as a form of energy depletion. While current solutions mainly look at ways to achieve information transfer under jamming attacks, solutions to face jamming from the energy depletion perspective mostly fall in the jamming detection area, e.g., [10]. However, they cannot be generalized to detect and face any PHY/MAC energy depletion attack.

The authors in [6] further decreased the energy necessary by an adversary to launch energy depletion attacks. They proposed delivering a packet containing only the PHY header, letting an IoT node receive a packet of the Maximum Transmission Unit (MTU) size. As explained in this paper, such attacks can be detected efficiently, and necessitate the knowledge of the reception time(s) on the target to be successful.

Our solution emerges as a horizontal strategy compared to the cited works, preventing any external adversary from injecting bogus packets into the network. On the one hand, our countermeasure significantly mitigates external energy depletion attacks, as it *hides* the time when a device can receive a packet. On the other hand, the previously-mentioned strategies can be combined with the proposed approach, further enhancing the protection of an LPW against stealthy energy depletion attacks.

Finally, note that energy depletion attacks are not limited to PHY and MAC layers, but they can target also routing and application layers (see, e.g., the vampire attack investigated in [11]). To face such attacks, specific protocol-based countermeasures should be conceived, with a limited impact on the behaviour of the lowest layers of the protocol stack.

III. System and Adversary Model

This section introduces the system model and the adversarial model considered in our manuscript. Specifically, Section III-A depicts the scenario, while Section III-B describes the features and objective of the adversary.

A. System Model

The scenario assumed in this work is depicted in Figure 1.

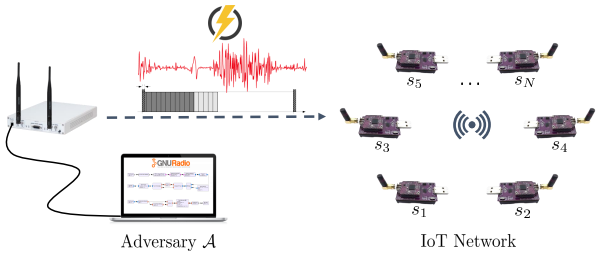

Fig. 1. Scenario assumed in this work.

We consider an LPW IoT network made up of N nodes, namely $s_1, s_2, \ldots, s_N$, wirelessly connected to each other in a full-mesh topology. We assume that the IoT nodes use the widespread IEEE 802.15.4 PHY/MAC layer to orchestrate the access to the physical communication medium and exchange radio packets [12]. We do not make any additional assumption about the protocol stack of the IoT nodes, so as to consider the worst case of nodes connected to each

other only at the MAC layer (the reason for this conservative assumption will become clear when we discuss the attack detection strategy in Section V-C). In line with the latest IEEE 802.15.4 specifications, we assume that the IoT nodes can transmit packets with a random MAC payload size, varying from the minimum of $P_{MIN} = 28$ bytes to the maximum of $P_{MAX} = 127$ bytes. Thanks to adopting the IEEE 802.15.4 MAC layer, we can safely assume that all the IoT nodes are synchronized. Specifically, in line with the latest IEEE 802.15.4 specifications, we assume that time is divided into *time-slots*, lasting a fixed amount of time $T = 10$ ms. We recall that IEEE 802.15.4 is a deterministic MAC, meaning that any event happening within the time-slot T occurs at a specific, predictable time $t \leq T$. IEEE 802.15.4 also guarantees that any packet that a device needs to transmit gets transmitted within the time-slot T. Note that a packet can be transmitted on any of the 16 IEEE 802.15.4 channels, in the frequency range $[2,405 - 2,480]$ Mhz [13].

We also assume that any couple of directly-connected IoT nodes, e.g., s_1 and s_2, share a secret symmetric key, namely $K_{1,2}$. The specific way this secret is generated and installed in the devices is out of the scope of our manuscript, and it can be either pre-installed or dynamically obtained by the devices through an interactive key agreement scheme [14], [15].

Finally, we assume that the IoT devices support the execution of an hashing function, such as SHA-256, being supported efficiently in hardware by many IoT devices [16].

B. Adversary Model

The adversary assumed in this work, namely $\mathcal{A}$, features both passive and active capabilities. On the one hand, $\mathcal{A}$ is a global passive eavesdropper listening on the same wireless communication channel(s) used by legitimate IoT devices. Therefore, $\mathcal{A}$ can detect and decode any packet exchanged by any couple of devices, independently from the specific frequency and location of the devices. On the other hand, $\mathcal{A}$ also features active capabilities, i.e., it can inject radio packets on the wireless channel. In particular, as will be detailed in Section IV, $\mathcal{A}$ can also craft special bogus packets, announcing a packet length different from the real one, and aborting the transmission at any time, before its expected conclusion. These powerful capabilities identify $\mathcal{A}$ as a Software-Defined Radio (SDR), able to change the standard behaviour of an IEEE 802.15.4 transceiver at the PHY layer.

We also assume that $\mathcal{A}$ is an energy-constrained adversary, i.e., it has limited energy available. This assumption is reasonable, especially when the adversary targets remote IoT deployments, and needs to power up the SDR with an external portable battery. The less energy the attack requires, the more time the adversary can have its equipment fully powered, increasing its chances to fully deplete the energy of the target nodes before the exhaustion of its own energy budget.

Overall, the aim of the adversary $\mathcal{A}$ is to stealthily deplete the energy of the legitimate IoT device as much and as quickly as possible, while minimizing its energy expenditure. Further, the adversary would also like be stealthy, that is, having its presence not detected. Note that *stealthiness* is not an additional feature of the assumed adversary, but a *requirement* of any energy depletion attack. Indeed, to exhaust the energy of a target device, the adversary should launch energy depletion attacks in multiple rounds. Thus, avoiding detection on a per-packet basis is the only way the adversary has to achieve its objective—i.e., stealthiness prevents the target nodes from adopting countermeasures.

Finally, note that this paper assumes an *external adversary*, i.e., an adversary using a device external to (not part of) the network. Carrying out the attacks described in this manuscript through an internal adversary would require the IoT network to natively support at least one SDR, which is unusual. Therefore, the case of an *internal adversary* is intentionally left out of the scope of this manuscript.

IV. Orchestrating Energy Depletion Attacks

This section sheds lights on energy depletion attacks, including the TaP attack described in [6], contextualizing them into the system model above discussed. Energy depletion attacks distinguish between a passive phase and an active phase, described in Section IV-A and Section IV-B, respectively.

A. Passive Phase

In this phase, $\mathcal{A}$ understands the channel access patterns of the target device(s) by observing the radio activities on the wireless channel. Figure 2 reports a qualitative sequence diagram of the legacy operations executed by a transmitting and a receiving IoT device compliant to the IEEE 802.15.4 PHY/MAC to transfer a unicast data packet in a time-slot.

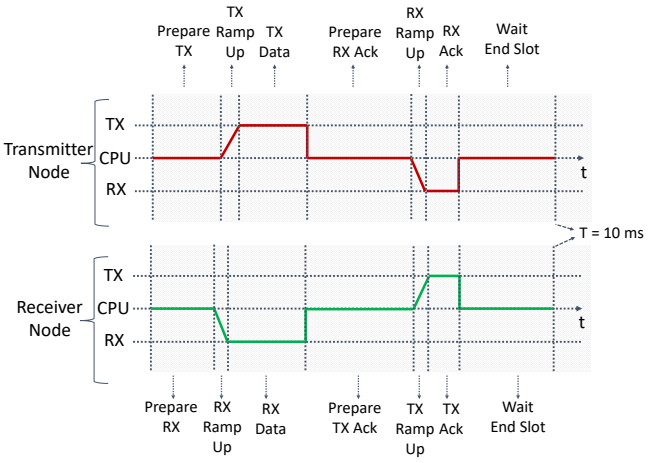

Fig. 2. Operations executed by a transmitting and receiving IoT device compliant to the IEEE 802.15.4 PHY/MAC to exchange a unicast packet, in a default time-slot of $T = 10$ ms. If the packet is broadcast, both the devices wait for the end of the slot after the reception of the data packet.

We notice that the Transmitter IoT node has a maximum time to prepare the packet, defined as *PREPARE TX*. Then, at a given (well-known) time, it instructs the radio to transmit the packet. The radio takes a time *TX RAMP UP* to turn on in TX Mode, and then transmits the packet for a maximum time *TX DATA*, based on its size. Then, the transmitter turns off

the radio and stays silent for a time *PREPARE RX ACK*. At a given (well-known) time, it instructs the radio to turn on in RX mode, to receive the acknowledgement. The radio takes a time *RX RAMP UP* to turn on in RX mode, and receives the ack packet for a (fixed) time *RX ACK*. In the remaining part of the slot, lasting for a time *WAIT END SLOT*, the nodes turn off the radio to minimize its energy consumption. The qualitative sequence diagram of the receiver node is the dual of the transmitter.

Note that the specific time offsets within the qualitative sequence diagram in Figure 2 are static, i.e., they do not change slot-by-slot. The only parts that have a variable duration are *TX DATA* and *RX DATA*, which takes a different time based on the size of the payload, with a consequent shrink of the part at the end of the slot, namely *WAIT END SLOT*. However, given that the first byte of the whole packet (the PHY header) reports the size of the packet, an external observer (e.g., $\mathcal{A}$) can easily acquire such information.

Based on the above considerations, in this phase, the adversary $\mathcal{A}$ simply tunes its radio on the same communication channel used by the target device and records the timestamps of all the packets (and corresponding acknowledgements) exchanged between the target device and other IoT nodes in the network. By observing only a few packets, $\mathcal{A}$ can easily *synchronize* with the network and predict the expected transmission and reception times of the target device.

Note that this knowledge is key to the successful deployment of any energy depletion attack. Indeed, it provides the adversary with the capability to guess the (reduced) portion of time when the target has its radio active in RX mode. Without such a knowledge, the adversary has a significantly reduced chance of successfully injecting its packets into the target device.

B. Active Phase

In the *active phase*, leveraging the knowledge previously acquired, the adversary $\mathcal{A}$ injects locally-generated packets to induce the receiver to process them and waste energy.

In the case of a *traditional* energy depletion, $\mathcal{A}$ simply injects its packet at the time instant where it expects the target to have its radio on, anticipating and replacing the legitimate one.

In the case of TaP attacks, as discussed by the authors in [6], $\mathcal{A}$ inserts specially-crafted packets. Specifically, IEEE 802.15.4 packets are logically divided into five parts: (i) the preamble and Start of Frame Delimiter (SFD), indicating the beginning of an IEEE 802.15.4 packet; (ii) the PHY header, indicating the size of the packet, in bytes; (iii) the MAC header, providing additional meta-data according to a well-defined format; (iv) MAC payload, containing upper-layer data; and, finally, (v) the MAC trailer, consisting of a Cyclic Redundancy Check (CRC) allowing to verify the integrity of the packet. Any receiver can detect an incoming IEEE 802.15.4 packets through the detection of the preamble and the SFD, and then *reads* the value of the PHY header to identify the time it needs to have its radio on in RX mode, based on the packet size. As shown in Figure 3, TaP attacks inject packets carrying a bogus PHY header, i.e., an incorrect packet size.

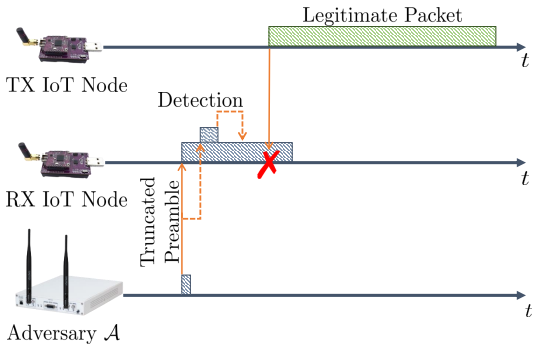

Fig. 3. Operation of the TaP attack.

Specifically, in the most effective configuration, the PHY header of a TaP packet reports the value 127, i.e., the MTU of an IEEE 802.15.4 packet. Then, the adversary simply truncates the transmission. On the receiving side, the target IoT node will spend time and energy receiving *noise* for the maximum possible time, and then it will discard the packet because of an incorrect CRC.

Note that TaP attacks are particularly advantageous for an energy-constrained adversary. Indeed, by transmitting only a single data byte (the PHY header), the adversary causes the target to have its radio on in RX mode for the time equivalent to the transmission of 127 bytes, causing the maximum possible disruption. Note that the adversary can stop the packet transmission any time during the expected payload transmission, not necessarily after the first byte. This feature will be helpful to the adversary, as described later on in Section V, where we will discuss the detection strategy included in RTSCA.

We recall that the authors in [6] discussed only the *active phase* of the TaP attack, i.e., the creation and delivery of the bogus packets, neglecting considerations about the preparation of the attack, i.e., the *passive phase*. Moreover, they only described a specific *configuration* of the attack, i.e., the transmission of a fixed payload of 10 bytes before aborting transmission. In this work, among other contributions (cfr. *Contribution*), we also extend the basic attack configuration, providing more flexibility to the attack at the expense of a slight increase in energy consumption.

V. Mitigating Energy Depletion Attacks

This section describes the countermeasure we conceived to mitigate energy depletion attacks. Moreover, we describe a detection strategy tailored to the unique features of the TaP attack, useful to further increase the energy consumption required on the adversary to launch the attack. Section V-A provides an overview of our defense strategy, while Section V-B and Section V-C describe the countermeasure and the detection strategy, respectively.

A. RTSCA in a Nutshell

The devised countermeasure, namely RTSCA, relies on (pseudo-random) synchronized communications, i.e., a pseudo-random time-slotted synchronized channel access. In brief, each couple of directly-connected IoT devices uses a shared secret to extract a shared delay, adopted to randomize (from the adversarial point of view) radio activities, being it a packet transmission or reception. The delay is extracted so that the overall time required to transmit (receive) a packet and receive (transmit) the corresponding acknowledgement does not exceed the time-slot duration T.

On the one hand, RTSCA has no impact on the energy consumption of the IoT devices and the *determinism* provided by the IEEE 802.15.4 MAC. Indeed, the time the IoT devices have their radio turned on (either in TX or in RX) is the same as the *standard* one. Moreover, the devised strategy still ensures that any packet is delivered to the receiver within the maximum delay T. On the other hand, an external adversary, not in possession of the shared secret, cannot deterministically predict when any IoT device would turn its radio on in RX mode, finding it hard to inject bogus packets. Note that this countermeasure effectively mitigates energy depletion attacks and apply to any run-time energy depletion attack.

RTSCA also includes a detection strategy specific to the TaP attack. The detection is grounded on the observation that an adversary carrying out a TaP attack has no control over the bytes received by the target IoT device. Therefore, so received packets have a different bytes error pattern than regular ones caused by mutual *interference*. In brief, instead of simply discarding a packet with an erroneous CRC, an IoT device running RTSCA processes the MAC header of an erroneous packet, and it counts the number and the distribution of such incorrect fields, i.e., the fields having an *incoherent* value. If such values exhibit the pattern typical of a TaP attack, RTSCA raises the alarm for TaP detection, compromising the stealthiness of the attack. The following subsections provide more details.

B. Preventing Energy Depletion Attacks

As previously summarized, RTSCA relies on randomized time-slotted channel access, as depicted in Figure 4.

In brief, RTSCA uses the time available at the end of a time-slot, namely *WAIT END SLOT*, currently unused, to randomly delay the radio operations, in a way to make them unpredictable to an external observer.

On the transmitter, the IoT device first prepares the packet to be transmitted (PREPARE TX). Then, instead of immediately transmitting the packet, it extracts a slot-dependent delay. To this aim, it executes the operation reported in Eq. 1.

$$l_{1,t} = \mathcal{H}\left[asn \oplus K_{1,2}\right] \mod R, \tag{1}$$

where $\mathcal{H}$ refers to an hashing algorithm (such as SHA-256), asn is the current value of the Absolute Slot Number (ASN), i.e., the ordinal number of the slot since the deployment of the network, $K_{1,2}$ is the key shared between the transmitter and the receiver, $\mod$ refers to the modulo operator, $\oplus$ refers to the bitwise x-or operator, while R is the maximum time available within the slot, corresponding to *WAIT END SLOT* in Figure 2. Such a formulation, adopted in recent works such as [17], provides confidentiality properties equivalent to well-known symmetric encryption algorithms, at significantly reduced computation cost.

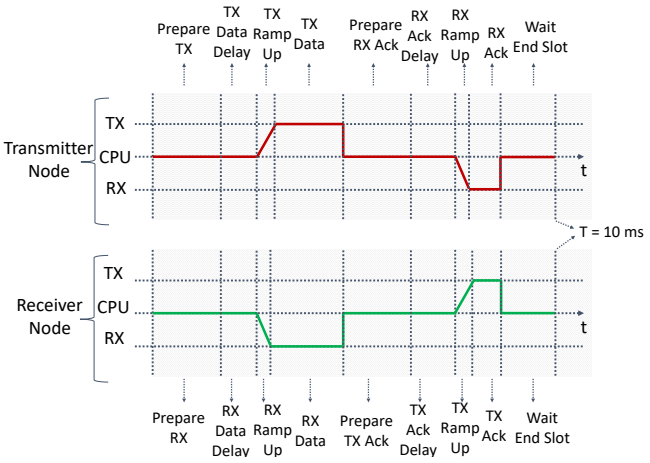

Fig. 4. Operations executed by a Transmitter and Receiver IoT device in a IEEE 802.15.4 time-slot integrating RTSCA.

The time $l_{1,t}$ is the value of the *TX DATA DELAY* in Figure 4. During this time, the transmitter does not turn on the radio but stays idle, consuming reduced energy (CPU only). At the expiration of the delay, the node transmits the packet.

Both the key $K_{1,2}$ and the current value of the asn are known to the receiver, as the two nodes are synchronized thanks to the implicit IEEE 802.15.4 synchronization mechanism. Therefore, the receiver can also compute the value *TX DATA DELAY*, and avoid turning on the radio in advance. At the time *TX DATA DELAY - 1*, the receiver turns on the radio and receives the packet. Then, if the received packet requires an acknowledgement (unicast packet), it prepares the ack. Similarly to the previous behavior, it extracts a new delay, namely *TX ACK DELAY*, as in Eq. 2.

$$l_{2,t} = \mathcal{H}\left[asn \oplus (K_{1,2}+1)\right] \mod \left(R - l_{1,t}\right), \tag{2}$$

where asn has the same value as in Eq. 1, while $l_{1,t}$ refers to the delay previously extracted. Given that, by definition, $l_{1,t}+l_{2,t} \leq R$, any acknowledgement will be processed within the time-slot duration T, guaranteeing the delivery of a packet within an IEEE 802.15.4 time-slot.

Algorithm 1 and Algorithm 2 provide the pseudo-code of the operations of a transmitter and a receiver IoT device running RTSCA, respectively.

Considerations from the adversary point of view. With our countermeasure in place, the adversary described in Section III-B could not predict precisely the time instant when a target node has the radio enabled in RX mode, for receiving either a data packet or an acknowledgement. Indeed, to inject its bogus packet, the adversary should be able to *guess* the fraction of time between the activation of the radio of the target

Input: New Slot with ASN asn_n starts;

Prepare packet to transmit;
Compute delay $l_{1,t}$ as in Eq. 1; Wait a delay $l_{1,t}$;
Open the radio in TX mode and transmit packet;
if *packet is unicast* **then**
    Wait for ack preparation;
    Compute delay $l_{2,t}$ as in Eq. 2; Wait a delay $l_{2,t}$;
    Open the radio in RX mode and receive ack packet;
    Process ack packet;
end
Wait End Slot;

Algorithm 1: Pseudo-code of a Transmitter IoT device running RTSCA.

Input: New Slot with ASN asn_n starts;

Wait for packet preparation;
Compute delay $l_{1,t}$ as in Eq. 1; Wait a delay $l_{1,t}$;
Open the radio in RX mode and receive packet;
if *packet is unicast* **then**
    Prepare ack packet;
    Compute delay $l_{2,t}$ as in Eq. 2; Wait a delay $l_{2,t}$;
    Open the radio in TX mode and transmit ack packet;
    Wait for ack processing;
end
Wait End Slot;

Algorithm 2: Pseudo-code of a Receiver IoT device running RTSCA.

in RX mode and the time where the legitimate transmitter starts injecting its legitimate packet, that typically lasts for a short time T_p. Let us assume a receiver locks on a particular packet. In that case, it could not lock to any other packet in the same slot, nullifying the effects of the injection of the bogus packet by the adversary. Overall, assuming a maximum delay R, the adversary would be forced to make random guessing, with an average success probability $p_s = \frac{T_p}{R}$. Based on a target success probability $p_{s,MAX}$, the legitimate IoT devices can establish whether to keep the slot duration to the default value of $T = 10$ ms or to increase it, in a way to meet the requirement $\frac{T_p}{R} \leq p_{s,MAX}$. The increase of the slot duration is always allowed by the IEEE 802.15.4 standard, though it has to be traded off with a corresponding reduction of the network throughput.

On the adversary side, a more effective strategy would be to act as a *reactive jammer*, i.e., to listen for incoming transmissions on the wireless channel and, as soon as the SFD of a packet is received, start injecting its packet. A legitimate packet that is reactively jammed still generates reception of bytes on a target IoT device. However, including an erroneous CRC, this message will be discarded. Therefore, a reactive jammer can be conceived as a specific form of energy depletion attack. Note that, although using the same features, this is a different adversary model than the one in Section III-B, requiring the adversary to have a dedicated (and more expensive) setup, necessary to switch (within a very short time-delay) from the receiving mode (RX) to a high-power transmission mode (TX). In addition, note that the effect of a *reactive jammer* on the energy expenditure of the target is different than a *traditional* energy depletion attack. Indeed, the adversary cannot control the size of a *jammed* packet, thus not directly controlling the amount of energy spent by the target. We will evaluate all these aspects quantitatively in Section VI.

C. Detecting TaP Attacks

In this section, we describe a method that allows detecting the TaP energy depletion attacks introduced in [6].

The detection strategy is triggered by the reception of a packet with an erroneous CRC. We recall that an adversary realizing TaP energy depletion attacks injects packets consisting of a single MAC header byte containing the value 127, causing the receiver to receive 127 bytes (of noise). However, given that the adversary has no control over the bytes decoded by the target, the CRC of such a malicious packet will be erroneous with overwhelming probability.

Note that an erroneous CRC does not directly imply a TaP attack. Indeed, the CRC can be wrong also because of non-malicious interference. However, our observation, based on real experiments (see Section VI), is that *interferenced* packets usually present irregular patterns of erroneous bytes within the packet itself, while TaP packets present a trail of consecutive incorrect bytes, making them distinguishable from interference.

Therefore, as described through pseudo-code in Alg. 3, the proposed detection logic discriminates a TaP attack from interference based on error patterns within the packet. In particular, we focus the attention on the bytes that have a set of expected values, where it is possible to detect inconsistent values (e.g., the *FrameType* field can have only values from 0 to 3, and other values are inconsistent and lead to frame discard). If a byte has a consistent value, we mark this byte as a 0, while a 1 denotes an inconsistent value. We introduce the notion of *transition* in the bytes error patterns, defined as the consecutive presence of two different error marks, either 0 followed by a 1 or 1 followed by a 0. In brief, if a packet contains either a high number of errors or a high number of *transitions* (higher than predefined thresholds B and τ, respectively), it is marked as a TaP attack. Otherwise, it is marked as an interference.

Following detection, the target can take different countermeasures. For instance, it can decide to stop packet reception on the particular channel or time-slot where detection occurred, preventing the adversary from draining the battery further. In any case, through detection, the target becomes aware of the presence of the adversary, compromising the stealthiness objective explained in Section III-B. We discuss the calibration of the thresholds B and τ and the degree of effectiveness of our countermeasures n Section VI.

Input: A packet with an erroneous CRC is received;

Parse the MAC header of the received packet;
Evaluate each byte as erroneous (1) or correct (0);
Count the number of erroneous bytes e_b in the MAC header ;
Count the number of bytes transitions tr;
if $e_b \geq B$ **then**
    Mark the packet as a TaP attack;
    Stop reception on this frequency and time-slot;
else if $tr \leq \tau$ **then**
    Mark the packet as a TaP attack;
    Stop reception on this frequency and time-slot;
else
    Mark the packet as an interference;
    Continue receiving on this frequency and time-slot;
end

Algorithm 3: Pseudo-code of the TaP Detection Strategy in RTSCA.

Considerations from the adversary point of view. With the deployment of the detection strategy above-described, an adversary carrying out a TaP attack as in [6] could be immediately detected, losing chances to deplete the energy of the target. If the adversary $\mathcal{A}$ would like to avoid detection, it could increase the number of bytes sent in a TaP packet. In brief, instead of delivering only a single byte, the adversary can deliver more bytes, in a way to reduce the number of erroneous fields at the receiver and avoid detection. Given that we assume IoT nodes connected at the MAC-layer, to avoid detection, the adversary should modify the logic of the TaP attack by transmitting at least the PHY header (1 byte) and (part of) the MAC header (26 bytes). As experimentally shown in Section VI, this strategy decreases the energy gain of the TaP attack on the adversary side.

VI. Performance Evaluation

In this section, we provide the results of an experimental performance campaign verifying the effectiveness of RTSCA. Section VI-A describes our experimental setup, while Section VI-B reports experimental results.

A. Experimental Setup

To provide further insights on the performance of RTSCA, we set up an experimental testbed.

We used two Openmote-b devices as the legitimate IoT nodes. The Openmote-b hardware platform is among the state-of-the-art hardware boards for real experimentation and rapid prototyping of IoT algorithms and solutions [18] [19]. It integrates the CC2538 SoC by Texas Instruments, a ROM of 512 kB and a RAM of 32 kB, and it can transmit and receive wireless packets according to the latest IEEE 802.15.4-2015 specification. The Openmote-b devices run the OpenWSN protocol stack, integrating the time-slotted channel access mechanism of the IEEE 802.15.4-2015 specifications [20]. Conversely, we implemented the adversary $\mathcal{A}$ through a USRP Ettus Research X310 SDR, integrated with a daughterboard UBX160 [21]. As for the software, we used the GNURadio development toolkit, extending a reference implementation of the IEEE 802.15.4 standard in a way to achieve the described energy depletion attacks [22], [23].

Moreover, we measured the energy consumption of the devices using an oscilloscope Keysight InfiniVision DSOX2012A, providing two input channels and a resolution bandwidth of 100 MHz, by sampling the voltage drop to the terminals of a 1Ω probe resistor, bridging the pins in series with the CC2538 chipset. Finally, we used Matlab R2020b for running simulations on the estimated energy consumption of all the devices and adversary models assumed in this work.

B. Assessment

In the following, we report some results on the effectiveness of the countermeasure and TaP attack detection strategy included in RTSCA. We first report some results on the TaP attack detection, and then focus on the countermeasure applying to any energy depletion attack.

TaP Attack Detection. To investigate the effectiveness of the detection strategy discussed in Section V-C, we ran some real experiments, by testing both interference and TaP attack scenarios. To generate interference, we used two transmitting Ettus Research X310 SDRs, each one emitting packets with a very high rate (1 every 100 ms), so to generate packets collisions. Note that such a setup aims to reproduce a realistic scenario, where interferences and errors occur due to simultaneous radio operations. In addition, we replicated and extended the TaP attack proposed in [6], by configuring $\mathcal{A}$ to transmit an increasing number of bytes in the MAC header, from 1 to 27 bytes, while setting the value of the PHY header always to the MTU, i.e., 127, so to trigger the reception of a packet of the MTU size. Then, in line with our detection strategy, after the occurrence of the first erroneous byte, we evaluated the number of erroneous bytes e_b and *transitions* tr. As we can see from Figure 5—showing average results over 300 packets for each scenario—interference scenarios are mostly characterized by a relatively high number of transitions, while TaP attack scenarios usually generate 0 transitions or 1, at most. At the same time, relatively high number of errors undoubtedly refer to TaP attack scenarios where $\mathcal{A}$ transmit only a few bytes.

Overall, by setting the detection thresholds to $B = 13$ and $\tau = 1$, we can distinguish attacks from interferences up to the cases of TaP attacks with 26 bytes in 98.3% of the cases, with a false-positive rate of 0.11. To lower the false-positive rate, the network administrator can decide in advance the expected values of some other bytes, in a way to force the adversary to really transmit them. However, we cannot identify a TaP attack modified to inject at least 27 bytes, as in this scenario there are no errors or transitions to count (we assumed that only errors in the bytes of the MAC header can be detected, as described in Section III). Overall, the deployment of this detection method increases the energy required by $\mathcal{A}$ to carry out stealthy energy depletion attacks (more details below).

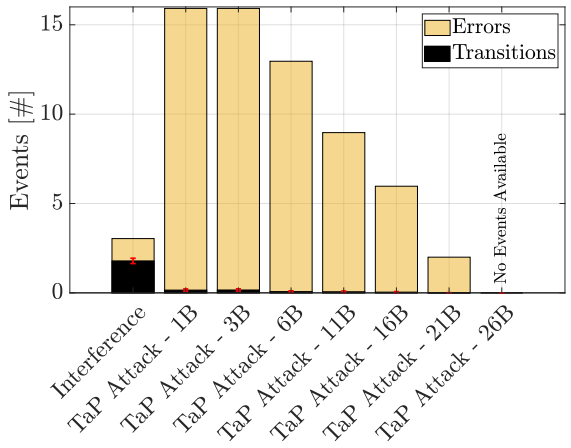

Fig. 5. Average number of errors and transitions with interference and TaP attack scenarios with increasing number of bytes emitted by $\mathcal{A}$. Each bar is the average of 300 tests, while red lines indicate 95% confidence intervals.

Random Time-Slotted Channel Access. To characterize the energy consumption of all the involved entities, we first measured the time and the energy spent by the Openmote-b IoT devices to execute the most energy-consuming operations within a IEEE 802.15.4 time-slot. The corresponding measured values are reported in Table I, with reference to the notation introduced in Figure 2 and a default time-slot of $T = 10$ ms.

TABLE I
DURATION AND ENERGY CONSUMPTION OF ENERGY-CONSUMING OPERATIONS IN AN IEEE 802.15.4 TIME-SLOT OF $T = 10$ MS.

Notation	Time [ms]	Energy [mJ]
TX RAMP UP	0.8125	73.66
TX DATA (127 bytes)	3.7746	465.69
RX RAMP UP	1.375	105.17
RX DATA (127 bytes)	3.7746	347.94
WAIT END SLOT	2.26	90.46

We also evaluated the minimum time T_p that can be set up on a receiving Openmote-b IoT device without affecting communication performance with a transmitting device. We recall that T_p is the time between the power on of the radio on the RX device and the reception of a packet from the transmitter TX device. While the default value for the OpenWSN protocol stack is $T_p = 1$ ms, we were able to reduce it to $T_p = 0.24$ ms without affecting network throughput. With such a value of T_p, assuming a slot duration of 10 ms and the value of R indicated in Table I, the probability of a successful energy depletion attack is $p_s = \frac{T_p}{R} = 0.108$. Such a success probability for $\mathcal{A}$ can be decreased as desired, by increasing the time-slot duration. For instance, $p_s = 0.019$ and $p_s = 0.011$ when $R = 20$ ms and $R = 40$ ms, respectively. By using the inverse relationship $R = \frac{T_p}{p_s}$, it is possible to obtain the value of R necessary to achieve the desired p_s.

Using the values in Table I, we ran some simulations in Matlab, to measure the average energy consumption of the actors in our scenario. Specifically, we define the following actors.

- TX: it is a legitimate IoT device transmitting a IEEE 802.15.4 data packet.
- RX: it is a legitimate IoT device receiving a IEEE 802.15.4 data packet.
- $\mathcal{A}_1$: it is the adversary defined in [6], running a TaP attack by delivering a packet with a MAC payload of 0 byte.
- $\mathcal{A}_2$: it is a reactive jammer, listening for an incoming packet and then running the TaP attack described in [6].
- $\mathcal{A}_3$: it is a reactive jammer, listening for an incoming packet and then running a TaP attack, by delivering a packet with a MAC payload of 26 bytes, in a way to avoid with 100% probability our detection strategy.
- $\mathcal{A}_4$: it is a reactive jammer, listening for an incoming packet and then injecting a standard IEEE 802.15.4 packet with a payload of 127 bytes.

We set up $10,000$ simulations each for the broadcast and unicast scenarios, where we varied the data (and ack) delay value introduced by RTSCA and the size of the exchanged packets. The energy consumption of all the entities above-described in a broadcast and unicast setup is shown (with confidence intervals) in Figure 6 and Figure 7, respectively.

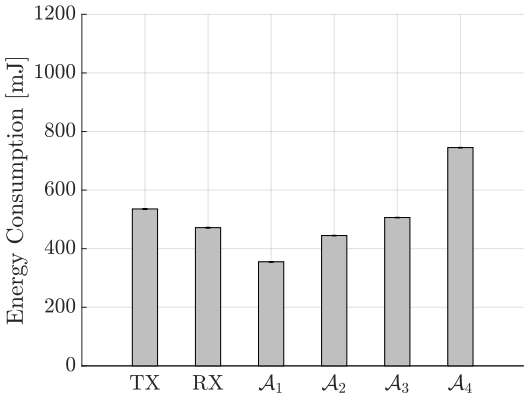

Fig. 6. Energy consumption of Transmitter, Receiver, and Adversary (in several configurations) after the integration of our strategies, assuming the delivery of a broadcast packet in a time-slot of $T = 10$ ms.

We first highlight that, independently from the scenario, the energy consumption of the legitimate TX and RX devices does not change when running our countermeasure and detection logic. Indeed, as highlighted in Section V, the duration of the radio activities is the same, and also the generation of the random delay can be pre-computed, outside the current slot. Therefore, the consumption of the legitimate IoT devices do not change after the integration of our schemes. Specifically, the TX and RX devices consume on average 535.7 mJ and 471.8 mJ in the broadcast scenario and 794.4 mJ and 741.2 mJ in the unicast scenario, respectively.

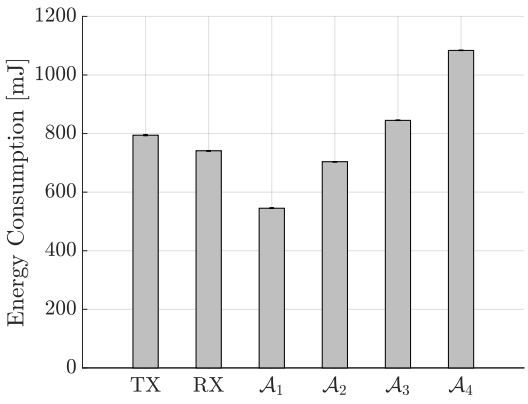

Fig. 7. Energy consumption of Transmitter, Receiver, and Adversary (in several configurations) after the integration of our strategies, assuming the delivery of a unicast packet in a time-slot of $T = 10$ ms.

$\mathcal{A}_1$ is the adversary that consumes the least energy. Indeed, when our countermeasure and detection are not in place, such an adversary can successfully inject bogus packets requiring only transmitting a packet of overall 6 bytes (4 bytes for the preamble, 1 byte for the SFD, and 1 byte for the PHY header), spending 355.3 mJ with a broadcast packet and 545.1 mJ with a unicast packet, respectively. We remark that when RTSCA is in place, the attack strategy of $\mathcal{A}_1$ has no success. Therefore, to increase its success probability, the adversary has to switch to a reactive jammer model, i.e., the one of $\mathcal{A}_2$. With this model, the energy consumption of the adversary rises to 444.8 mJ (broadcast) and 703.9 mJ (unicast), on average. Note that, with RTSCA in place, the attack strategy of $\mathcal{A}_2$ is characterized by a very low success probability, and also, when successful, it gets immediately detected by the target. Therefore, even if successful on a single attempt, it could not proceed further to exhaust the energy of the target (e.g., the target would stop radio operations on that time-slot).

To avoid detection while still having chances of success, the adversary has to behave as a reactive jammer, and to inject a packet containing at least a MAC header of 26 bytes. This strategy leads to an energy consumption of 506.3 mJ (broadcast) and 845.1 mJ (unicast) on average, that is 42.5% (broadcast) and 55% (unicast) more than $\mathcal{A}_1$, respectively. On the adversary side, note that the strategy $\mathcal{A}_3$—i.e., modifying the TaP attack with the reactive jammer behavior and the counter-detection logic—is still more convenient than injecting a full-size packet of 127 bytes, as modelled by the adversarial strategy $\mathcal{A}_4$. Indeed, using the strategy $\mathcal{A}_3$ the adversary could reduce its energy consumption by 47.17% in a broadcast time-slot and by 28.27% in a unicast time-slot, respectively, still potentially causing the same damage to the target.

VII. Conclusion and Future Work

In this paper we have presented RTSCA, a solution to prevent and mitigate energy depletion attacks in IoT networks based on the IEEE 802.15.4-2015 PHY/MAC standard. RTSCA randomizes radio activities execution time within an IEEE 802.15.4 time-slot, hence efficiently and effectively preventing an external adversary from guessing the time when a target can receive bogus packets—a key feature to carry out any energy depletion attacks. RTSCA also detects recently-disclosed TaP attacks via a pure-software solution, based on evaluating error patterns within received packets. RTSCA is a lightweight, energy-friendly, and flexible solution, requiring little-to-none additional energy on IoT devices. Moreover, RTSCA can be flexibly configured to increase robustness, requiring only a corresponding increase in the time-slot duration.

Our experimental performance assessment on real Openmote-b IoT devices showed that RTSCA reduces the effectiveness of traditional energy depletion attacks to at least 10.8% of their nominal effectiveness. To still have chances of success, the adversary has to deploy reactive jamming attacks, requiring on average at least 42.5% of energy more than the traditional setup.

Future work include the extension of RTSCA with capabilities to mitigate reactive jamming attacks, in a way to enhance the protection against any energy depletion attack further.

Acknowledgments

The authors would like to thank the anonymous reviewers, that helped improving the quality of the paper. This publication was partially supported by awards NPRP-S-11-0109-180242 from the QNRF-Qatar National Research Fund, a member of The Qatar Foundation. This work has been partially supported also by the INTERSCT project, Grant No. NWA.1162.18.301, funded by Netherlands Organisation for Scientific Research (NWO). The findings reported herein are solely responsibility of the authors.

References

[1] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, "Demystifying IoT Security: An Exhaustive Survey on IoT Vulnerabilities and a First Empirical Look on Internet-Scale IoT Exploitations," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2702–2733, 2019.

[2] V.-L. Nguyen, P.-C. Lin, and R.-H. Hwang, "Energy depletion attacks in low power wireless networks," *IEEE Access*, vol. 7, pp. 51 915–51 932, 2019.

[3] V. Desnitsky and I. Kotenko, "Modeling and analysis of IoT energy resource exhaustion attacks," in *International Symposium on Intelligent and Distributed Computing*. Springer, 2017, pp. 263–270.

[4] S. Sciancalepore, G. Oligeri, and R. Di Pietro, "Strength of crowd (SOC)—defeating a reactive jammer in IoT with decoy messages," *Sensors*, vol. 18, no. 10, p. 3492, 2018.

[5] X. Cao, D. M. Shila, Y. Cheng, Z. Yang, Y. Zhou, and J. Chen, "Ghost-in-zigbee: Energy depletion attack on zigbee-based wireless networks," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 816–829, 2016.

[6] S. Gvozdenovic, J. K. Becker, J. Mikulskis, and D. Starobinski, "Truncate after Preamble: PHY-Based Starvation Attacks on IoT Networks," in *Proc. of ACM Conf. on Security and Privacy in Wireless and Mobile Netw.*, 2020, p. 89–98.

[7] K.-F. Krentz and C. Meinel, "Denial-of-sleep defenses for IEEE 802.15.4 coordinated sampled listening (CSL)," *Computer Networks*, vol. 148, pp. 60–71, 2019.

[8] H. Pirayesh and H. Zeng, "Jamming Attacks and Anti-Jamming Strategies in Wireless Networks: A Comprehensive Survey," *arXiv preprint arXiv:2101.00292*, 2021.

[9] G. Chen and W. Dong, "Reactive Jamming and Attack Mitigation over Cross-Technology Communication Links," *ACM Trans. Sen. Netw.*, vol. 17, no. 1, pp. 1–25, 2020.
[10] M. Strasser, B. Danev, and S. Čapkun, "Detection of Reactive Jamming in Sensor Networks," *ACM Trans. Sen. Netw.*, vol. 7, no. 2, Sep 2010.
[11] E. Y. Vasserman and N. Hopper, "Vampire attacks: Draining life from wireless ad hoc sensor networks," *IEEE Transactions on Mobile Computing*, vol. 12, no. 2, pp. 318–332, 2011.
[12] "IEEE Standard for Low-Rate Wireless Networks," *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)*, pp. 1–709, Apr. 2016.
[13] E. Faulkner, Z. Yun, S. Zhou, Z. Shi, S. Han, and G. B. Giannakis, "An advanced gnu radio receiver of ieee 802.15.4 oqpsk physical layer," *IEEE Internet of Things J.*, pp. 1–1, 2021.
[14] P. Tedeschi, S. Sciancalepore, A. Eliyan, and R. Di Pietro, "LiKe: Lightweight certificateless key agreement for secure IoT communications," *IEEE Internet of Things J.*, vol. 7, no. 1, pp. 621–638, 2019.
[15] S. Sciancalepore, G. Oligeri, G. Piro, G. Boggia, and R. Di Pietro, "EXCHANge: Securing IoT via channel anonymity," *Computer Communications*, vol. 134, pp. 14–29, 2019.
[16] S. Sciancalepore, M. Vučinić, G. Piro, G. Boggia, and T. Watteyne, "Link-layer security in TSCH networks: effect on slot duration," *Trans. on Emerging Telecommun. Technol.*, vol. 28, no. 1, pp. 1–14, 2017.
[17] S. Sciancalepore and R. Di Pietro, "PPRQ: Privacy-Preserving MAX/MIN Range Queries in IoT Networks," *IEEE Internet of Things Journal*, 2020.
[18] X. Vilajosana, P. Tuset, T. Watteyne, and K. Pister, "OpenMote: Open-Source Prototyping Platform for the Industrial IoT," in *Int. Conf. on Ad Hoc Networks*. Springer, 2015, pp. 211–222.
[19] S. Sciancalepore and R. Di Pietro, "Bittransfer: Mitigating Reactive Jamming in Electronic Warfare Scenarios," *IEEE Access*, vol. 7, pp. 156 175–156 190, 2019.
[20] T. Watteyne, X. Vilajosana, B. Kerkez, et al., "OpenWSN: a standards-based low-power wireless development environment," *Trans. on Emerging Telecommunications Technologies*, vol. 23, no. 5, pp. 480–493, 2012.
[21] Ettus Research, "UBX160 Daughterboard," https://www.ettus.com/product/details/UBX160, 2020, (Accessed: 2021-04-16).
[22] G. Baldini, et al., "Security aspects in software defined radio and cognitive radio networks: A survey and a way ahead," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 2, pp. 355–379, 2011.
[23] B. Bloessl, C. Leitner, F. Dressler, and C. Sommer, "A gnu radio-based ieee 802.15. 4 testbed," *12. GI/ITG KuVS Fachgespräch Drahtlose Sensornetze (FGSN 2013)*, pp. 37–40, 2013.