IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 17, 2019, accepted October 22, 2019, date of publication October 25, 2019, date of current version November 6, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2949716

Bittransfer: Mitigating Reactive Jamming in Electronic Warfare Scenarios

SAVIO SCIANCALEPORE AND ROBERTO DI PIETRO
Division of Information and Computing Technology (ICT), College of Science and Engineering (CSE), Hamad Bin Khalifa University (HBKU), Doha, Qatar
Corresponding author: Savio Sciancalepore (ssciancalepore@hbku.edu.qa)

This work was supported in part by the QNRF-Qatar National Research Fund, a member of the Qatar Foundation, under Award NPRP11S-0109-180242, Award UREP23-065-1-014, and Award NPRP X-063-1-014.

ABSTRACT Electronic Warfare (EW) scenarios contemplate powerful and stealthy jamming attacks, able to disrupt any competing wireless communication in the target area. Reactive jamming techniques are especially suitable to this aim. Indeed, by first eavesdropping on the whole radio spectrum used for communications, and then timely injecting random noise as soon as a transmission is detected, reactive jamming represents both an effective and hard-to-detect attack tool. In such a challenging EW scenario, all the solutions currently available in the literature to mitigate reactive jamming require either the deployment of specialized hardware, or the modifications of physical layer protocols—the former solution being expensive, and the latter one usually not viable when considering commercially available wireless devices. In this paper we propose BitTransfer, an anti-jamming protocol enabling wireless communications between neighboring devices even under the above-described stringent requirements and powerful attacker model. BitTransfer embeds information bits in radio activity operations, a 0 being represented by the absence of any radio activity, and a 1 by the reception of a (corrupted) packet at the receiver. To demonstrate its applicability to a wide class of commercial wireless devices, BitTransfer has been implemented using a real constrained hardware platform (the Openmote-b), released as freely available and open-source, and tested using the IEEE 802.15.4 communication technology, adopted within the Bluetooth and Zigbee 3.0 protocol stacks. When under attack by a reactive jammer, BitTransfer can transfer a message of 127 Bit in 11.17 seconds, while competing approaches simply fail. Other than being completely tunable, BitTransfer can also enjoy further improvements by simply increasing the transmission rate of the devices. Finally, its detailed design, open-source availability, robustness, and superior performance when compared against competing solutions, make it a solution of choice in challenging EW scenarios, also paving the way to further research along the highlighted directions.

INDEX TERMS Jamming, electronic warfare, communication system security, wireless communication, anti-jamming protocols, cyber-physical systems security.

I. INTRODUCTION

Electronic Warfare (EW) scenarios involve a variety of powerful attacks against wired and wireless networks, where attackers use any meaningful tool to disrupt the operation of the communication infrastructure of the competing entities in the target area [1]. The increasing advancements in manufacturing and embedding technologies experienced in the last decade have boosted the effectiveness of EW systems and strategies, especially in the military application domain [2]. EW systems inspect thoroughly the electromagnetic environment in the target area, analyze the communication technologies used by the competing party, and design ad-hoc stealthy methods to shut them down and reduce organization and response capabilities of the contenders [3], [4].

In this context, *jamming* is still the most powerful and convenient Denial of Service (DoS) attack that can be performed to disrupt wireless communications [5]. By simply deploying a single device emitting noise at high power on the same channel used for ongoing wireless communications, any radio operation is disrupted, independently from the selected communication technology [6]. In addition, the commercial diffusion of low-cost and low-effort Software Defined Radios (SDRs) has further lowered the technological barrier necessary to launch jamming attacks [7].

Between the large number of jamming attacks that can be achieved, *Reactive Jamming* is the most effective and difficult to detect [8]. A reactive jammer remains passive for

The associate editor coordinating the review of this manuscript and approving it for publication was Wen Chen.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see http://creativecommons.org/licenses/by/4.0/

most of the time, listening to the wireless communication channel for incoming radio operations. As soon as the ramp-up of a radio operation is detected (typically, the preamble of a packet), it switches to the transmission mode and starts injecting noise on the communication channel, disrupting the correct reception of the packet [9]. Thus, any receiver will not be able to verify the Cyclic Redundancy Code (CRC) of the message, usually inserted in any packet to allow for the verification of the integrity of the information, and the packet will be discarded. Such a jamming model is, at the same time, very effective, difficult to detect, and energy-efficient on the attacker side [10].

A few contributions in the literature provided useful techniques to mitigate reactive jamming (see Sec. II for a comprehensive overview). However, these solutions usually assume weak adversary models, characterized either by limited access to the physical topology of the network, or by limited eavesdropping or jamming capabilities. Thus, these valuable approaches are not effective when the adversary is powerful enough to be distributed across the network and to disrupt communication on any channel or frequency used by legitimate devices—such as in EW scenarios. In addition, while a few works considered such a powerful adversary model, the adoption of the proposed countermeasures requires either modifications of the physical layer of the involved devices, or the access to the raw signals on the wireless channel, achievable only using very specialized hardware, such as the commercially available SDRs. Thus, they are not applicable in a typical EW scenario, where the devices are already deployed, and cannot be accessed or replaced by the operating entities.

A. CONTRIBUTION

In this paper, we propose BitTransfer, an anti reactive-jamming protocol able to guarantee wireless communications in a distributed network even under a typical EW attack scenario, where the adversary is a powerful reactive jammer, able to disrupt any communication in the network independently from its origin location and used frequency. To enable communication, BitTransfer takes advantage of the logic of the jamming, translating a silent time-slot (i.e., the absence of any radio operation) into valuable and meaningful information. Compared with previous work on reactive jamming mitigation, to the best of our knowledge, BitTransfer is the first solution able to guarantee the delivery of the message under such a strong adversary model, without requiring to either access or modify the hardware already deployed— a typical requirement in EW scenarios.

To demonstrate the wide audience of cyber-physical wireless devices enjoying compatibility with BitTransfer, we implemented the proposed protocol on real constrained Internet of Things (IoT) devices, i.e., the OpenMote-b hardware platform, and we tested its performance under a variety of system parameters using the widespread OpenWSN protocol stack. For instance, BitTransfer enables the transmission of a 127 B message within 1,117 time-slots, being equal to 11.17 seconds using the default configuration of the OpenWSN protocol stack— this time can be further reduced by tuning the time-slots schedule.

Finally, the source code of our proof-of-concept has been released as open-source [11]. This could allow practitioners, industries, and academia to verify our claims and to compare their own solutions with BitTransfer, eventually using our source code as a ready-to-use basis for their software development.

B. ROADMAP

The rest of the paper is organized as follows: Section II provides an overview of related work; Section III introduces the scenario and the adversary model assumed in this work; Section IV details the proposed BitTransfer protocol; Section V provides the results of a real experimental evaluation conducted on real constrained devices; Section VI compares our solution with related work, and highlights its advantages and limitations; finally, Section VII tightens conclusions.

II. RELATED WORK

Jamming attacks have been investigated by a few works in the literature, mainly because of the low effort they require on the attacker side and their disruptive potential [5].

Several different jamming models have been studied, as discussed in the following.

- *Constant Jammers*. They are the easiest to be deployed, as they require only a Radio Frequency (RF) device continuously emitting random noise on a given set of frequencies, thus being not compliant to any wireless communication protocol. At the same time, they are also the easiest to be detected, and they require a significant energy budget on the attacker side [12].
- *Deceptive Jammers*. These jammers complicate the detection process by injecting (fake) packets compliant to given communication technology, instead of random noise. Even if they are slightly more difficult to be identified, they still require a significant energy budget on the attacker side [6].
- *Proactive Jammers*. These types of jammers select a subset A of the overall number of channels F used by legitimate devices for communications and disrupt only these channels. Even if they could be less effective than Constant and Deceptive jammers, they require less energy on the attacker side [13].
- *Reactive Jammers*. These powerful jammers continuously listen to the wireless communication channel for new wireless communications. As soon as the start of a new communication is detected, they disrupt it by injecting noise. They are very effective, hard to be detected, and require the least amount of energy on the attacker side.

Despite a few research papers recently focused on methods to boost network performances under proactive jamming attacks [14]–[18], the most contributions focused on

techniques to mitigate reactive jamming attacks, as they are the most challenging ones ([19]–[30]).

The authors in [19] and [20] introduced a technique to preserve communications even in the presence of a reactive jammer, by assuming that the adversary could disrupt the communications only in a pre-defined portion of the network. Thus, the adversary model they assume is spatially limited, and it assumes that the adversary could disrupt only a subset of the communication links available in the network.

The authors in [21] surveyed the most popular techniques to achieve jamming, as well as techniques to evade it. At the same time, they propose a strategy involving first the detection of the attack, and then the evasion by exploiting either frequencies or locations in the network where jamming is not effective. Alternatively, they propose a mechanism to escape jamming by tuning physical-layer parameters, including the communication coding. These latter strategies, however, cannot be applied in regular devices, and require special hardware such as SDRs.

The authors in [22] and [23] proposed to defeat reactive jamming by identifying the *trigger nodes*, i.e., the nodes whose transmission trigger the reactive jammers. Thus, they select the optimal routing path, to avoid such nodes. However, the adversary model they consider is spatially limited, and the decisions on the optimal routing paths are taken by a centralized Base Station, that could be subject to reactive jamming, too.

A very effective solution was introduced by the authors in [24], where the legitimate devices exploit the reaction time of the jammers to communicate. However, the proposed method requires a special encoding of the I/Q bits at the physical layer of the wireless communication, and thus it would require either changing their physical layer, or replacing them with specialized SDRs.

The authors in [25] presented *DEEJAM*, a novel MAC-layer protocol for defeating reactive jammers with IEEE 802.15.4-based hardware. It layers four defensive mechanisms to hide communication from a jammer, to evade its search, and to reduce its impact. Despite representing a valuable solution, *DEEJAM* either assumes an adversary with limited capabilities, not able to eavesdrop and jam any time/frequency channel used by legitimate devices, or it recommends to change the Start of Frame Delimiter (SFD) of the IEEE 802.15.4 packets, thus not being applicable in most commercial devices.

The authors in [26] and [27] proposed Friendly Crypto-Jam (FCJ), a scheme that can avoid reactive jamming by de-correlating the payload modulation scheme from other transmission attributes, embedding information symbols into the constellation map of the highest-order modulation scheme supported by the system. Similarly to other approaches discussed before, FCJ requires special hardware to be deployed, such as the SDRs. Thus, it cannot be adopted when the devices have been already deployed and cannot be accessed or replaced.

Another approach leveraging the reaction time of the jammer is *BitTrickle*, proposed by the authors in [28]. *BitTrickle* is an anti-jamming wireless communication scheme that encodes information in the preamble of the wireless communication, modifying its default structure at the expense of a higher error probability. Unfortunately, commercial devices only allow to shorten or increase the duration of the preamble sequence, but not to modify its content or store the received preamble. Thus, this approach requires specialized equipment such as SDRs to be deployed.

The authors in [29] introduced *Strength of Crowd (SoC)*, a distributed protocol that allows evading reactive jamming by transmitting decoy messages, exhausting the jamming resources of the attacker. However, the attacker model assumed in this work can jam only a fraction of the available spectrum. If the attacker can jam any channel used for legitimate communication, the approach would not be effective.

The authors in [30] proposed *Silence is Golden (SiG)*, a protocol that achieves communication between two neighboring devices under the same adversary model assumed in our work, i.e., a location-unbounded, global eavesdropper, and frequency-unlimited reactive jammer. To establish a communication channel, *SiG* is based on a mechanism to detect the presence of *energy* on the communication channel, independently from the presence of a radio packet. On the one hand, the effectiveness of such a scheme drastically decreases when the channel is noisy, leading to the detection of many false packets. On the other hand, estimating the energy of the channel without the presence of any packet requires specialized equipment, such as the SDRs. Thus, like many other approaches described above, it cannot be applied when the legitimate devices are already deployed and cannot be accessed— a typical situation in EW scenarios.

To summarize, we notice that, at the time of this writing, most of the available solutions to thwart reactive jamming assume *weak* adversary models, having limitations in their spatial distribution, transmission detection, jamming capabilities, or reaction time. When the limitations on the attackers are removed, the adopted solutions require the access to the raw signals (I/Q components) transmitted on the wireless channel, usually achievable either with specialized equipment, such as the commercially available SDR, or requiring costly hardware modifications. Thus, they cannot be adopted when the legitimate devices cannot be accessed or changed after the initial deployment, such as in most Cyber Physical Systems (CPSs) under *Electronic Warfare* attacks. More details on the relationship between our proposal and these system requirements are provided in Sec. VI.

III. SCENARIO AND ADVERSARY MODELS

In this section we introduce both the system and the adversary models assumed throughout our paper, as well as some basic preliminary assumptions.

A. SCENARIO

We consider a generic wireless network being part of a Cyber Physical System (CPS), constituted by N nodes, uniformly distributed in a generic area. Each node is equipped with a wireless radio, thus being able to transmit and receive packets in a spectrum of F frequencies. For instance, the network can be a generic IoT network, made up of constrained devices, organized hierarchically, able to sense data the surrounding environment, e.g. temperature, humidity, light, acceleration, pressure, movement, and to report data to the system administrator via wireless links. Alternatively, it can be a network made up of static or mobile wireless routers communicating according to IEEE 802.11 standards, or a network of connected vehicles. We also assume that each node in the network can produce new information, e.g., by sensing the surrounding environment. Then, the node wants to report the information to the central node in the hierarchy, connected to the Internet via a regular wired connection.

We assume that all the nodes are loosely time-synchronized, and that the wireless RF communications are scheduled on a time-slot basis, in line with modern CPS protocols such as IEEE 802.15.4 [31] and IEEE 802.11 [32]. At each time-slot, the node n_i, with $i \in \{1, \ldots, N\}$, can either transmit or receive, by tuning its radio on a selected frequency f_j, with $j \in \{1, \ldots, F\}$. We assume that the frequencies selected by neighboring nodes to communicate are accorded, and they have been established in advance through a dedicated scheduling algorithm (see, for instance, [33]–[35]). Thus, each node knows exactly the slot where it could send (receive) packets to (from) a neighboring device. In line with the previously mentioned standards for slotted wireless communications, we assume that the slot duration T is large enough for a given node to transmit a packet and to receive the corresponding acknowledgment from the receiver.

Finally, we assume that the network administrator only has physical access to the devices making up the network at the first deployment time. After the initial deployment, the network administrator cannot access the devices anymore, e.g., to modify their software or firmware, or to replace them with upgraded hardware components. In this context, our proposed BitTransfer protocol is intended to run on any device, and it involves two (or more) generic neighboring devices, within the same RF transmission range. More details will be provided in Sec. IV.

The notation used in the sequel of the paper is reported in Tab. 1.

TABLE 1. Notation used throughout the paper.

Notation	Description
N	Number of nodes in the network.
L	Messages to be transmitted, encoded with a Forward Error Correction (FEC) code.
L_i	Generic i-th bit of the message L.
M	Number of bits in a message to be transmitted.
B	Maximum of bits in a message to be transmitted.
T	Time-slot duration.
T_i	Generic i-th time-slot
L'_i	Local copy of the message L on the receivers.
t_M	Time needed to transmit a message with M bits.
F	Number of frequencies in the radio spectrum.
t_{JD}	Duration of the jamming operation.
A	Number of jammed frequencies by the adversary.
t_b	Time needed by legitimate devices to transmit a bit.
t_{JT}	Jamming triggering Time, i.e., the time needed by the adversary to switch from receiving to transmitting random noise.
t_{W_j}	Acknowledgment delay for the node j, i.e., number of time-slots to wait after the reception of the Message Exchange Start packet before sending an acknowledgement packet.
K	Number of time-slots with no message exchange to declare a jamming attack.
t_{r_j}	Random Backoff Time extracted by the node j.
t_R	Maximum Backoff Time.
t_S	Synchronization Delay, i.e., number of time-slots allocated for the synchronization phase.
G	Maximum number of unsuccessful synchronization phases.

B. ADVERSARY MODEL

In this paper, consistently with an EW scenario, we assume a very powerful adversary, namely $\mathcal{E}$, equipped with the following features:

- *Global Eavesdropper*. $\mathcal{E}$ can listen to ongoing wireless communications taking place in the network, independently from the particular used frequency.
- *Spatially Unlimited*. $\mathcal{E}$ can identify the presence of communication, independently from its distance from the transmitting source. In practice, $\mathcal{E}$ can achieve this property by deploying a large number of receiving antennas in the area under attack.
- *Frequency Unbounded Jammer*. $\mathcal{E}$ can achieve jamming by transmitting random noise on any frequency used by legitimate devices to communicate. Thus, we assume that $A \geq F$, i.e., the frequencies that $\mathcal{E}$ can disrupt are greater than the frequencies that the legitimate devices can use for their communications.
- *Reactive Jamming Capabilities*. Being stealthy, $\mathcal{E}$ waits for a meaningful packet to be transmitted on the wireless communication channel, and starts jamming as soon as the transmission of a packet is identified.
- *Technology Independent*. We assume that $\mathcal{E}$ starts jamming as soon as any wireless signal is detected on the wireless communication channel, independently from the selected communication technology.

- *Short Reaction Time*. As soon as $\mathcal{E}$ recognizes the presence of a transmission on the wireless communication channel, it starts jamming. We assume that the time needed by $\mathcal{E}$ to switch from the receiving state (RX) to the transmitting state (TX), i.e., the *Jamming Triggering Time*, is $t_{JT} \neq 0$, where $t_{JT} > 2\ t_b$, being t_b the time needed by legitimate devices to transmit a bit on the wireless channel.
- *Stealthy Jamming Duration*. We assume that the duration of the jamming, namely t_{JD}, is large enough to potentially corrupt any bit transmitted in the remaining part of the time-slot. Thus, assuming t_M is the maximum duration of a message M on the wireless channel, $t_{JD} \geq t_M$. In this way, as soon as the transmission of the legitimate message is finished, also the jamming activity is stopped, further contributing to the stealthiness of the adversary.

We remark that the proposed BitTransfer protocol is effective also when the adversary $\mathcal{E}$ is technology-aware, i.e., she knows in advance the communication technology used by legitimate devices to communicate, and waits for a signal compliant to this particular technology to start jamming. In this scenario, BitTransfer is valid also assuming $\mathcal{E}$ to be able to switch from RX to TX instantaneously, i.e., being $t_{JT} = 0$. However, we believe that the combination of the *Technology Independence* and *Short Reaction Time* features are more effective for the adversary, and more suitable for the modeling of an *Electronic Warfare* (EW) scenario.

IV. THE BITTRANSFER PROTOCOL AT GLANCE

In this section, we introduce the details of the proposed BitTransfer protocol, designed to enable wireless communications in the presence of a powerful reactive jammer in the *Electronic Warfare* scenario. Section IV-A introduces the logic of BitTransfer via a baseline example, while Section IV-B provides the description and rationale of its additional features. Finally, Section IV-C details the full BitTransfer protocol.

A. BASELINE EXAMPLE

First, we illustrate the rationale of the proposed BitTransfer protocol via a baseline example, whose sequence diagram is provided by Figure 1.

Our baseline example consists of two generic devices, namely A and B, featuring the same wireless communication technology to exchange messages. The communication link between the two devices is disrupted by a reactive jammer, following the model described in Sec. III-B. We also assume that A and B are willing to communicate, and to exchange a reference message $L = [1001]$, consisting of $M = 4$ bits.

At the detection of the jamming attack, the two devices launch a synchronization protocol to assign receiver and transmitter roles (see Sec. IV-B for more details). We hereby assume that the device A is the transmitter of the message, and B is the receiver.

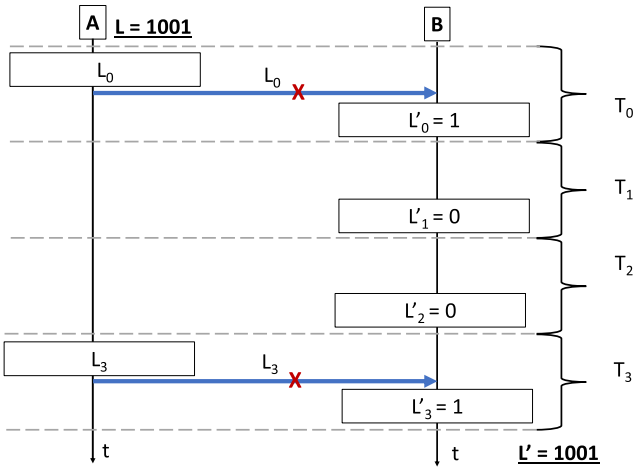

FIGURE 1. Baseline example of BitTransfer with a message $L = 1001$, being $M = 4$. The device A needs to transmit the message $L = [1, 0, 0, 1]$. In each time-slot, when the bit to be transmitted is a 1, the message L is transmitted; otherwise, there is no transmission. On the receiving side, if a packet is detected, despite being corrupted, a 1 is logged; otherwise, if no packet is detected, a 0 is logged.

At each time-slot T_i, if the bit of the packet to be transmitted is $L_i = 1$, the device A transmits a message, including L as the payload of the message. Otherwise, if the bit to be transmitted is $L_i = 0$, the device A does not transmit any packet. Thus, considering our baseline example shown in Fig. 1, the device A transmits the message L during the time-slots T_0 and T_3, while it does not transmit any message during the time-slots T_1 and T_2.

On the reception side, being the communication channel disrupted by the adversary, the device B receives an incorrect message, where some bits are poisoned by the adversary through the injection of random noise on the communication channel. Thus, the packet is discarded without storing its content. At the same time, having detected a packet on the communication channel, the device B logs a bit $L'_0 = 1$. During the time-slots T_1 and T_2, instead, the device B does not detect any ongoing transmission on the communication channel, and thus it logs two consecutive bits $L'_1 = 0$ and $L'_2 = 0$. Finally, during the time-slot T_3, the device B can detect the preamble of the message, and log a bit $L_3 = 1$. After exactly M time-slots, the device B is in possession of the full message $L' = L = [1, 0, 0, 1]$. To sum up, for each time-slot T_i, the BitTransfer protocol requires the transmission of the message in a packet when the bit to be delivered is $L_i = 1$, while it does not require any transmission when $L_i = 0$.

The above-described core logic of BitTransfer needs to be enriched with further features, to allow the devices to establish the roles of the transmitter and the receiver, to improve its robustness to random errors on the communication channel, and to minimize the probability of false jamming detection events. Further details are provided in Sec. IV-B.

B. ADDITIONAL FEATURES

The previous subsection detailed how a generic information message P of M bits can be delivered thanks to the

BitTransfer protocol. However, the basic scheme could not work without additional mechanisms, useful to detect a jamming attack, synchronize the devices to elect the transmitter and the receivers, and correct random errors on the communication channel. These strategies are detailed below.

1) JAMMING DETECTION

The initiation of the BitTransfer scheme is tied to the strategy to detect an ongoing jamming attack on the communication link between two neighboring devices. Jamming detection involves a few effective strategies, already discussed in the literature and outside the scope of the proposed protocol (see, for instance, [36], [37], and [38]). All these techniques are perfectly suitable for integration with BitTransfer. However, to simplify the discussion, we assume that the jamming activity is detected when no packets are successfully received by a given device for a specific time frame. Indeed, the number of time-slots to wait to declare a jamming attack should be selected trading off between the data reporting delay and the probability of false positives, i.e., erroneous jamming events detection.

On the one hand, BitTransfer should be launched as soon as the jamming starts, to minimize the delay in exchanging data. On the other hand, there could be situations where consecutive packets are lost due to a temporary interference, not originated on purpose to disrupt the communication. BitTransfer trades off efficiently between these two requirements by including the information message P as the payload of any transmission operation. Thus, in case packet corruption events are not caused by a malicious jammer, the receiver can successfully receive the message P and transmit an acknowledgment, indicating the successful reception. In this case, the execution of BitTransfer is aborted and the regular behavior of the devices is restored. It is worth noting that this feature is particularly useful also in other cases, e.g. when the jammer is proactive, and when the reactive jammer has not a 100% hit probability, i.e., it is not capable (or, it is not configured) to jam every packet.

2) DEVICES SYNCHRONIZATION

Immediately after the jamming detection, the communicating devices should elect which device is going to transmit a message and, consequently, which devices are going to receive it. To this aim, the devices execute a deterministic *Synchronization Protocol*, whose logic is illustrated via an example in Fig. 2.

Each device j extracts a random backoff time t_{r_j}, necessary to elect the transmitting and the receiving device for the current instance of the protocol, and then it immediately switches its radio in reception mode (RX). The extracted random number specifies the number of time-slots the device has to: (i) wait to transmit a given packet; and, (ii) listen to the communication channel for the start of a packet to be detected, i.e., for a preamble sequence. The logic is that the device extracting the smallest random backoff time will be the one to transmit its message.

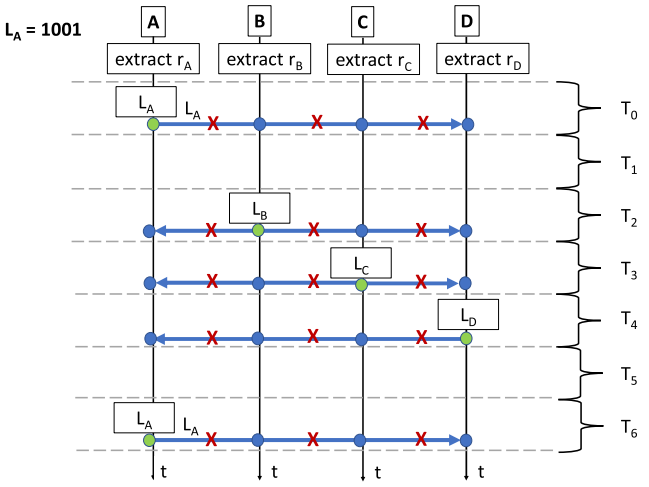

FIGURE 2. Example of the synchronization protocol of BitTransfer, with $N = 4$ devices and $S = 5$ time-slots. The device *A*, being the message transmitter, transmits a *Message Exchange Start* packet. The reception of this packet is acknowledged by each potential receiver in the following slots, by transmitting an *acknowledgment packet* after a fixed *acknowledgment Delay*. Finally, the transmitting device *A* informs all the devices about the correct completion of the synchronization protocol with a *Finalization Synchronization Packet*.

Let us assume that the device A extracts the smallest value, i.e., r_A. As soon as r_A expires, to notify this event to the other nodes, the device A transmits the message L_A in the payload of a *Message Exchange Start* packet. This packet is not meant to deliver a bit of the information message P (as described in Sec. IV-A, we recall that the adversary reactively jams any communication in the network), but only to indicate that one of the devices has won the contention, and it is ready to deliver its message. After the delivering of this message, the device A sets up its radio in *RX mode*.

We assume that each device j has been previously configured with a unique value t_{W_j}, namely the *acknowledgment Delay*. This delay indicates how many time-slots the node j has to wait after receiving a *Message Exchange Start* packet to acknowledge its reception, by sending a packet on the communication channel. We also define t_S as the *Synchronization Delay*, i.e., the number of time-slots allocated for the synchronization protocol.

For the case of Fig. 2, $W_A = 1$, $W_B = 2$, $W_C = 3$, $W_D = 4$, and $S = 5$. Thus, in the time-slot T_1, immediately following the *Message Exchange Start* packet, there is not any transmission, given that the device A is the one that transmitted the *Message Exchange Start* packet. Then, being $W_B = 2$, in the time-slot T_2, B transmits an *acknowledgment packet*, including a random payload L_B (note that the payload could be whatever, given that only the presence of the radio activity is necessary). In turn, the device C transmits the *acknowledgment packet* in the slot T_3 and the device D transmits the *acknowledgment packet* in the slot T_4. Given that $S = 5$, the time-slot T_5 will not contain any transmission, and it could be useful if a new node joins the network after its initial deployment. Finally, at the slot T_6, the device A evaluates the number of received *acknowledgment packets*. Being this

number equal to $N-1$, i.e., the number of neighbors in the radio range, A finalizes the synchronization protocol by sending a new message, namely, the *Finalization Synchronization Packet*, whose radio presence (despite the jamming) informs all the other devices that the synchronization phase ended successfully. From the following time-slot, the information message P, encoded in the message L_A, is delivered according to the scheme described in Sec. IV-A.

On the one hand, we highlight that the selection of the domain of the random number generator (used to establish the backoff period) is key to avoid collisions, i.e., the extraction of the same backoff period on multiple devices. To this aim, the maximum value t_R of the backoff time should be selected trading off between the number of nodes in the network and the synchronization delay t_S.

On the other hand, situations where two (or more) devices extract the same random period are possible. The described deterministic synchronization protocol allows detecting such situations. In fact, if two devices, namely A and C, extract the same backoff time $r_A = r_C$, their *Message Exchange Start* transmissions will collide. Thus, even assuming a neighboring device, namely B, clearly detects one of the packets and transmits the acknowledgment after W_B, neither A nor C will transmit it (they will remain in RX mode), and thus both the devices will realize that the transmission has not been successful. Therefore, being the number of detected acknowledgments less than the number of neighbors, no devices will transmit any *Finalization Synchronization Packet* message, and all the devices will restart a new instance of the synchronization protocol.

The synchronization delay t_S should be selected by the network administrator based on the physical topology and size of the network, as well as considering the dynamicity of the network. In fact, if N is the number of nodes in the same transmission range, the available slots $t_S - N$ could be allocated for new devices joining the network after its initial deployment. At the same time, if the connection of a device is intermittent (i.e., the node frequently joins and leaves the network), it will always have the same acknowledgment delay t_{W_j}, not requiring any new dynamic assignment. We highlight that the synchronization delay should be tuned at run-time, every time a new node joins or leaves the network, by increasing it or decreasing it by one. Given that joining and leave events usually happen when the adversary is not active, the synchronization delay can be easily updated to reflect the number of nodes in the network. When the adversary becomes active and starts its reactive jamming attack, the configuration of the synchronization delay is already optimized to guarantee the election of the transmitter and the acknowledgments by the receiver in the minimum necessary time, without wasting of time.

We also remark that run-time situations where the number of devices joining the network exceeds the number of available slots (that is $t_S - N$), where *slots* refers to the casting of the number of nodes N in an equal number of time slots) could be easily avoided. Some problems could appear if new devices try to join the network when the adversary is already active. However, given that reactive jamming corrupts any packet transmitted in the network, including beacon frames used by new devices to join the network, any new device trying to join the network will not be able to join. Therefore, the number of nodes in the network after the activation of the reactive jamming cannot further increase, nullifying the probability that the total number of devices in the network, namely N, exceeds the synchronization delay t_S.

Finally, we highlight that if a device leaves the network, the synchronization protocol cannot be completed successfully. In fact, this device will not emit the *acknowledgment packet*, and the number of received acknowledgments will be lower than the expected. To manage this situation, BitTransfer fixes a value G, indicating the maximum number of unsuccessful synchronization phases to be performed for a given instance of BitTransfer, having the same missing node. For instance, assuming $G = 3$, if the synchronization protocol fails for 3 consecutive times, always because of the missing of an acknowledgment packet from the same device j, BitTransfer will assume that the specific device has left the network, and thus the node emitting the *Message Exchange Start* packet will emit the *Finalization Synchronization Packet*, starting the delivering of the message according to the scheme described in Sec. IV-A.

3) ERROR CORRECTION

BitTransfer requires the delivering of an information message P from a transmitting to a receiving device bit-by-bit, where one bit is established during a single time-slot when the preamble of a packet is detected. It is worth noting that the bit-size of the preamble sequence used by legitimate devices should be selected considering both the protection against the reactive jammer and the error probability. On the one hand, the shortest the preamble sequence, the more the probability that the adversary could not disrupt it. In fact, being a reactive jammer, the adversary would have to decode the presence of ongoing transmission and switch from the RX to the TX mode before starting injecting noise on the communication channel. On the other hand, the bit-size of the preamble sequence has a direct impact on the reliability of the communication link. The longer the preamble sequence, the finer will be the synchronization between the transmitter and the receiver, and the minimum the probability to decode wrongly a bit in the packet, thus lowering the overall error probability. In addition, the longer the preamble sequence, the lower the probability to detect a false packet, i.e., a sequence of bits similar to the preamble only due to random fluctuations of the communication channel.

To reduce at minimum the probability that the adversary could disrupt the preamble of the message, BitTransfer optionally enables to reduce the size of the preamble to the minimum, i.e. 1 byte. Even if this choice breaks the compliance of BitTransfer to any standard, we highlight that most of the embedded devices, e.g., the ones produced by the popular Texas Instruments company, allow reducing the preamble to

such a short size, through the setup of a dedicated hardware register via firmware [39].

At the same time, this choice increases the probability of false positives, i.e., erroneous detection of packets that turn a bit value 0 into a bit value 1. To reduce the impact of these events on the final message, the original information message P is encoded with a FEC code. For instance, a repetition code of size n can be added to the message, increasing by a factor n its bit-size, and thus the time needed to deliver the message. However, for a bit to be flipped, at least $\lfloor \frac{n}{2} - 1 \rfloor$ consecutive false preambles should be detected—an even more unlikely event based on the selection of the repetition factor n. Indeed, the usage of a repetition code is just an example, and more efficient codes could be used to maximize the trade-off between error correction capabilities and the resulting bit-string size, such as turbo-codes and convolutional codes [40].

C. THE COMPLETE BITTRANSFER PROTOCOL

The pseudo-code of the complete BitTransfer protocol, for a generic TX and RX node, is reported in Algorithm 1.

Overall, BitTransfer consists of three different phases, namely Setup, Synchronization, and Message Exchange Phase.

1) SETUP PHASE

This phase is executed at the boot-up of the network, when the devices are initialized (lines 1-2 in Algorithm 1). Each device j is configured with a table, indicating the *Ack Delays* t_{W_j} for each device in the neighborhood, and it is crucial for the following phases.

2) SYNCHRONIZATION PHASE

This phase is triggered when jamming is detected (line 3). Each device extracts a random backoff time t_{r_j} (line 5), and sets up its radio in RX mode (line 6). The first device where the backoff period expires encodes a bit-string message P into a FEC string L, and transmits it over the wireless communication channel (lines 36–38). Being the channel disrupted by a reactive jammer consistent with the adversary model described in Sec. III-B, the receiving devices can detect the presence of a message by identifying the preamble sequence, but they are unable to decode it correctly. Thus, each receiving device waits for t_{W_j} time-slots and delivers an *acknowledgment Packet*, indicating the successful synchronization with the device transmitting the first message (lines 7–11). After t_S time-slots, the device transmitting the first message evaluates the number of received acknowledgment Packets: if they are equal to the number of neighboring devices, it passes to the following *Message Exchange Phase* (lines 39–43).

3) MESSAGE EXCHANGE PHASE

In this phase, the device that previously extracted the minimum backoff time starts transmitting the message L according to the scheme previously described in Sec. IV-A. In the first useful time-slot, it transmits a *Message Exchange Start* packet, having the message L included as a payload (line 38).

Algorithm 1 Pseudo-Code of BitTransfer

```
1  Setup Phase;
2  Store W = [t_W0, t_W1, ..., t_Wj, ..., t_WN] /* Ack Delays */
3  if no packet exchange for K time-slots then
       /* Jamming detected */
4      Synchronization Phase;
5      r_j ← [0, t_R] /* Extract random backoff time */
6      Switch the radio to RX mode;
7      Wait t_rj time-slots;
8      if a preamble is detected in t_rj time-slots then
           /* Message Reception mode */
9          Wait t_Wj time-slots;
10         Send an acknowledgment packet;
11         Wait t_S − t_Wj time-slots;
12         if Message Exchange Start is received then
13             Message Exchange Phase;
14             for i=1:B do
15                 if a preamble is detected then
16                     if CRC is not correct then
17                         L'_i = 1;
18                     end
19                     else
20                         L' received;
21                         L' = L /* Message
                               successfully received */
22
23                     end
24                 end
25                 else
26                     L'_i = 0;
27                 end
28             end
               /* Message successfully received */
29
30         end
31         else
               /* Two transmissions collided */
32             Extract a new random backoff time /* Restart
                   the protocol */
33         end
34     end
35     else
           /* Message Transmission Mode */
36         P ← [0, 1] /* Information Message */
37         L = FEC(P) /* Apply FEC */
38         Transmit L in a Message Exchange Start packet;
39         for j=1:S do
40             if a preamble is detected then
                   /* Log neighbors acks */
41                 count_ack = count_ack + 1;
42             end
43         end
44         if count_ack == N then
               /* All the nodes are ready */
45             Message Exchange Phase;
46             Send a Finalization Synchronization Packet in this slot;
47             for i=1:M do
48                 if L_i == 1 then
49                     Transmit L;
50                 end
51                 else
52                     nop; /* No transmission */
53                 end
54             end
               /* Message successfully transmitted
                   */
55
56         end
57     end
58 end
```

In this way, all the receiving devices can realize the successful completion of the previous synchronization phase and switch the radio in RX mode (lines 12-13).

In each following time-slot t_i, where $i \in [1, M]$, being M the number of bits in the message L, if the bit of the message $b_i = 1$, the transmitting device emits a message including the full message L in the payload, while if the bit of the message $b_i = 0$, it does not transmit and stay silent (lines 48–53). The receiving devices log 0 and 1 according to the same logic: thus, in each time-slot t_i, if a preamble is detected, $L'_i = 1$, otherwise $L'_i = 0$ (lines 14–28). Note that the number of slots where the RX devices turn their radio on to listen to incoming transmissions, i.e., the size of the information message, can be determined by the receiving devices using a strategy where the first byte (8 bits) of the information message defines the length of the message, thus informing also the receiver about the number of time-slots where it has to turn its radio on to the evaluate the presence of a packet.

If the adversary can jam every packet, after exactly B time-slots the receiving devices receive the message L—it will be padded with zeros (line 29). Otherwise, if any of the transmissions is not disrupted by the adversary, i.e., the CRC embedded in one of the messages is correctly verified, the message L is immediately delivered (line 21).

It is worth noting that this latter feature enables boosting the performance of BitTransfer when the adversary follows a *proactive* behavior, i.e., it disrupts only a portion A of the communication channels F available for legitimate communications. Indeed, a single *clear* slot is necessary to transfer the whole message L.

V. PERFORMANCE ASSESSMENT

To demonstrate the wide audience of devices enjoying compatibility with BitTransfer, the BitTransfer protocol has been implemented in real constrained wireless IoT devices, and its performance with a different setup of the parameters have been evaluated using a standardized protocol stack for IoT devices. In Section V-A we provide some preliminary implementation details, Section V-B provides some tests on the duration of BitTransfer with different configurations, while Section V-C includes our experimental energy consumption evaluation.

A. IMPLEMENTATION DETAILS

As the hardware board for our experimentation, we selected the OpenMote-b hardware platform, i.e., the state-of-the-art hardware board for real experimentation and rapid prototyping of IoT algorithms and solutions [41] [42]. The board features a 32 MHz CC2538 System on Chip (SoC), equipped with 512 kB of ROM, and 32 kB of RAM, and it supports the integration with a large number of sensors. As for the RF communication technology, the OpenMote-b leverages the CC2538 SoC to transmit layer-2 messages according to the IEEE 802.15.4-2015 standard, having a Maximum Transmission Unit (MTU) of 127B. To operate according to this standard, the OpenMote-b devices have been equipped with a 2.4GHz Rubber Duck STUB Antenna, with a RIGID 90 degrees RP-SMA Plug connector and a unity gain of 0 dBi.

The operating system selected to run on-board of the OpenMote-b has been the well-known OpenWSN protocol stack, already adopted by some contributions in the literature [29], [43], [44], since it integrates a slotted channel access mechanism and the widely accepted IEEE 802.15.4 standard operating in the Time Synchronized Channel Hopping (TSCH) mode [45].

We remark that, being equipped with a CC2538 SoC provided by Texas Instruments, the OpenMote-b allows to reduce the size of the preamble sequence, from the standardized length of 32 bits down to 8 bits, i.e., the hexadecimal sequence $0x00$. Thus, such a configuration of the preamble has been achieved to provide enhanced robustness to reactive jamming.

We recall that a generic device running the OpenWSN protocol stack executes Radio Frequency operations according to the IEEE 802.15.4 schedule. The schedule defines a *slotframe* as a pre-defined number of time-slots, repeating over time. Each time-slot in the schedule has a pre-defined time duration, i.e. 10 ms. In addition, each time-slot can be configured in several different modes:

- Transmission Slot (TX): used to transmit packets compliant to the IEEE 802.15.4 standard.
- Reception Slot (RX): used to receive packets compliant to the IEEE 802.15.4 standard.
- Transmission/Reception Slot (TX/RX): if the device has a packet to transmit, the slot is conceived as a transmission slot; otherwise, it is a reception slot.
- Serial Slot (SERIAL): used to transfer data to the serial port where the device is attached;
- Sleep Slot (OFF): no RF operation is performed.

By default, the OpenWSN protocol stack defines a slotframe of 11 slots, where one slot is a TX/RX slot, one is SERIAL, and the remaining nine (9) slots are OFF. However, this configuration can be changed to accommodate more RF operations in the time unit, and thus higher throughput.

Overall, the time required by BitTransfer to transfer a message of size M from a transmitter to potentially multiple receivers is strongly dependent on the configuration of the schedule. In fact, the higher the number of active slots (TX, RX, and TX/RX slots), the less the time required to transmit the message.

Figure 3 shows our live measurement setup. A total number of six (6) OpenMote-b devices were placed on a desk in an office scenario. One device has been elected as the root node of the IEEE 802.15.4 network, and it has been connected via USB to the host laptop, having gateway and network debugging functionalities. The other devices have been powered using two AA batteries.

Implementing a reactive jammer is a challenging task, and it is often conceived as a standalone research topic, as demonstrated by a few papers on the topic, including [46], [47], and [48], to name a few. Thus, we emulated the action of the reactive jammer in the network by modifying on purpose the bits within the IEEE 802.15.4 packets. Using such a strategy,

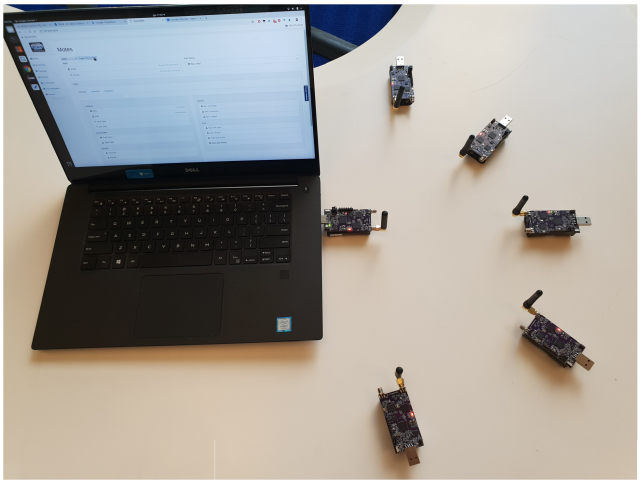

FIGURE 3. Photo of our testbed. A total number of six (6) OpenMote-b devices have been placed on a desk in an office scenario. One device has been selected as the root node of the IEEE 802.15.4 network, and it has been connected via USB to the host laptop, having gateway and network debugging functionalities. The other devices have been powered via 2 AA batteries.

given that the Frame Check Sequence (FCS) of the message does not match with the locally computed one, the packets are discarded as *corrupted* when received by any receiving device, thus emulating the effect of the reactive jamming in the network. We remark that this strategy is a common approach, in line with other scientific contributions working on anti-jamming solutions, such as [19]–[23], [26], [27], and [30].

B. TIME OVERHEAD

To provide a quantitative measure of the performance of the protocol, considering a couple of OpenMote-b devices communicating using the IEEE 802.15.4 technology at the PHY/MAC layer, we experimentally evaluated the time required by BitTransfer in the *Message Exchange Phase* to transfer a message of the maximum allowed size for the IEEE 802.15.4 technology, i.e., 127 B (1016 bits). Each test was executed by configuring a specific size of the slotframe, and also a specific number of active slots (TX/RX) within such a slotframe. Specifically, we configured deterministically the protocol, starting delivering messages always from the first slot in the slotframe (slot no. 0), and all the active slots were allocated after this one, from slot no. 0 to slot no. K-1, where K is the number of allocated active slots.

Fig. 4 reports our results, where the slotframe size has been set to the default value of the OpenWSN protocol stack, i.e. 11 slots. Fig. 5 instead, summarizes the results also for other values of the slotframe size.

Overall, the results indicate that increasing the number of active slots in the slotframe severely speeds up the execution of BitTransfer, with a trend that is decreasing almost exponentially as the percentage of active slots in the slotframe increases. Considering the default OpenWSN configuration (1 active slot over 11 in the slotframe), the time to transfer 1016 bits is the highest, i.e. 111.66 s. Much better

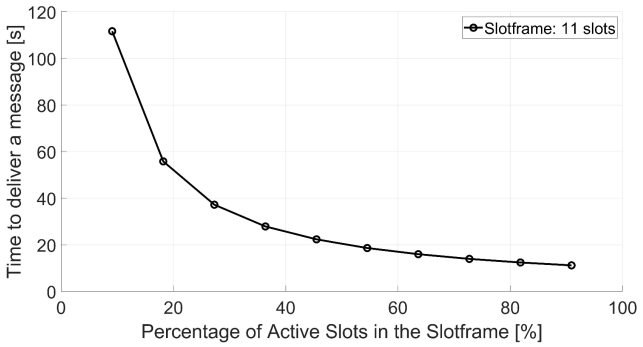

FIGURE 4. Time to transmit a message of size $M = 127$ B (1016 bits), while varying the percentage of active slots in a slot-frame with 11 slots.

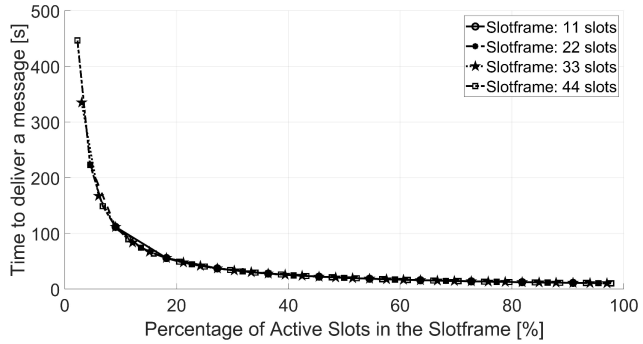

FIGURE 5. Time to transmit a message of size $M = 127$ B (1016 bits), while varying the percentage of active slots in the slot-frame for different slot frame size.

performance can be obtained increasing the number of active slots, up to 10 (1 slot in the slotframe should always be of the serial type, to allow communication with the gateway of the IoT network). In this case, only 11.17 seconds are necessary to transfer a message of the maximum size. Increasing the size of the slotframe has the potential to further decrease the time necessary to complete the protocol. As shown in Fig. 5, with a slotframe of 44 slots and 43 active slots, the time needed to transfer the full message is 10.39 s. Thus, when there is the requirement to minimize the time to transfer the message, increasing the slotframe size and the number of active slots is the solution to boost the throughput of BitTransfer.

Indeed, the size of the message to be transmitted also has an impact on the time required by BitTransfer in the *Message Exchange Phase* to complete. Thus, by assuming the same slotframe size of Fig. 4 (11 slots), we evaluated the time necessary to transfer a message of a specific size, by varying the number of active slots from 1 to 10.

In line with the results previously reported, increasing the number of active slots has a positive effect on the time required to complete the BitTransfer protocol. At the same time, as expected, the delivery of messages of a reduced size requires less time than delivering a message of a longer size. While these considerations can appear trivial, they have an impact on the overall end-to-end delay of a message, i.e., the time needed for the receiver to be in full possession of a message. In fact, embedding information in a longer packet can take much more time than transmitting such information

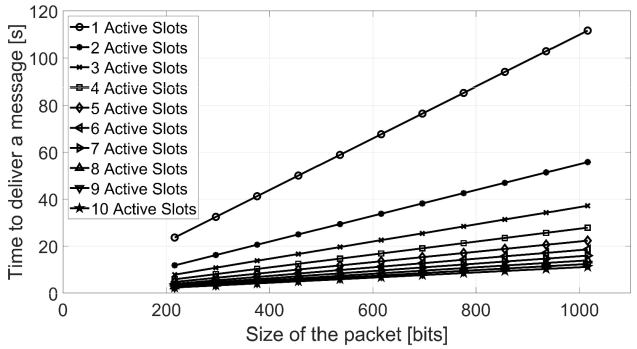

FIGURE 6. Time to transmit a message M of increasing bit size, increasing the number of active slots in a slotframe of 11 slots.

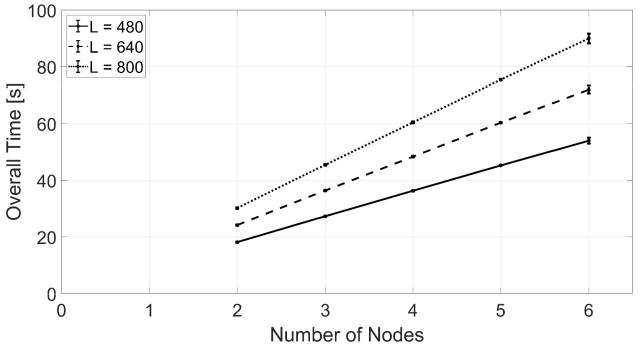

FIGURE 7. Tests with multiple nodes. We investigated the time required by all the nodes in the network to deliver their message, for a network with an increasing number of nodes, from 2 to 6, and a different size of the message to be delivered, i.e., 480, 640, and 800 bits, with a slotframe size of 11 slots and 6 active slots.

alone in a single instance of BitTransfer, thus shortening the overall end-to-end delay. Therefore, the information message to be encoded and transmitted by a node should be carefully selected to deliver the most information with the smallest number of bits (i.e., time-slots).

We also evaluated the time required by BitTransfer to let each node to deliver a message to all the other nodes in the network. Specifically, we considered the 6-nodes network shown in Fig. 3, and we set up the schedule with 6 active slots over a slotframe size of 11 slots. Then, we run consecutive runs of the BitTransfer protocol, where each node only transmits its message once, and we evaluated the time necessary for all the nodes to deliver their message, considering an increasing number of nodes, from 2 to 6, and an increasing size of the message to be delivered, i.e., 480, 640, and 800 bits. The synchronization protocol has been set up with a maximum backoff time value of $t_R = 20$. It is worth noting that, to reduce the time overhead due to the synchronization protocol, in each test we set up the value of the synchronization delay t_S to be the minimum possible, i.e., $t_S = N + 1$. Each configuration has been tested 10 times, and the corresponding results have been reported along with the 95% confidence interval, computed using the T-distribution formula. The results of our experiments are shown in Fig. 7.

We notice that the duration of the single run of the protocol is only slightly affected by the number of nodes in

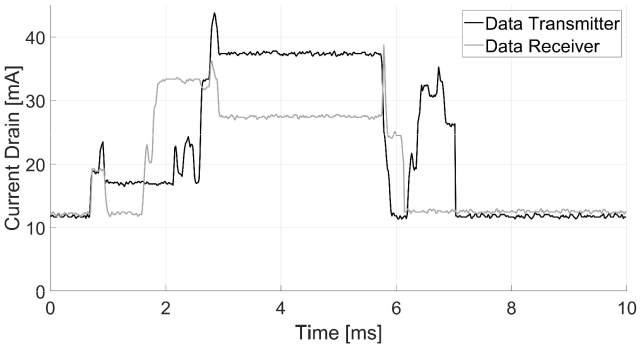

FIGURE 8. Current Consumption of simultaneous Data Transmission (black line) and Data Reception (grey line) slots during the execution of the BitTransfer scheme, over a slot duration of 10 ms.

the network. Indeed, being the synchronization protocol targeted to select a single transmitter, the duration of the single run of BitTransfer mainly depends on the size of the message to be delivered. When the nodes in the network increase, there is a linear increase in the synchronization delay, required to allow each device to deliver its *acknowledgement message* during the synchronization phase. However, being $t_S = N+1$, there is an increase of one slot for every new node joining the network. Overall, including the synchronization phase, and considering the present setup of the protocol, we notice that the single run of BitTransfer (i.e., the time required for a node to deliver its message) takes about 9.0495 s to deliver a message of 480 bits, 12.066 s for a message of 640 bits, and, finally, 15.0825 s for a message of 800 bits.

C. ENERGY CONSUMPTION

An important aspect to be considered when executing the BitTransfer protocol is energy consumption. In fact, BitTransfer requires the transmission of one packet for each bit to be delivered, thus requiring a significant energy consumption. To evaluate the energy expenditure of the BitTransfer protocol in the *Message Exchange Phase*, we experimentally measured the energy expenditure of the selected hardware board during a TX and a RX operation. Then, we obtained the overall energy consumption of BitTransfer on a TX and RX device by evaluating the number of TX and RX slots required to transmit a message of a specific bit-size M. To measure the energy consumption on the hardware board, we used an oscilloscope Keysight InfiniVision DSOX2012A, equipped with 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. The oscilloscope has been set with a vertical resolution of 8 bits, a vertical range of 50 mV/div, and a horizontal range of 1 ms/div. Figure 8 reports the measured current drain during a simultaneous TX and RX operation.

Focusing on the Data Transmitter (black line), we notice two current consumption spikes. The first one is located at an offset of about 3 ms from the start of the time-slot, it lasts for about 3 ms, and it has an average amplitude of 36.74 mA. This is the current consumption of the OpenMote-b board when the RF radio is on, in TX mode. The second spike, located

a few ms after the end of the first spike, lasting for about 1 ms, and having an average amplitude of 28.41 mA, indicates that the Data Transmitter turns on its RF radio again, in RX mode. This happens because, being the transmitted packet a unicast packet, according to the IEEE 802.15.4 standard, the transmitting device should expect an acknowledgment packet to be received at an offset of about 6.5 ms from the start of the slot. However, being the transmission disrupted by the reactive jammer, the integrity of the packet is not verified by the Data Receiver, and thus no ack is sent/received. It is worth noting that verifying the eventual reception of an acknowledgment packet in the BitTransfer protocol is neither casual nor useless. In fact, if the adversary is not able to jam every single transmission, or if the jamming activity is not present anymore on the channel, the protocol can be immediately aware of this change thanks to the successful exchange of the acknowledgment packet, and thus it can be immediately disabled, to restore the normal operation of the devices.

Focusing on the data receiver (grey line), we notice that it turns its radio on in RX mode at an offset of about 2 ms, with an average current consumption of 28.41 mA up to an offset of 6 ms from the start of the slot. Then, given that the integrity of the received packet is not verified (the CRC computed locally does not match the one in the packet), no acknowledgment is transmitted. We notice that the current consumption of both the devices when the radio is OFF and only the CPU is active is about 12.14 mJ. The overall energy consumption of BitTransfer on a TX and RX node, measured in mJ, can be obtained by integrating the instantaneous current drain i(t) over the time duration T of the slot, and multiplying it by 3.3V, i.e., the voltage of the OpenMote-b board, as indicated by the following Eq. 1.

$$E[mJ] = 3.3V \cdot \int_0^T i(t)dt. \quad (1)$$

It results that a TX slot consumes exactly 476.303 mJ, while a RX slot consumes 419.216 mJ.

We remark that all these operations, executed in a single time-slot of 10 ms, are connected with the transmission and reception of a single bit in the BitTransfer protocol. The overall energy consumption of BitTransfer in the *Message Exchange Phase* in the TX and RX mode has been computed by multiplying the above values for the number of time-slots and bits required to exchange a message. The results are reported in the following Fig. 9.

In line with our previous investigations, the node in TX mode consumes the most energy. Taking as a reference the transmission of a message having a size $M = 216$ bits (27 B), the node in TX mode consumes 102.88 J, while the node in RX mode consumes 90.55 J. These values increase when transmitting largest messages. The upper bounds for the transmission/reception of a message having size $M = 1016$ bits (i.e., 127 B, the MTU of IEEE 802.15.4) are 483.92 J for the node in TX mode and 425.92 J for the node in RX mode. We provide in Sec. VI a few details about the impact of this energy consumption on the battery lifetime.

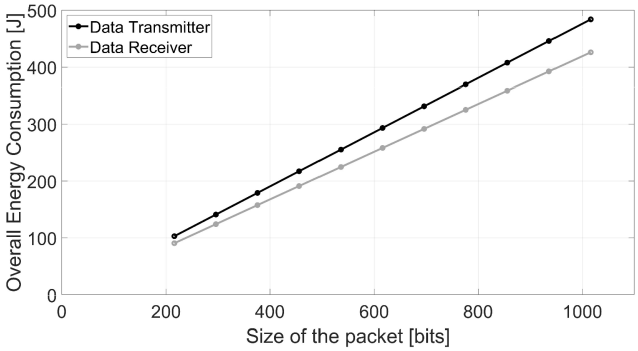

FIGURE 9. Overall Energy Consumption of BitTransfer, with different bit-string size *M*.

VI. COMPARISON, DISCUSSION, AND LIMITATIONS

In this section, we compare BitTransfer with the related work introduced in Sec. II, and we discuss its advantages and main limitations.

A. COMPARISON

Tab. 2 reports a qualitative comparison between the proposed BitTransfer protocol and the related work introduced in Sec. II, with reference only to the approaches robust to reactive jamming adversaries. [19]–[30]

We notice that some approaches assume simplified adversary models, where the jamming is limited to a given area of the network and it is not able to either listen to some frequencies, or to jam on the whole spectrum used by legitimate devices to communicate. Overall, a large part of the proposals introduced in the literature provide techniques to avoid reactive jamming that require the access to the raw signals (I/Q components) transmitted on the wireless channel. Thus, these solutions require either very specific hardware, such as SDR, or the modification of the hardware already deployed. Thus, they cannot be adopted when the devices in the Cyber Physical System have been already deployed or cannot be changed or accessed by the network administrator. Under these system requirements, the only approaches that can be adopted without hardware modifications are described by the authors in [22] and [23]. However, these approaches are successful only if the adversary is spatially limited, i.e., she can disrupt communications only in a part of the network, while another part can continue its operation. If the adversary is powerful and physically distributed, as in an EW scenario, also these approaches are not successful in maintaining wireless communication.

The proposed BitTransfer scheme, instead, can defeat even a reactive, spatially-unlimited, and frequency-unbounded adversary, without requiring modifications of the hardware already deployed, or to use specialized equipment. These features are particularly useful for any Cyber Physical System in a Electronic Warfare scenario. As summarized by Tab. 2, to the best of our knowledge, the combinations of these powerful features in a single protocol are still not available in the literature.

TABLE 2. Comparison of BitTransfer against competing solutions.

Contribution	Reactive Jamming Robustness	Spatially Unlimited Adversary	Global Eavesdropper Adversary	Frequency Unbounded Adversary	Noisy Channels Robustness	No need for Specialized Equipment	No Hardware Modifications
[19] [20]	✓	×	✓	✓	✓	×	×
[21]	✓	×	✓	✓	✓	×	×
[22] [23]	✓	×	×	✓	✓	✓	✓
[24]	✓	✓	✓	✓	✓	×	×
[25]	✓	✓	×	×	×	×	×
[26] [27]	✓	✓	✓	✓	✓	×	×
[28]	✓	✓	✓	✓	×	×	×
[29]	✓	✓	✓	×	✓	✓	✓
[30]	✓	✓	✓	✓	×	×	×
BitTransfer	✔	✔	✔	✔	✔	✔	✔

Despite the logic of the BitTransfer protocol could appear to be similar to the one proposed by the authors in [30], several differences and improvements can be found between the two contributions.

First of all, to identify transmission and silent periods, the authors in [30] used a mechanism based on the evaluation of the energy of the wireless channel. This strategy requires the communicating devices to access the physical characteristics of the wireless channel, and specifically the raw I/Q samples. This feature, however, is typically not available in regular Commercial-Off-The-Shelf (COTS) devices, where the micro-controller only provides access to the bits retrieved from an incoming packet. Thus, the deployment of the strategy proposed by the authors in [30] requires either the deployment of dedicated hardware, such as the Software Defined Radios (SDR), enabling the access to I/Q samples, or the replacement of regular COTS devices already deployed with enhanced hardware features, enabling the access to the I/Q samples. Therefore, this technique cannot be applied in Electronic Warfare scenarios, where the devices are already deployed and operational, and they cannot be replaced or physically accessed by the network administrator. The proposed BitTransfer protocol, instead, can detect transmit and silent periods thanks to the reception (or the absence) of a corrupted packet, by appropriately reducing the duration of the preamble to be less than the reaction time of the jammer. Thanks to this feature, BitTransfer can be implemented in COTS devices already deployed, thus becoming a valuable solution in Electronic Warfare scenarios.

In addition, the proposal by the authors in [30] only works between two neighboring nodes. When more than two nodes are involved and need to share information, such a scheme could not work effectively. The proposed BitTransfer protocol, instead, is equipped with a synchronization protocol robust to reactive jamming adversaries. Thus, it could work also in networks composed of more than two devices.

Finally, we highlight that the proposal by the authors in [30] is evaluated only via theoretical analysis and simulations. In this contribution, instead, we integrate the logic of the BitTransfer protocol in a real communication technology, and we implemented BitTransfer using real COTS devices, i.e., the OpenMote-b hardware platform. Furthermore, we also released the source code as open-source, further demonstrating that the logic proposed by BitTransfer is really suitable for integration as an anti-jamming scheme in regular COTS devices as a simple software update, without requiring any physical modification of the devices.

Being completely software-oriented, the proposed BitTransfer protocol can be installed on any device already deployed as a simple software update. This is possible since: (i) BitTransfer does not depend on the underlying physical layer communication technology; (ii) it does not rely on any specific physical-layer solution; and, (iii) it does not require any specific hardware capability. We remark that these are common requirements in any *Electronic Warfare* scenario, where Commercial-Off-The-Shelf (COTS) devices are used and are already deployed when attacks such as reactive jamming are initiated. We stress that, considering the adversary model detailed in Section III-A of our paper, the simultaneous fulfillment of all the requirements listed above is not possible with other protocols such as [30], given that one or more of the requirements detailed above cannot be achieved.

Finally, we recall that the source code of our proof-of-concept has been released as open-source [11]. This could allow practitioners, industries, and academia to verify our claims and to compare their own solutions with BitTransfer, eventually using our source code as a ready-to-use basis for their software development.

B. DISCUSSION AND LIMITATIONS

As any other protocol tailored for EW scenarios, BitTransfer has some limitations.

First, BitTransfer requires that the jamming equipment needs some time to start the jamming activity. Considering our reference implementation on the OpenMote-b hardware platform using the IEEE 802.15.4 technology, at least the first 8 bits of the preamble of the IEEE 802.15.4 packet should be received without any error by any receiver in the network. The IEEE 802.15.4 standard transmits messages with a physical data-rate of 250 kbps, meaning that 8 bits take about 32 μs to be delivered. As reported in [24], commercially available reactive jammers take about 1 ms to detect energy on the

channel and further 50 μs to switch from RX to TX. Thus, our solution is robust in the presence of such jammers.

Second, we highlight that BitTransfer could have issues when multiple networks featuring the same communication technology share the same spectrum over the same physical area. We recall that our reference scenario assumes that all the nodes are loosely time-synchronized, and that the wireless RF communications are scheduled on a time-slot basis, in line with modern CPS protocols such as IEEE 802.15.4 [31] and IEEE 802.11 [32]. These communication technologies define specific time windows, within the time slot, where RF messages can be transmitted. For instance, according to the deterministic Time Synchronized Channel Hopping (TSCH) mode of the IEEE 802.15.4 standard, the packets are transmitted exactly at a specific delay after the starting of the time-slot, and there is a small tolerance on the accuracy of this time, typically a few milliseconds. Thus, the Medium Access Control (MAC) layer protocol instructs the receiving devices to turn their radio on in RX mode only when a transmission is expected, while the radio is off when a transmission is not expected. In this way, RX devices can turn on their radio only when a packet is expected, reducing the overall energy consumption. For instance, within the OpenWSN protocol stack, at the MAC layer, over a time-slot duration of 10 milliseconds, the receiving device turns on its radio after 1.8 milliseconds (as shown in Figure 8), and it waits for a maximum time of 1.1 milliseconds before deciding that no packets have been transmitted.

Therefore, when adopted on top of these communication technologies, the BitTransfer protocol can be designed to be robust also when multiple concurrent independent networks are using the same communication technology over the same spectrum, at the same time. In fact, when the reactive jamming disrupts packets transmitted over a network different than the actual one, it is likely that the devices being part of the network where BitTransfer runs have their radio off. Indeed, we agree that there is a non-negligible probability that two (or more) neighboring networks define (partially) overlapping time windows used to transmit a packet. In this case, the performance of the BitTransfer protocol could be decreased.

On the one hand, we highlight that this is a networking issue, that affects the performance of the network also when there is no jamming and the BitTransfer protocol is not running (indeed, packets from different networks would overlap, decreasing the reliability of both the networks). Thus, in situations where multiple networks using the same communication technology coexist in the same area, the network administrator should be aware of the likelihood of collision events, and it should select the time windows in the MAC-layer protocol in order not to overlap each other. On the other hand, we remark that this is one of the reasons why the original message P to be transmitted over the wireless communication channel is encoded with a Forward Error Correction (FEC) function, such as a Repetition Code, resulting in the final bit-string L. Using these techniques, errors due to interferences from neighboring networks could be reduced, increasing the chances that BitTransfer succeeds in transferring the message.

In all the other situations where a packet could be transmitted anytime within the time-slot and two (or more) networks sharing the same communication technology insist on the same physical area, BitTransfer could not guarantee message delivery.

Lastly, we highlight that BitTransfer is an ideal solution when the communicating devices are mainly-supplied, and when energy availability is not an issue, such as with wireless routers, connected cars, autonomous vehicles, aircraft, and connected IoT home appliances. When the devices are battery-supplied, the network administrator should consider that the adoption of BitTransfer could lead to a significant amount of energy consumption, depending on the battery capacity on-board of the device. To provide a few reference values, we can consider the batteries powering the OpenMote-b (a low-end device), a Sky Viper Dash Nano Drone (a medium-end device), and a Samsung Galaxy S9 mobile phone (a high-end device). It is possible to compute the overall impact of the energy consumption on the lifetime of the device by using the values of the storage capacity and the voltage of the batteries.

Considering the OpenMote-b hardware platform, each board is powered by two Manganese/Alkaline AA cells, rated at about 2.4 ampere-hours, with 1.5 volts average. Overall, we have approximately 3.84 watt-hours, equivalent to 13, 824 Joules of storage capacity [49]. Thus, considering a message of 1016 bits, the node in TX mode uses about the 3.5% of the battery, while the node in RX mode uses about the 3.08%.

A Sky Viper Dash Nano Drone (see https://www.walmart.com/ip/Sky-Viper-Dash-Nano-Drone/177270524 for more details) is powered by a typical 1-cell Lithium Polymer battery; this battery drains 2.1 ampere-hours with a voltage of 2.7 volts, thus consuming roughly 7.77 watt-hours, equivalent to 27,720 Joules of storage capacity. Thus, considering a message of 1016 bits, BitTransfer consumes 1.74% of the battery in TX mode and 1.54% of the battery in RX mode.

Finally, a high-end device such as a Samsung Galaxy S9 mobile phone is powered by a battery draining 3.5 ampere-hours with a voltage of 4.4V, thus consuming roughly 15.4 watt-hours, equivalent to 554,400 Joules of storage capacity. Thus, considering a message of 1016 bits, BitTransfer requires 0.087% of the battery from the TX node and 0.077% of the battery from the RX devices.

Compared to the other solutions available in the literature, BitTransfer is characterized by significant energy consumption. For instance, in the recent contribution [29], considering the same hardware platform used for our paper, we experimentally measured overall energy consumption in the range $[0.12 - 0.16]$% of the battery. In our case, BitTransfer requires 1.74% of the available energy. At the same time, the comparison with other solutions can be established by looking at the required RF transmission and reception operations, that are the main source of energy consumption.

To provide a reference example, the authors in [25] reported a 200% message overhead, when compared to the delivering of a regular message, while our solution requires L messages to deliver one, where L is the number of bits equals to the ones the message M after the encoding with a generic error correction code. Thus, BitTransfer requires higher message overhead than competing solutions.

On the one hand, we highlight that the design of BitTransfer is a trade-off: when the devices cannot be accessed or changed, an increased amount of energy is required to deliver the message in a Electronic Warfare scenario. Thus, despite the energy consumption, BitTransfer is the only available option to allow the devices to communicate even in the presence of a powerful, location-unbounded, global eavesdropper, and frequency-unlimited adversary. On the other hand, wireless devices can be likely equipped with energy harvesting solutions, such as the ones based on renewable energy or RF power harvesting, to regain the energy drained during the execution of BitTransfer between consecutive instances of the protocol, thus increasing the battery lifetime [50].

Overall, despite the significant energy consumption, to the best of the authors' knowledge, BitTransfer is the first solution able to guarantee message delivery even under a powerful reactive jammer, being spatially-unlimited, and frequency-unbounded, without requiring modifications of the hardware already deployed, or to use specialized equipment.

VII. CONCLUSION

In this paper, we proposed BitTransfer, a protocol exploiting the presence (or absence) of wireless signals to enable communications between neighboring devices in Electronic Warfare scenarios. BitTransfer achieves its objective even in the presence of a powerful reactive jammer, disrupting any wireless communication independently from the used frequency, time, and physical location. In the worst case scenario, BitTransfer enables the transmission of a single bit per active time-slot, encoding the presence of the signal as a 1 and the absence of any radio activity as a 0. Due to its design simplicity and flexibility, BitTransfer can be easily implemented in any cyber-physical wireless device commercially available, without requiring any access or modification to the underlying hardware.

As a reference example, BitTransfer has been implemented on the OpenMote-b hardware platform, and it has been integrated within the widespread IEEE 802.15.4 communication technology, at the basis of the Zigbee 3.0 and Bluetooth technologies. For instance, using BitTransfer with the default IEEE 802.15.4 configuration available in the OpenWSN protocol stack, two neighboring devices can exchange a message of the maximum size of 127 B in 11.17 s, while competing approaches simply fail. Increasing the number of active slots per slotframe, a reduced end-to-end delay can be achieved. Finally, the source code of BitTransfer has been released as open-source, to enable industry, practitioners, and the scientific community to verify our claim, as well as to extend our protocol with further improvements.

Thanks to its flexibility and dependability even against powerful adversaries, BitTransfer emerges as a powerful anti-jamming solution, able to provide a communication channel (though degraded) even in challenging Electronic Warfare scenarios, where competing solutions do fail.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their comments and suggestions, that helped improving the quality of the manuscript.

The information and views set out in this publication are those of the authors and do not necessarily reflect the official opinion of the QNRF.

REFERENCES

[1] A. E. Spezio, "Electronic warfare systems," *IEEE Trans. Microw. Theory Techn.*, vol. 50, no. 3, pp. 633–644, Mar. 2002.

[2] R. Poisel, *Introduction to Communication Electronic Warfare Systems*. Norwood, MA, USA: Artech House, 2008.

[3] S. You, M. Diao, and L. Gao, "Deep reinforcement learning for target searching in cognitive electronic warfare," *IEEE Access*, vol. 7, pp. 37432–37447, 2019.

[4] M. Suchanski, P. Kaniewski, J. Romanik, E. Golan, and K. Zubel, "Electronic warfare systems supporting the database of the radio environment maps," *Proc. SPIE*, vol. 11055, Mar. 2019, Art. no. 110550M.

[5] K. Pelechrinis, M. Iliofotou, and S. Krishnamurthy, "Denial of service attacks in wireless networks: The case of jammers," *IEEE Commun. Surveys Tuts.*, vol. 13, no. 2, pp. 245–257, 2nd Quart., 2011.

[6] A. Mpitziopoulos, D. Gavalas, C. Konstantopoulos, and G. Pantziou, "A survey on jamming attacks and countermeasures in WSNs," *IEEE Commun. Surveys Tuts.*, vol. 11, no. 4, pp. 42–56, 4th Quart., 2009.

[7] G. Baldini, T. Sturman, A. R. Biswas, R. Leschhorn, G. Godor, and M. Street, "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, 2nd Quart., 2012.

[8] W. Xu, W. Trappe, Y. Zhang, and T. Wood, "The feasibility of launching and detecting jamming attacks in wireless networks," in *Proc. ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2005, pp. 46–57.

[9] M. Zhu, J. Mo, N. Xiong, and J. Wang, "Legitimate monitoring via cooperative relay and proactive jamming," *IEEE Access*, vol. 7, pp. 40133–40143, 2019.

[10] S. Vadlamani, B. Eksioglu, H. Medal, and A. Nandi, "Jamming attacks on wireless networks: A taxonomic survey," *Int. J. Prod. Econ.*, vol. 172, pp. 76–94, Feb. 2016.

[11] CRI-Lab. (2019). *Code of BitTransfer Protocol*. [Online]. Available: https://github.com/ssciancalepore/BitTransfer and https://cri-lab.net/bittransfer/

[12] A. Hamieh and J. Ben-Othman, "Detection of jamming attacks in wireless ad hoc networks using error distribution," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2009, pp. 1–6.

[13] R. Di Pietro and G. Oligeri, "Jamming mitigation in cognitive radio networks," *IEEE Netw.*, vol. 27, no. 3, pp. 10–15, May/Jun. 2013.

[14] R. Di Pietro and G. Oligeri, "GopJam: Key-less jamming mitigation via gossiping," *J. Netw. Comput. Appl.*, vol. 123, pp. 57–68, Dec. 2018.

[15] D. Ciuonzo, A. Aubry, and V. Carotenuto, "Rician MIMO channel- and jamming-aware decision fusion," *IEEE Trans. Signal Process.*, vol. 65, no. 15, pp. 3866–3880, Aug. 2017.

[16] M. Tiloca, D. De Guglielmo, G. Dini, G. Anastasi, and S. K. Das, "JAMMY: A distributed and dynamic solution to selective jamming attack in TDMA WSNs," *IEEE Trans. Depend. Sec. Comput.*, vol. 14, no. 4, pp. 392–405, Jul./Aug. 2017.

[17] A. Azim, S. Mahiba, T. A. K. Sabbir, and S. Ahmad, "Efficient jammed area mapping in wireless sensor networks," *IEEE Embedded Syst. Lett.*, vol. 6, no. 4, pp. 93–96, Dec. 2014.

[18] S. Khattab, D. Mosse, and R. Melhem, "Honeybees: Combining replication and evasion for mitigating base-station jamming in sensor networks," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, Apr. 2006, p. 8.

[19] X. Tang, P. Ren, Y. Wang, Q. Du, and L. Sun, "Securing wireless transmission against reactive jamming: A Stackelberg game framework," in *Proc. IEEE Global Commun. Conf.*, Dec. 2015, pp. 1–6.
[20] X. Tang, P. Ren, and Z. Han, "Jamming mitigation via hierarchical security game for IoT communications," *IEEE Access*, vol. 6, pp. 5766–5779, 2018.
[21] W. Xu, K. Ma, W. Trappe, and Y. Zhang, "Jamming sensor networks: Attack and defense strategies," *IEEE Netw.*, vol. 20, no. 3, pp. 41–47, May 2006.
[22] Y. Xuan, Y. Shen, N. Nguyen, and M. Thai, "A trigger identification service for defending reactive jammers in WSN," *IEEE Trans. Mobile Comput.*, vol. 11, no. 5, pp. 793–806, May 2012.
[23] Y. Xuan, Y. Shen, I. Shin, and M. T. Thai, "On trigger detection against reactive jamming attacks: A clique-independent set based approach," in *Proc. IEEE Int. Perform. Comput. Commun. Conf.*, Dec. 2009, pp. 223–230.
[24] S. Fang, Y. Liu, and P. Ning, "Wireless communications under broadband reactive jamming attacks," *IEEE Trans. Depend. Sec. Comput.*, vol. 13, no. 3, pp. 394–408, May/Jun. 2016.
[25] A. D. Wood, J. A. Stankovic, and G. Zhou, "DEEJAM: Defeating energy-efficient jamming in IEEE 802.15.4-based wireless networks," in *Proc. IEEE Conf. Sensor, Mesh Ad Hoc Commun. Netw.*, Jun. 2007, pp. 60–69.
[26] H. Rahbari and M. Krunz, "Friendly CryptoJam: A mechanism for securing physical-layer attributes," in *Proc. ACM Conf. Secur. Privacy Wireless Mobile Netw.*, 2014, pp. 129–140.
[27] H. Rahbari and M. Krunz, "Full frame encryption and modulation obfuscation using channel-independent preamble identifier," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 12, pp. 2732–2747, Dec. 2016.
[28] Y. Liu and P. Ning, "BitTrickle: Defending against broadband and high-power reactive jamming attacks," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 909–917.
[29] 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.
[30] R. Di Pietro and G. Oligeri, "Silence is golden: Exploiting jamming and radio silence to communicate," *ACM Trans. Inf. Syst. Secur.*, vol. 17, no. 3, Mar. 2015, Art. no. 9.
[31] *IEEE Standard for Low-Rate Wireless Personal Area Networks (WPANs*, IEEE Standard 802.15.4-2015, IEEE 802.15.4-2011, Apr. 2016, pp. 1–709.
[32] *Local and Metropolitan Area Networks–Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput*, IEEE Standard 802.11n-2009, IEEE Standard for Information Technology, Oct. 2009, pp. 1–565.
[33] M. R. Palattella, N. Accettura, L. A. Grieco, G. Boggia, M. Dohler, and T. Engel, "On optimal scheduling in duty-cycled industrial IoT applications using IEEE802.15.4e TSCH," *IEEE Sensors J.*, vol. 13, no. 10, pp. 3655–3666, Oct. 2013.
[34] N. Accettura, E. Vogli, M. R. Palattella, L. A. Grieco, G. Boggia, and M. Dohler, "Decentralized traffic aware scheduling in 6TiSCH networks: Design and experimental evaluation," *IEEE Internet Things J.*, vol. 2, no. 6, pp. 455–470, Dec. 2015.
[35] G. Micoli, P. Boccadoro, G. Valecce, A. Petitti, R. Colella, A. Milella, and L. A. Grieco, "ASAP: A decentralized slot reservation policy for dynamic 6TiSCH networks in industrial IoT," in *Proc. IEEE Convergent Internet Things (C-IoT)*, May 2019, pp. 1–6.
[36] M. Spuhler, D. Giustiniano, V. Lenders, M. Wilhelm, and J. B. Schmitt, "Detection of reactive jamming in DSSS-based wireless communications," *IEEE Trans. Wireless Commun.*, vol. 13, no. 3, pp. 1593–1603, Mar. 2014.
[37] O. Osanaiye, S. A. Alfa, and G. P. Hancke, "A statistical approach to detect jamming attacks in wireless sensor networks," *Sensors*, vol. 18, no. 6, p. 1691, May 2018.
[38] Q. Feng, H. Xu, Z. Wu, and W. Liu, "Deceptive jamming detection for SAR based on cross-track interferometry," *Sensors*, vol. 18, no. 7, pp. 2265, 2018.
[39] *CC2538 System-on-Chip Solution for 2.4-GHz IEEE 802.15.4 and ZigBee/ZigBee IP Applications*, Texas Instrum., Dallas, TX, USA, May 2013.
[40] R. Johannesson and K. S. Zigangirov, *Fundamentals of Convolutional Coding*, vol. 15. Hoboken, NJ, USA: Wiley, 2015.
[41] P. Tuset-Peiró, X. Vilajosana, and T. Watteyne, "OpenMote+: A range-agile multi-radio mote," in *Proc. Int. Conf. Embedded Wireless Syst. Netw. (EWSN)*, 2016, pp. 333–334.
[42] X. Vilajosana, P. Tuset, T. Watteyne, and K. Pister, "OpenMote: Open-source prototyping platform for the industrial IoT," in *Proc. Int. Conf. Ad Hoc Netw.* Cham, Switzerland: Springer, 2015, pp. 211–222.
[43] S. Sciancalepore, G. Piro, E. Vogli, G. Boggia, L. A. Grieco, and G. Cavone, "LICITUS: A lightweight and standard compatible framework for securing layer-2 communications in the IoT," *Comput. Netw.*, vol. 108, pp. 66–77, Oct. 2016.
[44] S. Sciancalepore, M. Vučinić, G. Piro, G. Boggia, and T. Watteyne, "Link-layer security in TSCH networks: Effect on slot duration," *Trans. Emerg. Telecommun. Technol.*, vol. 28, no. 1, p. e3089, 2017.
[45] T. Watteyne, X. Vilajosana, B. Kerkez, F. Chraim, K. Weekly, Q. Wang, S. Glaser, and K. Pister, "OpenWSN: A standards-based low-power wireless development environment," *Trans. Emerg. Telecommun. Technol.*, vol. 23, no. 5, pp. 480–493, 2012.
[46] M. Wilhelm, I. Martinovic, J. B. Schmitt, and V. Lenders, "Short paper: Reactive jamming in wireless networks: How realistic is the threat?" in *Proc. 4th ACM Conf. Wireless Netw. Secur.*, 2011, pp. 47–52.
[47] D. Nguyen, C. Sahin, B. Shishkin, N. Kandasamy, and K. R. Dandekar, "A real-time and protocol-aware reactive jamming framework built on software-defined radios," in *Proc. ACM Workshop Softw. Radio Implement. Forum (SRIF)*, 2014, pp. 15–22.
[48] D. S. Berger, F. Gringoli, N. Facchi, I. Martinovic, and J. Schmitt, "Gaining insight on friendly jamming in a real-world IEEE 802.11 network," in *Proc. ACM Conf. Secur. Privacy Wireless Mobile Netw. (WiSec)*, 2014, pp. 105–116.
[49] G. Pistoia, Ed., *Batteries for Portable Devices*. Amsterdam, The Netherlands: Elsevier, 2005, pp. 33–76.
[50] H. H. R. Sherazi, L. A. Grieco, and G. Boggia, "A comprehensive review on energy harvesting MAC protocols in WSNs: Challenges and tradeoffs," *Ad Hoc Netw.*, vol. 71, pp. 117–134, Mar. 2018.

SAVIO SCIANCALEPORE received the master's degree in telecommunications engineering and the Ph.D. degree in electric and information engineering from the Politecnico di Bari, Italy, in 2013 and 2017, respectively. He is currently a Postdoctoral Researcher with HBKU-CSE-ICT, Doha, Qatar. His major research interests include security issues in cyber-physical systems (CPS), including the Internet of Things (IoT), wireless sensor networks, avionic communication technologies, drones communications, and network security. He received the prestigious award from the ERCIM Security, Trust, and Management (STM) Working Group for the best Ph.D. Thesis in Information and Network Security, in 2018.

ROBERTO DI PIETRO was in the capacity of Global Head Security Research, Nokia Bell Labs, and an Associate Professor (with tenure) of computer science with the University of Padova, Italy. He has been working in the security field for more than 23 years, leading both technology-oriented and research-focused teams in the private sector, government, and academia (MoD, United Nations HQ, EUROJUST, IAEA, and WIPO). He is currently a Full Professor in cybersecurity with HBKU-CSE. He has been publishing more than 200 scientific articles over these topics, coauthored two books, edited one, and contributed to a few others. His main research interests include security and privacy for wired and wireless distributed systems (e.g. Blockchain technology, Cloud, the IoT, and OSNs), virtualization security, applied cryptography, computer forensics, and data science. He is serving as an AE for Elsevier ComCom and other International journals. From 2011 to 2012, he was awarded the Chair of Excellence from University Carlos III, Madrid. He has been receiving more than 7500 citations, with an H-index of 42, and an i-index of 115 according to Google Scholar.