DISH: DIstributed SHuffling against selective jamming attack in IEEE 802.15.4e TSCH networks

MARCO TILOCA, RISE SICS
DOMENICO DE GUGLIELMO, University of Pisa
GIANLUCA DINI*, University of Pisa
GIUSEPPE ANASTASI, University of Pisa
SAJAL K. DAS, Missouri University of Science and Technology

The MAC standard amendment IEEE 802.15.4e is designed to meet the requirements of industrial and critical applications. In particular, the Time Slotted Channel Hopping (TSCH) mode divides time into periodic, equally-sized, slotframes composed of transmission timeslots. Then, it combines time slotted access with multi-channel and channel hopping capabilities, providing large network capacity, high reliability and predictable latency, while ensuring energy efficiency. Since every network node considers the same timeslots at each sloframe and selects physical channels according to a periodic function, TSCH produces a steady channel utilization pattern. This can be exploited by a selective jammer to entirely thwart communications of a victim node, in a way that is stealthy, effective and extremely energy efficient. This paper shows how a selective jamming attack can be successfully performed even though TSCH uses the IEEE 802.15.4e security services. Furthermore, we propose DISH, a countermeasure which randomly permutes the timeslot and channel utilization patterns at every slotframe in a consistent and completely distributed way, without requiring any additional message exchange. We have implemented DISH for the Contiki OS and tested its effectiveness on TelosB sensor nodes. Quantitative analysis for different network configurations shows that DISH effectively contrasts selective jamming with negligible performance penalty.

CCS Concepts: • **Security and privacy** → **Network security**; • **Networks** → *Network protocols*;

Additional Key Words and Phrases: IEEE 802.15.4e, TSCH, Security, Selective Jamming, Denial of Service, Secure Schedule Permutation

*The corresponding author

Author's addresses: M. Tiloca is with the Security Lab of RISE SICS, Kista, Sweden (email: marco.tiloca@ri.se); D. De Guglielmo is with Mind srl, Modena, Italy (email: dome@mind.cc); G. Dini and G. Anastasi are with the Dept. of Information Engineering, University of Pisa, Pisa, Italy, (email: {gianluca.dini, giuseppe.anastasi}@unipi.it); S. K. Das is with the Dept. of Computer Science, Missouri University of Science and Technology, Rolla, MO, United States (e-mail: sdas@mst.edu).

This project has been funded by the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no. 607109. This work has also been supported by the NSF grants CNS-1545037, DGE-1433659, CNS-154050, and NeTS-1818942, the EIT Digital High Impact Initiative project ACTIVE, the PRIN project TENACE (n. 20103P34XC) funded by the Italian Ministry of Education, University and Research, and the University of Pisa (PRA 2015 program).

1550-4859/0/0-ART0 $15.00
https://doi.org/0000001.0000001

ACM Reference format:
Marco Tiloca, Domenico De Guglielmo, Gianluca Dini, Giuseppe Anastasi, and Sajal K. Das. 0. DISH: DIstributed SHuffling against selective jamming attack in IEEE 802.15.4e TSCH networks. *ACM Trans. Sensor Netw.* 0, 0, Article 0 (0), 30 pages.
https://doi.org/0000001.0000001

1 INTRODUCTION

The medium access control (MAC) standard amendment IEEE 802.15.4e [IEEE Computer Society 2012] extends the previous IEEE 802.15.4 standard [IEEE Computer Society 2011] for low-rate, low-power, and low-cost *Personal Area Networks* (PANs). The amendment has been designed to meet critical requirements of embedded and industrial applications, by reducing idle-listening and improving reliability in the presence of narrow-band interference and multi-path fading.

In particular, the *Time Slotted Channel Hopping* (TSCH) mode combines time slotted access with multi-channel and channel hopping capabilities, providing large network capacity, high reliability and predictable latency, while ensuring energy efficiency thanks to the time slotted access mode. The standard does not specify how the channel hopping sequence is set, and leaves the design of adaptive management algorithms to the user. TSCH can be used with any network topology, but it is particularly well-suited for multi-hop networks where multi-channel communication enables an efficient use of the available resources.

In this paper, we show that the advantages of TSCH can be severely impaired by a *selective jamming attack*, a specific kind of jamming attack which is particularly effective against time slotted wireless networks that retain the same communication pattern over time. In fact, TSCH divides the time into a sequence of periodic *slotframes*, each consisting of a fixed number of transmission *timeslots*. Timeslots are allocated to the nodes such that each node needs to be active only during its own timeslot(s), while it can *sleep* for the rest of the time. In a given timeslot, the node picks one of the available channels, according to a *channel-hopping function* which is periodic in the number of channels. However, periodicity over time and channels allows an adversary to monitor communications, and then quickly and easily determine timeslots and related channels that a victim node is going to use. Then, the adversary can selectively jam those channels in the related timeslots, thereby completely thwarting the victim's communications.

A selective jamming attack against TSCH turns out to be *energy efficient* and *hardly detectable* as it requires the adversary to be active only during (part of) the timeslots to jam. Furthermore, the attack is *efficient* since in order to determine the victim's communication pattern, an adversary has to monitor its communications for a fixed maximum number of slotframes, which is equal to the number of available channels (usually 16). Finally, the attack is also *effective* because the adversary is able to perform it despite the fact that IEEE 802.15.4e security services are in place.

The selective jamming attack is not new. We have shown that it may affect time division multiple access (TDMA) WSNs in general [Tiloca M., De Guglielmo D., Dini G., Anastasi G. and Das S. K. 2017] and the IEEE 802.15.4 guaranteed time slot (GTS) mechanism in particular [Daidone R., Dini G. and Tiloca M. 2013]. In this paper we show that even TSCH, a state-of-the-art standard for industrial systems, does not contain a more resistant approach to this kind of attack.

As a further contribution, we also propose DISH (DIstributed SHuffling), a preventive solution that counteracts selective jamming in TSCH and successfully mitigates its impact.

DISH is based on the basic technique of randomization. The timeslot and channel utilization patterns are permuted at every slotframe, so that the adversary is unable to determine the pattern of the victim node and predict its next timeslot-channel pair(s). It follows that the adversary is forced to jam timeslot-channel pairs at random, unless she is willing to increase the energy expenditure and network exposure. There are two salient aspects of randomization in DISH. First, it employs randomization in a *distributed* yet *consistent* way. This means that, starting from a collision-free timeslot-channel scheduling received at network joining, at every slotframe, every node computes the next timeslot-channel pair(s) autonomously, based only on local data, without exchanging any additional message and without causing collisions. The second salient aspect is that DISH integrates randomization in a way that does not break the current standard, as randomization acts on the timeslot-channel allocation strategy that IEEE 802.15.4e leaves to the upper layers.

We have implemented DISH for the Contiki OS [Dunkels A., Grönvall B. and Voigt T. 2004] and tested its effectiveness on TelosB sensor nodes [Moteiv Corporation 2006], in the presence of a real selective jammer implemented on the same platform. Experimental results and additional quantitative analysis confirm that DISH effectively contrasts selective jamming with negligible performance penalty. To the best of our knowledge, this paper is the first contribution that presents an effective selective jamming attack against IEEE 802.15.4e TSCH and proposes a MAC-level countermeasure to efficiently contrast it.

The rest of the paper is organized as follows. Section 2 surveys related work. Section 3 briefly discusses IEEE 802.15.4e and the TSCH mode, whereas Section 4 introduces the system model. Section 5 presents the adversary model and shows the selective jamming attack. Section 6 presents DISH, describes how nodes compute the next timeslot-channel utilization pattern in a distributed way, and analyses the DISH security. Section 7 evaluates the performance and effectiveness of DISH. Finally, Section 8 draws conclusive remarks.

2 RELATED WORK

Jammers may employ a wide range of strategies to disturb wireless communications [Mustafa H., Zhang X., Liu Z., Xu W. and Perrig A. 2012][Lazos L., Liu S. and Krunz M. 2009][Stojanovski S. and Kulakov A. 2015][Xu W., Ma K., Trappe W. and Zhang Y. 2006][Xu W., Wood T., Trappe W. and Zhang Y. 2004][Xu W., Trappe W., Zhang Y. and Wood T. 2005][Lu Z., Wang W. and Wang C. 2014]. Among them, *reactive* jammers become active upon detection of transmissions over the physical medium. Reactive jamming has been shown not only to be the hardest to detect, but also the most energy-efficient strategy, so making it a very severe threat in wireless networks [Spuhler M., Giustiniano D., Lenders V., Wilhelm M. and Schmitt J. B. 2014]. Reactive jamming can be implemented on inexpensive commercial off-the-shelf platforms and triggered selectively, for example on any field of a packet header, making it a realistic and actual threat for wireless communications [Proaño A. and Lazos L. 2010][Proaño A. and Lazos L. 2012][Wilhelm M., Martinovic I., Schmitt J. B. and Lenders V. 2011].

Recent works have shown that *selective* jamming is a specific kind of reactive jamming, which is particularly effective in time-slotted wireless networks, where jamming is selectively performed against specific communication slots. Specifically, in a time-slotted-based wireless network, a node gets assigned a number of timeslots, and typically retains them for a long time. Therefore, an adversary has an easy task in monitoring communication, detecting the slots assigned to the victim node to determine its communication pattern, and, finally, jamming those slots in order to completely thwart the node's communications. In addition to

being very effective, selective jamming displays both a very limited power expenditure and a very short network exposure, as the adversary has to periodically activate its transceiver only during the victim's slots. Moreover, the adversary's transmissions can be efficiently limited to a fraction of each targeted slots, while still effectively thwarting the victim's communications.

In general, th countermeasures against jamming can be adopted both at the physical and upper layers. Besides, the two kinds of countermeasures can of course complement each other. Relevant examples of strategies implemented at the physical layer include [Popper C., Strasser M. and Čapkun S. 2010][Chiang J. T. and Hu Y.-C. 2011][Jones K., Wadaa A., Olariu S., Wilson L. and Eltoweissy M. 2003][Spuhler M., Giustiniano D., Lenders V., Wilhelm M. and Schmitt J. B. 2014][Strasser M., Danev B. and Čapkun S. 2010][Mansour I., Chalhoub G. and Quilliot A. 2011][Xu W., Wood T., Trappe W. and Zhang Y. 2004][Xu W., Trappe W., Zhang Y. and Wood T. 2005]. However, the main drawback of physical countermeasures is that they focus on making jamming more complicated to carry out, rather than fundamentally preventing it. Thus they are not actually able to neutralize it. Also, they often result in additional overhead and worse performance, e.g., when network nodes have to reach a common understanding in frequencies/codes to enforce spread-spectrum techniques. In this paper, we mainly focus on countermeasures against reactive/selective jamming at the upper layers.

Richa et al. proposed ANTIJAM, a new MAC protocol which is robust to unintentional and malicious interference originated by a selective jammer that can determine whether the channel is currently idle or not [Richa A., Scheideler C., Schmid S. and Zhang J. 2013]. Wood *et al.* proposed DEEJAM, a new MAC protocol providing defence against jammers using IEEE 802.15.4-based hardware [Wood A. D., Stankovic J. A. and Zhou G. 2007]. DEEJAM relies on frequency hopping, redundant encoding and packet fragmentation to hide packets from a jammer, thus evading her search and limiting the impact of packets that are corrupted anyway. DEEJAM is compatible with existing nodes' hardware, but is specifically tailored to 802.15.4-based wireless sensor networks (WSNs) and introduces significant computational and energy costs in resource constrained sensor nodes.

Proaño *et al.* analysed a specific selective jamming attack, where the adversary thwarts the transmission of particularly important kinds of packets [Proaño A. and Lazos L. 2012]. They also proposed some methods, based on cryptographic primitives, to mitigate the attack effects. Encryption of transmitted packets is an effective solution against packet classification, but it requires that the entire packet, including the header, is encrypted (it is common practice to leave the header unencrypted, so that the receivers can early abort the reception of packets not destined to them). In their work, Proaño *et al.* considered a jammer that continuously senses and classifies packets to perform selective jamming based on their importance. Instead, this paper considers a different type of attack, where the adversary does *not* continuously monitor the channel to effectively perform selective jamming.

A completely different approach consists of reducing the predictability of transmissions in order to make selective jamming less efficient and convenient to carry out. Ashraf *et al.* proposed Jam-Buster, a low overhead framework against selective jamming [Ashraf F., Hu Y.-C. and Kravets R. H. 2012]. Jam-Buster relies on multi-block payloads, equally-sized packets, and randomisation of nodes' wake up time, in order to eliminate the differentiation of packet types and reduce predictability of transmission times. Hence, the adversary is forced to transmit more jamming signals, and thus spend more energy to be effective. Also, more jamming transmissions eventually result in a faster detection of the jamming source.

Jam-Buster does not try to outsmart the adversary through an actual anti-jamming solution, but focuses on making selective jamming less efficient and convenient to perform.

Sokullu *et al.* [Sokullu R., Korkmaz I. and Dagdeviren O. 2009] identified a selective jamming attack against IEEE 802.15.4 that exploits the Guaranteed Time Slot (GTS) mechanism. GTS is a form of time-slotted communication where up to seven reserved time slots in each superframe are allocated to the sensor nodes by a central Coordinator node [IEEE Computer Society 2011]. The authors illustrated two possible incarnations, namely the random attack and the intelligent attack. In the random attack, the adversary selects the slot to jam at random. In contrast, in the intelligent attack, the adversary exploits the knowledge of the allocation decision to select the longest slot. Sokullu *et al.* evaluated that an intelligent attacker can achieve a corruption strength of 50.48%, which means that only half of the available bandwidth is effectively used for communication during the content free period [Sokullu R., Dagdeviren O. and Korkmaz I. 2008]. Daidone *et al.* [Daidone R., Dini G. and Tiloca M. 2013] proposed a countermeasure against a GTS-based selective jamming where the Coordinator randomly generates a new slot allocation pattern at every superframe, and provides it to the GTS nodes. This countermeasure reduces the attack effectiveness down to at most $1/7$.

Tiloca *et al.* proposed JAMMY, a more general approach to counteract selective jamming [Tiloca M., De Guglielmo D., Dini G., Anastasi G. and Das S. K. 2017]. They considered a generic timeslotted, yet single channel, multi-hop wireless network where multiple nodes can join and leave dynamically, and proposed a distributed solution where each node autonomously computes the slots to use in the next superframe, without resorting to a centralised coordinator. JAMMY assures that the nodes' transmissions never collide and that the resulting slot allocation pattern always "appears" as random to a selective jammer. As it has been conceived for single channel networks, JAMMY cannot be used in TSCH.

A preliminary, non optimized, version of JAMMY, namely SAD-SJ, was presented in [Tiloca M., De Guglielmo D., Dini G. and Anastasi G. 2013]. SAD-SJ focuses on single-hop WSNs, where new sensor nodes are allowed to join the network only one at a time, and the ones already present in the network have to transmit additional information at every superframe.

A concise and recapitulatory view of the related works is reported in Table 1.

3 IEEE 802.15.4E

The IEEE 802.15.4e standard [IEEE Computer Society 2012] extends the previous IEEE 802.15.4 standard [IEEE Computer Society 2011] for low-rate, low-power and low-cost *Personal Area Networks* (PANs), to address industrial or embedded applications with critical requirements. To this end, it introduces two categories of enhancements, *MAC behaviors* and *general functional improvements*.

MAC behaviors are aimed to support specific applications, while general functional improvements are not tied to any specific application domain. Like in the original 802.15.4 standard, a PAN is formed by one PAN coordinator in charge of managing the whole network, and, optionally, one or more coordinators that are responsible for a subset of nodes in the network. Ordinary nodes must associate with a (PAN) coordinator in order to communicate using a specific MAC behavior mode. The IEEE 802.15.4e standard [IEEE Computer Society 2012] defines different MAC behavior modes (a detailed description is available in [De Guglielmo D., Brienza S. and Anastasi G. 2016a]). In this paper, we focus on the *Time Slotted Channel Hopping* (*TSCH*) mode, which is the most general and complex one.

Category	Papers	Pros and cons	Comparison with DISH
Physical layer	Popper *et al.*, 2010 Chiang *et al.*, 2011 Jones *et al.*, 2003 Spuhler *et al.*, 2014 Strasser *et al.*, 2010 Mansour *et al.*, 2011 Xu *et al.*, 2004 Xu *et al.*, 2005	They face selective jamming attack at the lowest level. They only mitigate the attack. They introduce overhead.	Not applicable, as enforced at a different layer than DISH.
Upper layers (New MAC)	ANTIJAM - Richa *et al.*, 2013 DEEJAM - Wood *et al.*, 2007	They are robust to jammers. They introduce relevant energy and computational overhead.	Not applicable, since DISH does not define a new MAC protocol.
Upper layers (Encrypted packet)	Proaño *et al.*, 2012	It prevents packet classification. It requires an encrypted header.	Proaño *et al.* consider continuous sensing and face jamming on selected packet types, using packet encryption. DISH considers limited sensing and jamming against target nodes, and does not require packet encryption.
Upper layers (Randomization)	Jam-Buster - Ashraf *et al.*, 2012	Low overhead framework. It only makes jamming less efficient and convenient.	It faces selective jamming by preventing packet differentiation and reducing predictability of transmission times. DISH does not alter packets and defeats selective jamming altogether.
	Daidone *et al.*, 2013	Efficient and effective. Only for the GTS mode of IEEE 802.15.4.	It considers a centralised solution to selective jamming against target nodes, for the GTS mode of IEEE 802.15.4. DISH addresses the same class of attack, in time-slotted and multi-channel IEEE 802.15.4e TSCH networks.
	JAMMY - Tiloca *et al.*, 2017	Multiple nodes can join at the same time. No communication overhead.	JAMMY is a decentralised solution to selective jamming against target nodes, in time-slotted, single channel networks. DISH addresses the same class of attack, in time-slotted and multi-channel IEEE 802.15.4e TSCH networks.
	SAD-SJ - Tiloca *et al.*, 2013	One node at the time can join. Small communication overhead.	SAD-SJ is a preliminary and non optimized version of JAMMY, where new nodes join the network once at the time, and the present ones regularly transmit additional information.

Table 1. Different approaches to selective jamming.

3.1 Time Slotted Channel Hopping

TSCH [IEEE Computer Society 2012] combines time slotted access with multi-channel and channel hopping capabilities. Hence, it provides large network capacity, high reliability and predictable latency, while ensuring energy efficiency, thanks to the time slotted access mode. TSCH can be used with any network topology (e.g. star, tree, partial/full mesh), and is particularly well-suited for multi-hop networks where multi-channel communication enables an efficient use of the available resources.

3.1.1 TSCH Access Mode. In TSCH, nodes synchronize on a periodic slotframe consisting of a number of timeslots. Figure 1 shows a slotframe composed of 4 timeslots. Each timeslot allows a node to send a maximum-size data frame and receive the related acknowledgment. If the acknowledgment is not received, the retransmission of the data frame will occur in the next time slot assigned to the same (sender-receiver) pair of nodes.

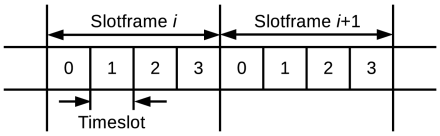


Fig. 1. Slotframe format.

TSCH relies on multi-channel communication and channel hopping. In principle, $N_C = 16$ different channels are organized as a *channel hopping sequence* and available for communication. The specific channel to consider is identified by means of a *channel offset*, i.e. an integer value in the range $[0, 15]$. In practice, the number of available channels N_C may be lower than 16, as some channels could be blacklisted due to low communication quality. In TSCH, a *link* is defined as the pairwise assignment of a direct communication between nodes in a given timeslot on a certain channel offset [IEEE Computer Society 2012]. Thus, a link can be represented as a pair $\{s, chOff\}$ where s specifies the timeslot s in the slotframe and $chOff$ the channel offset in that timeslot. Links are assigned to nodes for communication according to a *link scheduling algorithm*. The standard does not define any algorithm for link scheduling. Instead, it just defines some mechanisms to execute a link schedule provided by the upper layers (e.g., the application or network layer).

Let $\{s, chOff\}$ denote a link between two nodes. Then, the channel (or frequency) f to be used for communication in the timeslot s is derived as:

$$f = F[(ASN + chOff) \bmod N_C] \tag{1}$$

where mod indicates the modulo operation, while ASN is the *Absolute Slot Number*, i.e., the total number of timeslots elapsed since the start of the network (or an arbitrary start time determined by the PAN coordinator). Specifically, ASN is globally incremented in the network at every timeslot, and is thus used by nodes as a timeslot counter. Function F simply selects a channel from the channel hopping sequence according to the value of the index argument, and can generally be implemented as a lookup table. Thus, Equation 1 implements the channel hopping mechanism by returning a different channel for the same link at different slotframes. Due to multi-channel communication, many simultaneous transmissions can take place in the same timeslot, provided that they use different channel offsets. At the same time, this efficient link usage displays a number of properties that greatly simplify the performance of a selective jamming attack. We introduce such properties in Section 4, before we describe the selective jamming attack in Section 5.

Figure 2 shows a possible link schedule for periodic data collection in a simple network with a tree topology. In the considered example, the slotframe consists of 4 timeslots and there are only 5 channel offsets available. Thanks to the multi-channel communication, 8 transmissions are accommodated in a time interval corresponding to 4 timeslots. In the allocation shown in Figure 2, all links but one are *dedicated* links, i.e., allocated to a single node for transmission. TSCH also allows *shared* links, i.e., links intentionally allocated to more

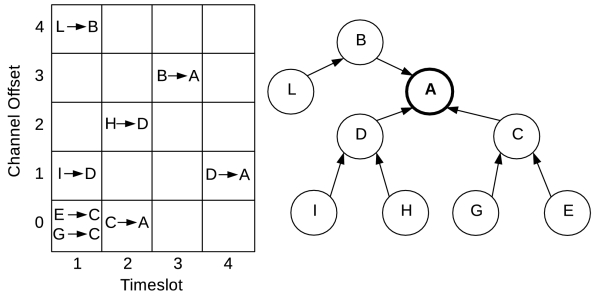


Fig. 2. Possible link schedule in a tree-topology network.

than one node for transmission. This is the case of the link $\{1, 0\}$ allocated for transmission to both nodes E and G.

3.1.2 TSCH Network Management. Network management relies on *Enhanced Beacons*, hereafter referred to as *Beacons* for brevity. They are special TSCH frames transmitted by network nodes at regular times, in order to disseminate control information (e.g. synchronization information, link schedule, etc.). Also, they allow new devices to dynamically join the network. Each Beacon includes the following information:

• *Synchronization information*: allows a new node to synchronize to the network, and includes the current ASN;

• *Channel hopping information*: allows to learn the channel hopping sequence used in the network;

• *Timeslot information*: allows to learn when to expect a frame transmission and when to send an acknowledgment;

• *Initial link and slotframe information*: allows to know: (i) when to listen for transmissions from the advertising node, and (ii) when to transmit to the advertising node.

A node that wishes to join the network starts scanning for possible Beacons on a given channel. Upon receiving a valid Beacon, the node initializes the slotframe and links, and starts operating in TSCH mode. Then, it typically allocates communication resources (i.e. links within the slotframe). The joining procedure may also include a security handshake to mutually authenticate the joining node, configure encryption keys, and configure routing information. However, the mechanism and rules for setting up communication resources and configure security and routing policies are not defined in the standard, as they are under the responsibility of the higher layers. Once connected and configured appropriately, a node can send Beacons on its turn. The Beacon advertising policy (i.e. slots and channel offset to be used by nodes for sending Beacons) is part of the link scheduling algorithm and, hence, it is under the responsibility of the higher layers.

3.1.3 Security in IEEE 802.15.4e and TSCH. IEEE 802.15.4e provides the same security services of the previous 802.15.4 standard [IEEE Computer Society 2011]. Specifically, it provides data confidentiality, data authenticity, and replay protection of MAC frames on a per-slotframe basis. If communications are secured, sender nodes build an *Auxiliary Security Header* (*ASH*), insert it next to the standard MAC header, and secure MAC frames before transmitting them. Then, based on the information carried in the ASH, recipient nodes correctly unsecure the received MAC frames.

The standard includes a security suite based on the *Advanced Encryption Standard* (*AES*) 128 bits symmetric-key cryptography [National Institute of Standards and Technology 2001]. Also, three different security modes are available, i.e. encryption only (*CTR*); authentication only (*CBC_MAC*); as well as both encryption and authentication (*CCM*). Both CBC_MAC and CCM modes rely on a *Message Integrity Code* (*MIC*), which can be 4, 8, or 16 bytes in size. Finally, IEEE 802.15.4e does not explicitly address the establishment of key material or device authentication, which are possibly entrusted to the higher layers. Therefore, both sender and recipient nodes must share common security settings and key material before they can start to securely communicate.

4 SYSTEM MODEL

This section describes the system model and introduces a number of properties of the link usage in TSCH.

Hereafter, we refer to an IEEE 802.15.4e network where nodes communicate according to the TSCH mode, with $N_C = 16$ available channels. This means that time is divided into periodic *slotframes* of equal duration, each one of which is in turn composed of N_S equally-sized *timeslots*, used by sensor nodes for transmitting/receiving data packets. Specifically, each sensor node remains active only during its own timeslot(s), while it turns off its radio interface and *sleeps* in the remaining time. We denote by s_i, $i = 1, \ldots, N_S$, the i-th timeslot in the slotframe.

In particular, we consider a multi-hop network represented by a communication graph $G = (U, L)$, where $U = \{u_1, \ldots, u_n\}$ is the set of nodes in the network and $L = \{l_1, \ldots, l_m\}$ is the set of directed edges $l = (u_i, u_j)$, representing a *link* between node u_i and u_j (hereafter, when there is no risk of ambiguity, we use "link" and "edge" interchangeably). Specifically, an edge $l = (u_i, u_j)$ exists iff node u_i transmits data to node u_j. Thanks to the presence of multiple channels, many links can be simultaneously active during the same timeslot, provided that they do not interfere with each other. In particular, at every link, no collisions have to occur, during both the data packet and ACK transmission (*Collision-Free Property*). In other words, for every link $l = (u_i, u_j) \in L$, it must be guaranteed that, when link (u_i, u_j) with channel offset c is active: i) no other node within the interference range of u_j transmits data with the same channel offset c; and ii) no other node within the interference range of u_i receives data (and, hence, sends ACKs) with the same channel offset c.

More formally, for each link $l \in L$, we define the set of interfering links $\mathcal{I}(l)$ which includes all the links belonging to L that interfere with l (note that $\mathcal{I}(l)$ contains l itself). Furthermore, we introduce a binary variable $x_l(s, c)$ such that $x_l(s, c) = 1$ if link $l \in L$ is active during timeslot s with channel offset c, and 0 otherwise. This means that, if link l is active during timeslot s with channel offset c, the associated interfering set $\mathcal{I}(l)$ contains one active link only, i.e. l itself. The Collision-Free Property can be now defined as follows:

Definition 4.1 (Collision-Free Property).

$$\forall\, l \in L, \forall\, s, \forall\, c, \sum_{i \in \mathcal{I}(l)} x_i(s, c) = 1 \text{ if } l \text{ is active for } \{s, c\}$$

We assume that any active node has succesfully joined the IEEE 802.15.4e network and received a collision-free link schedule. In particular, hereafter we assume that a node V has been granted with $0 < N_V \leq N_S$ timeslots per slotframe to communicate with other nodes. Then, the node stores and refers to N_V pairs $\{s, c\}$, where $0 \leq s < N_S$ indicates the timeslot

to access, and $0 \leq c < N_C$ indicates the channel offset to consider for that timeslot. For each link, the node maintains also the associated paired node, as well as the communication direction considered during the timeslot, i.e. either transmission or reception.

Without loss of generality, we also make the following assumptions, adopted in typical network settings to assure the full usage of all the available channels [De Guglielmo D., Brienza S. and Anastasi G. 2016b]:

(1) N_S and N_C are *coprime*;
(2) Function F in Equation 1 is *bijective.*

Under Assumptions (1)–(2), a number of properties hold. For simplicity and without loosing in generality, in the following we refer to the channel hopping sequence $\{0, 1, \ldots, N_C - 1\}$ and consider the *identity function* $F(x) = x$. Hence,

$$f = (ASN + chOff) \bmod N_C \tag{2}$$

Notice that this choice is made in the Contiki implementation of TSCH, for example [Contiki 2016].

Ch Off	ASN																
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0			f = 2			f = 1			f = 0			f = 3			f = 2		
1		f = 2			f = 1			f = 0			f = 3			f = 2			f = 1
2																	
3	f = 3			f = 2			f = 1			f = 0			f = 3			f = 2	
	T = 0			T = 1			T = 2			T = 3			T = 4			T = 5	

Fig. 3. Example of channel sequences ($N_S = 3$, $N_C = 4$).

Furthermore, in order to illustrate the properties, we refer to the example in Figure 3 which depicts the pattern of allocation of channels to links $l_1 = \{0, 3\}, l_2 = \{1, 1\}$, and, $l_3 = \{2, 0\}$, when $N_C = 4$ and $N_S = 3$.

Property 1 (Periodicity Property). *The sequence of channels used for communication by a certain link repeats with period $N_C \cdot N_S$ timeslots.*

With reference to Figure 3, the sequence of channels allocated to link l_1, for example, is periodic and the period is 12 timeslots.

Property 2 (Usage Property). *Within a period, every link uses all the available channels, each of which only once.*

With reference to Figure 3, link l_1, for example, uses all channels within a period. The same applies to all the other links as well.

Property 3 (Offset Property). *All links follow the same sequence of channels with a certain offset.*

With reference to Figure 3, let us consider links l_1 and l_2. The former uses channels $\{3, 2, 1, 0\}$ whereas the latter uses channels $\{2, 1, 0, 3\}$. Actually, they follow the same sequence with different starting points. The same remark applies to link l_3 as well.

These properties were originally presented in [De Guglielmo D., Brienza S. and Anastasi G. 2016b], which we refer to for the related proofs.

Definition 4.2 (Predictability). We say that the sequence of channels used by a given link is *predictable* if the knowledge of the channel that the link uses in a given timeslot allows us to compute the remaining channel hopping sub-sequence.

Property 4 (Predictability Property). *For each link, the sequence of channels is predictable.*

Proof. Let $l = \{s, c\}$ be an allocated link and f the channel used by the link in timeslot s of slotframe T. As the ASN of timeslot s is $ASN = s + T \cdot N_S$ then

$$f = (s + T \cdot N_S + c) \bmod N_C. \tag{3}$$

By solving Equation 3 in c, one becomes able to predict the channels used by the link in the next slotframes. □

With reference to Figure 3, if one knows that link l_1 uses channel 1 in timeslot $s = 0$ of slotframe $T = 2$, then it is possible to compute c from equation $1 = (6 + c) \bmod 4$, that gives $c = 3$. It follows that Equation 3 becomes $f = (3T + 3) \bmod N_C$, which allows us to predict frequencies used by link l_1 in every slotframe $T > 2$.

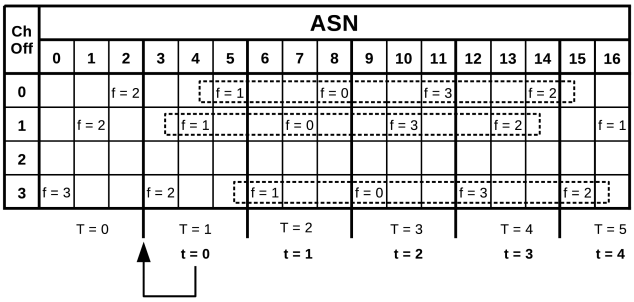


Fig. 4. Example of slotframe re-numbering.

It is important to notice that due to the Offset Property, one does not really need to know the absolute number T of the slotframe (and thus the ASN) in which the timeslot s uses a certain channel f. Actually, one may number slotframes starting from any slotframe arbitrarily assumed as "slotframe-zero". For instance, with reference to Figure 4, one may assume that slotframe $T = 1$ is the slotframe-zero $t = 0$. This means that now link l_1 uses channel $f = 1$ in timeslot $s = 0$ of slotframe $t = 1$. Considering these values in Equation 3, one gets to solve $1 = (3 + c) \bmod 4$ in c, which gives $c = 2$. Now, Equation $f = (3t + 2) \bmod 4$ makes it possible to predict channels allocated to link l_1 according to the relative slotframe numbering.

5 SELECTIVE JAMMING ATTACK IN TSCH

With reference to the system model defined in Section 4, we consider an *external* adversary whose objective is to disrupt all transmissions from one specific victim node, by performing a *selective jamming* attack, i.e. by maliciously transmitting during the victim's transmission timeslots. Since we consider an ideal communication channel, corrupted frames are due only to the selective jamming attack. Also, we assume that the adversary does not compromise any network node, either physically or logically, but she is able to listen to and jam any communication within the IEEE 802.15.4e network. In addition, while performing the attack,

the adversary is willing to be as much invisible as possible, in order to limit the likelihood of being detected, and to save as much energy as possible.

In order to perform such an attack, the adversary must be aware of the full communication pattern of her victim node V, i.e. of all the pairs $\{s, c\}$ that node V uses to communicate. Section 5.1 shows in detail that the adversary can always easily and quickly derive the communication pattern of her victim node, even if security services are adopted. To this end, the adversary has simply to monitor the communications of node V for a given number of slotframes, until the communication pattern is fully disclosed. Properties 1–4 introduced in Section 4 greatly simplify this task.

The specific approach that the adversary adopts to physically identify and track the communications of node V is not important here. For instance, she may exploit some prior knowledge such as the victim's identifier, its position in the network, or the type of traffic it produces. Furthermore, we reasonably assume that the adversary can derive system parameters as N_C, N_S, and the duration of single timeslots by means of simple traffic inspection techniques, in order to correctly synchronize with the IEEE 802.15.4e network.

Once the communication pattern has been fully derived, the adversary systematically jams all the timeslots assigned to the victim node as follows. First, the adversary stays quiet until one of the victim's timeslots s. Then, she considers the associated channel offset value c, and determines the correct channel f according to Equation 1. Finally, she starts transmitting a radio signal on channel f, as soon as it senses the activity from the victim node during timeslot s. This behavior features a form of reactive jamming which is harder to detect than traditional wide-band jamming [Xu W., Ma K., Trappe W. and Zhang Y. 2006]. Since the victim node refers to the same communication pattern in all slotframes, the attack is 100% *effective* even if channel hopping is performed on a slotframe basis. Besides, the attack is also *energy-efficient*, as the adversary has to activate her transceiver and jam victim's communications only during N_V timeslots per slotframe, on one (different) radio channel each, while she can turn off her radio during all other timeslots. Note that adversary's transmissions can be efficiently limited to a fraction of each jammed timeslot, while still effectively thwarting victim's communication. Finally, the attack becomes *hardly detectable*, as it exposes the adversary for a very limited amount of time, i.e. N_V timeslots per slotframe.

Throughout the paper, we refer to *jammed area* as the portion of the network within which no frames can be correctly received during the attack performance.

5.1 On determining the victim's communication pattern

In this section, we show that an adversary can determine the communication pattern of the victim node and jam all its communications even in case IEEE 802.15.4e security services are adopted. That is, we assume that MAC frames are authenticated and their payload is encrypted. This implies that the adversary is not able to retrieve the ASN and the channel hopping sequence from any transmitted Enhanced Beacon. However, the adversary may exploit Properties 1–4 to determine the victim's communication pattern, as follows.

As a first step, the adversary starts the attack at the beginning of a slotframe that she considers the "slotframe-zero", and to which she assigns slotframe number $t = 0$. Actually, the adversary cannot know the absolute slotframe number T of slotframe-zero. In order to know it, she would need to eavesdrop the ASN value from the Enhanced Beacon which, by assumption, is encrypted. However, by virtue of the Predictability and Offset Properties, she does not need this information. In practice, the adversary can "rename" slotframes starting from the slotframe-zero.

Then, as a second step, the adversary picks a channel at random, say $\hat{f}$, and monitors it for N_C consecutive slotframes to determine the timeslots in which the victim node communicates on that channel. By virtue of the Usage Property, the number of such timeslots is equal to the number of links assigned to the victim node.

As a third and final step, the adversary determines the channels on which the victim node is going to transmit in the next slotframes. Let us assume that the victim node uses a link $\hat{l}$, and thus communicates on channel $\hat{f}$ in timeslot $\hat{s}$ of slotframe $\hat{t}, 0 \leq \hat{t} < N_C$. Then, according to the Predictability Property, she can compute $\hat{c}$ and thus can predict the channels used by link $\hat{l}$ in the next slotframes.

In order to fix ideas, let us consider the example in Figure 4, and assume that links l_1, l_2 and l_3 have been allocated to the victim node. Let us assume that the adversary starts the attack at absolute slotframe $T = 1$, which thus becomes the slotframe-zero, namely $t = 0$ (step 1).

Furthermore, let us assume that the adversary decides to monitor channel $\hat{f} = 1$. By monitoring N_C slotframes, she determines that the victim node transmits on that channel in timeslots $s = 1$ and $s = 2$ of slotframe $t = 0$ and in timeslot $s = 0$ of slotframe $t = 1$. Thus, the adversary deduces that the victim node is using three different links l_1, l_2 and l_3 in timeslots 0, 1 and 2, respectively (step 2).

Also, by instantiating Equation 3 for link l_1, timeslot $s = 0$ and slotframe $t = 1$, she gets $1 = (3 + c_1) \mod 4$, that has solution for $c_1 = 2$. Thus, the function that predicts the channel to be used by link l_1 in a slotframe $t, t \geq 1$, is $f_1 = (2 + 3t) \mod 4$, which produces the correct periodic sequence of channels $\{1, 0, 3, 2\}$. Besides, by instantiating Equation 3 for link l_2, timeslot $s = 1$ and slotframe $t = 0$, the adversary obtains that the function f_2 that predicts the channel to be used by link l_2 in a slotframe $t, t \geq 0$, is $f_2 = (1 + 3t) \mod 4$ which produces the correct periodic sequence $\{1, 0, 3, 2\}$. Finally, by instantiating Equation 3 for link l_3, timeslot $s = 2$ and slotframe $t = 0$, the adversary obtains that the function f_3 that computes the channel to be used by link l_3 in a slotframe $t, t \geq 0$, is $f_3 = (5 + 3t) = (1 + 3t) \mod 4$, which produces the correct periodic sequence of channels $\{1, 0, 3, 2\}$.

5.2 Performance analysis of the attack

Discovering all the victim's transmission links takes exactly N_C slotframes, and requires the adversary to listen to $N_C \cdot N_S$ timeslots. Thus, the energy spent to find the complete victim's communication pattern is equal to $E = N_C \cdot N_S \cdot P_{RX} \cdot D_s$, where P_{RX} is the radio power consumption in receive mode, and D_s is the duration of a single timeslot. If we consider $N_C = 16$ channels, a slotframe composed of $N_S = 101$ timeslots of $D_s = 15$ ms each and a power consumption $P_{RX} = 35.46$ mW [Texas Instruments 2012], then the discovering phase lasts $D = N_S \cdot N_C \cdot D_s = 24.24$ s. If the adversary listens to each timeslot entirely, this phase results in an energy consumption $E = D \cdot P_{RX} = 859.55$ mJ.

As we have discussed above, typical network settings assure the full usage of all the available channels by considering coprime values of N_C and N_S. In the following, we give intuitions of how the attack can be performed when N_C and N_S are non coprime values. That is, the adversary can perform the same procedure described above for multiple *rounds*, by monitoring one of the available channels at each round. Then, determining the complete victim's communication pattern requires $r = N_C/D_r$ rounds, where $D_r \leq N_C$ is the duration of each round in slotframes and depends on the specific pair (N_C, N_S).

6 THE DISH ALGORITHM

The selective jamming attack considered in this paper is based on the following simple observation. In a conventional IEEE 802.15.4e network, every node gets assigned a communication pattern upon joining the network, and then uses that very same pattern for many consecutive slotframes, typically until it leaves the network. Thus, a way to counteract a selective jammer consists in changing the communication pattern of all nodes at every slotframe in an *unpredictable* way. The challenge in achieving this goal is to seamlessly preserve collision-free communications, while at the same time limiting the impact on performance and ideally avoiding the exchange of additional messages among network nodes.

Then, if nodes' communication pattern becomes unpredictable at every slotframe, the adversary is not able anymore to predict the links used by the victim node in the upcoming slotframes, even by observing its past network activities. Furthermore, if the adversary wants to remain power-efficient and hardly detectable, the only viable strategy consists in randomly select the link to jam. However, this greatly reduces the attack effectiveness. To fix ideas, let us consider a victim node that uses a single link. In this case, the effectiveness becomes $1/(N_S \cdot N_C)$, where N_S is the slotframe size in timeslots and N_C is the number of available channels. In Section 7, we discuss the effectiveness of this attack in the most general case where $N_V \geq 1$ links are allocated to the victim node.

In order to achieve this goal and thus counteract the attack, we compute the next communication pattern as a *random permutation* of the current one, at every slotframe. However, it is not sufficient that the new communication pattern is *unpredictable*. In fact, we also require that all nodes compute the next communication pattern *autonomously*, i.e. considering only locally available information without exchanging additional messages. Also, the new communication pattern must be *consistent*, i.e. all nodes must autonomously compute the *same* permutation. This is necessary in order to guarantee that the resulting link schedule is always collision-free.

To fulfil these requirements, we assume that each node executes a *random link permutation algorithm*. That is, at every slotframe, each node randomly permutes the current communication pattern and produces the next one. In particular, each node separately and independently permutes the *timeslot utilization pattern* and the *channel-offset utilization pattern*. Typically, a random permutation algorithm relies on a random number generator. Since collisions must be prevented, all nodes must compute the same permutation and, thus, have to produce the *same* sequence of random numbers. Hence, all nodes must use *pseudo-random number generators* which must be maintained in the same internal state over time. This also implies that, when a new node joins the network, its generators must be initialized into the same internal state as the ones of the nodes already in the network. Besides, in order to fulfill the unpredictability requirement, the sequence of psedo-random numbers must also be unpredictable, and thus the pseudo-random number generators must be secure [Menezes A. J., van Oorschot P. C. and Vanstone S. A. 2001].

In the next sections, we present DISH, our countermeasure against selective jamming. Specifically, in Section 6.1 we present the secure pseudo-random number generator (SPRNG) used in DISH, while in Section 6.2 we introduce the random link permutation algorithm. In Section 6.3, we discuss how nodes can join and leave the network at any time without jeopardizing other nodes' communications or the countermeasure against selective jamming. Finally, in Section 6.4 we analyze the impact of a selective jamming attack when the proposed countermeasure is in place.

We would like to point out that an adversary is of course still able to completely jam the network by performing a constant and/or wide-band jamming, i.e. by interfering with all the timeslots and/or physical channels. Alternatively, she can continuously monitor the network to detect the new timeslots used by her victim node, and then selectively jam them. However, by doing so she would severily compromise the attack's hard-detectability and power efficiency. In particular, a wide-band jamming would make the adversary considerably easier to be detected [Xu W., Ma K., Trappe W. and Zhang Y. 2006]. Moreover, wide-band jamming and continuous monitoring would increase the adversary's power consumption, thus making the attack much less convenient from the energy point of view.

6.1 A Secure Pseudo-Random Number Generator

In order to implement a Secure Pseudo-Random Number Generator (SPRNG) that is also affordable for resource-constrained nodes, we have considered a block cipher in the counter mode [Menezes A. J., van Oorschot P. C. and Vanstone S. A. 2001] (see Algorithm 1). Let $E(x, y)$ denote a cipher which encrypts a plaintext y by means of a key x. First, we provide the generator with an encryption key K, and initialize a counter z to a random seed z_0. Then, we apply the cipher to the sequence of values $z, (z + 1), (z + 2), \ldots$, so producing the output random sequence $E(K, z), E(K, z + 1), E(K, z + 2), \ldots$. Hereafter, we call counter z the *internal state* of the generator, and K the *permutation key*. We also assume that K is kept secret and that its length discourages an exhaustive key search.

ALGORITHM 1: Secure Pseudo-Random Number Generator.

```
1 unsigned random(unsigned K, unsigned z) {
2     unsigned val = E(K, z);
3     return val;
4 }
```

This is a common method to build a SPRNG out of a cipher [Menezes A. J., van Oorschot P. C. and Vanstone S. A. 2001][Paar C. and Pelzl J. 2010]. The crucial design requirement is that the cipher must be secure. Here we refer to the AES cipher, which has the following two advantages. The first one is security: there is currently no known analytical attack against AES with a complexity less than a brute-force attack [Paar C. and Pelzl J. 2010]. The other advantage is that AES is affordable on resource constrained nodes. Besides, commercially-available node platforms such as Tmote Sky provide AES-128 encryption in hardware, with negligible overhead in terms of delay, storage, and energy consumption [Daidone R., Dini G. and Tiloca M. 2011].

6.2 The secure link permutation algorithm

In this section, we describe the Secure Link Permutation (SLP) algorithm used by DISH to protect communications against the considered selective jamming attack. For simplicity of presentation, we consider the Fisher-Yates algorithm [Knuth D. E. 1998], also known as the Knuth shuffle algorithm, which runs in $O(n)$ time. This requires each node to store two vectors of N_S elements, to separately permute the *timeslot utilization pattern* and the *channel offset utilization pattern*. To better support resource constrained platforms, we have done an implementation that requires to store two vectors of N_V elements, so considerably limiting the memory occupancy on nodes. In principle, each node maintains only the information

about the timeslots and channel offsets that it actually considers for its own communication links. We report this implementation in Appendix A.

Hereafter, we assume that each node maintains two separate *permutation vectors* composed of N_S unsigned elements, which represent the node's view of the current communication pattern. We denote by X_V^s the permutation vector of node V where $X_V^s[i]$ refers to the i-th timeslot in the slotframe. Also, we denote by X_V^c the permutation vector of node V where $X_V^c[i]$ refers to the channel offset value considered in the i-th timeslot in the slotframe. Each node maintains its own permutation vectors as follows.

If node V does not use timeslot s_i, then $X_V^s[i] = 0$ and $X_V^c[i] = N_C$. If node V uses timeslot s_i and channel offset $0 \leq c < N_C$ to transmit data, then $X_V^s[i] = 1$ and $X_V^c[i] = c$. Finally, if node V uses timeslot s_i and channel offset $0 \leq c < N_C$ to receive data from an associated transmitter, then $X_V^s[i] = 2$ and $X_V^c[i] = c$.

We recall that, upon joining the network, each node receives a link schedule such that the overall communication pattern is collision-free. That is, at every slotframe, all links in the network active in a given timeslot s_i do not interfere with each other. More formally, let us consider four nodes a, b, c and d, as well as the system model introduced in Section 4. Then, for any pair of links $l_1 = (a, b)$ and $l_2 = (c, d)$ active during timeslot s_i, i.e., $X_A^s[i] = X_C^s[i] = 1$ and $X_B^s[i] = X_D^s[i] = 2$, we have $l_1 \notin \mathcal{I}(l_2)$ and $l_2 \notin \mathcal{I}(l_1)$, that is l_1 and l_2 do not interfere with each other. Note that element i can be 0 in every permutation vector X^s iff timeslot s_i is not associated to any node. Similarly, note that element i can be N_C in every permutation vector X^c iff channel offset c is not associated to any node for any timeslot.

Let us assume that nodes have been initialized via off-line methods or by the PAN Coordinator. Specifically, all nodes maintain two on-board SPRNGs, i.e. one SPRNG G^s to permute timeslots and one SPRNG G^c to permute channel offsets. All nodes initialize G^s to the same initial state $z^s = z_0^s$, and G^c to the same initial state $z^c = z_0^c$. Also, all nodes share the same permutation keys K^s and K^c, and consider them for the SPRNG G^s and G^c, respectively. Quantities K^s, K^c, z_0^s and z_0^c are randomly selected following the recommendations in [Schiller J. and Crocker S. 2005]. Finally, an initial link schedule pattern satisfying the Collision-Free property has been defined, and permutation vectors on nodes have been initialized accordingly. Without any loss of generality, we may assume that the initial link schedule has been defined off-line, or, alternatively, by the PAN Coordinator.

Since initialization, each node protects itself from the selective jamming attack by *periodically* performing the *Secure Link Permutation* (SLP) algorithm (Algorithm 2), i.e. once at every slotframe. In particular, the SLP algorithm first takes the two permutation vectors X_V^s and X_V^c as input, and performs the *same* (pseudo-)random permutation on both of them, relying on the SPRNG G^s (lines 2-8). At the end of this first step, vector X_V^s contains the permuted timeslot utilization pattern, and a consistent *temporary* channel offset utilization pattern is specified in vector X_V^c. The next step consists in performing an actual (pseudo-)random permutation of the channel offset utilization pattern specified in vector X_V^c, by sequentially and separately considering the active timeslots (lines 9-22). To this end, the SLP algorithm produces a vector Y, as a permutation of the first N_C natural numbers, i.e. $\{0, 1, \ldots, N_C - 1\}$, by means of the SPRNG G^c (lines 10-15). Then, it updates vector X_V^c by replacing each element $X_V^c[i]$, $i = \{0, 1, \ldots, N_S - 1\}$, with the only element $Y[j]$ in vector Y such that $j = X_V^c[i]$ (lines 16-22). After that, the SLP algorithm builds four sets, namely T_V^s, T_V^c, R_V^s, and R_V^c (lines 23-26). Specifically, either T_V^s is an empty set (if node V is a receiver-only node), or it contains the indexes of the timeslots to be used for transmission in the next slotframe. Also, T_V^c contains the channel offset values paired with the timeslots specified in T_V^s. Instead, either R_V^s is an

ALGORITHM 2: Secure Link Permutation.

1 *// By the current slotframe's expiration:*
2 *// Permute timeslots (vectors X_V^s and X_V^c)*
3 **for** $(i = 0; i < N_S; i++)$ **do**
4 n =*random*(K^s, z^s) % N_S;
5 $z^s = (z^s + 1)$;
6 swap $X_V^s[i]$ with $X_V^s[n]$;
7 swap $X_V^c[i]$ with $X_V^c[n]$;
8 **end**
9 *// Permutation of* $\{0, 1, \ldots, N_C - 1\}$
10 $Y[] = \{0, 1, \ldots, N_C - 1\}$;
11 **for** $(i = 0; i < N_C; i++)$ **do**
12 n =*random*(K^c, z^c) % N_C;
13 $z^c = (z^c + 1)$;
14 swap $Y[i]$ with $Y[n]$;
15 **end**
16 *// Permute channel offsets (entries in vector X_V^c)*
17 **for** $(i = 0; i < N_S; i++)$ **do**
18 **if** $(X_V^c[i] == N_C)$ **then**
19 **continue** // $X_V^s[i]$ is a non active timeslot
20 $j = X_V^c[i]$;
21 $X_V^c[i] = Y[j]$;
22 **end**
23 Build set T_V^s s.t. $T_V^s = \{i : X_V^s[i] = 1\}$;
24 Build set T_V^c s.t. $T_V^c = \{i : X_V^c[i] = 1\}$;
25 Build set R_V^s s.t. $R_V^s = \{j : X_V^s[j] = 2\}$;
26 Build set R_V^c s.t. $R_V^c = \{j : X_V^c[j] = 2\}$;
27 **return** T_V^s, T_V^c, R_V^s, R_V^c;

empty set (if node V is a transmitter-only node), or it contains the indexes of the timeslots to be used for reception in the next slotframe. Also, R_V^c contains the channel offset values paired with the timeslots specified in R_V^s. Finally, Algorithm 2 returns sets T_V^s, T_V^c, R_V^s and R_V^c (line 27).

In order to fix ideas, let us consider the first execution of the SLP algorithm, i.e. at the first slotframe. That is, before the slotframe ends, each node executes the SLP algorithm, specifying its permutation vectors X^s and X^c as input. Note that, when the slotframe starts, all the nodes in the network share the same permutation keys K^s and K^c, and have the related SPRNGs G^s and G^c in the same state z_0^s and z_0^c, respectively. Hence, all the nodes compute the *same* permutations, thus meeting the requirement of consistency.

Furthermore, since the permutations are based on SPRNGs, then they result unpredictable for an adversary who does not know the permutation keys. That is, the adversary cannot predict the pairs $\{timeslot, channelOffset\}$, i.e. the links used by the victim node to transmit/receive data in the next slotframe. The SLP algorithm operates only on locally available data, and thus each node can autonomously compute the permutations without exchanging

information with other nodes. Also, every execution of the SLP Algorithm causes the counter of the SPRNGs G^s and G^c to be incremented by N_S and N_C, respectively. Since all the nodes compute the same permutations, at the end of the first slotframe both the SPRNGs of all the nodes are in the same state, namely $z^s = z_0^s + N_S$ and $z^c = z_0^c + N_C$.

It follows that, in the next execution of the SLP algorithm (i.e. at the second slotframe), all the nodes compute the same permutations once again, and take their SPRNGs into the same next internal state. This reasoning can be iterated for any subsequent slotframe, i.e., after r slotframes, the internal state of the SPRNGs G^s and G^c will be $z^s = z_0^s + (r \cdot N_S)$ and $z^c = z_0^c + (r \cdot N_C)$, respectively. As it turns out, the value of the counters z^s and z^c grows at a speed that is equal to the number of timeslots N_S in a slotframe and the number of available physical channels N_C, respectively.

Note that the size of the counter of a SPRNG establishes an upper bound to the maximum length of the random output sequence that the generator is able to produce. As N_S is usually greater than N_C, the counter z^s grows (much) faster than the counter z^c. As a consequence, especially the size of the counter z^s must be adequately large, to avoid the counter to wrap-around during the network lifetime (e.g. 64-128 bits). One effective way to deal with a counter wrap-around is to refresh the associated permutation key, and then re-initialize the generator. As the internal states of all the SPRNGs G^s (and G^c) remain synchronized over time, the counter wrap-around occurs exactly at the same slotframe on all the nodes. It follows that, at that point in time, all the nodes can simultaneously and autonomously generate a new permutation key K^+ as $K^+ = E(K, K)$. Thereafter, all the nodes rely on K^+ for that SPRNG, until the next wrap-around occurs for the associated counter.

Finally, we argue that the Collision-Free property is maintained in the network.

Theorem 6.1. *The SLP algorithm maintains the Collision-Free property of the link schedule pattern, at every slotframe.*

Proof. See Appendix B. □

6.3 Node leave and join

In this section, we discuss how DISH behaves when nodes leave or join the IEEE 802.15.4e network. Upon leaving, a node U stops using all its links. Then, it informs the PAN Coordinator about its intention to leave the network, so that the released links can be assigned to some of the remaining nodes or reserved for new nodes joining the network in the future. The behavior of the remaining nodes which were not communicating with node U is not affected at all. Conversely, every node V involved in data communication with node U behaves as follows. Let us refer to T_l as the last slotframe during which node U was active. Also, let us refer to T_{l+k} as the slotframe when node V realizes that node U has left the network. Finally, we refer to S_U as the set of timeslots that node V is supposed to use to communicate with node U during slotframe T_{l+k}. Since node U is not active anymore, node V updates its own permutation vectors as $X_V^S[i] = 0$ and $X_V^C[i] = N_C$ for each $i \in S_U$. Hence, all timeslots in S_U become idle.

There are different ways for node V to realize that node U has left the network. For instance, the PAN Coordinator can explicitly notify node V that node U has left, and hence all the links between U and V are not going to be active any further. Alternatively, node V can assume that node U is not active anymore in case no successful communication with U takes place for k consecutive slotframes. As a further alternative, node U can explicitly alert node V about its own leaving from the network, by means of a dedicated flag in its last data/acknowledgment packet sent to node V.

To assure and preserve security in the network, it is necessary to provide a new pair of permutation keys $\{K_s,K_c\}$ to the remaining nodes, by excluding, and thus logically evicting, the leaving ones. This can be done by means of rekeying, i.e. by revoking the current permutation keys and distributing a new pair to all nodes but the leaving ones. DISH does not pose any particular requirement on rekeying nor mandate the adoption of any specific rekeying scheme, thus any available one can be adopted. The literature provides many rekeying schemes for wireless networks, including [Wong C. K., Gouda M. and Lam S. S. 2000][Dini G. and Savino I. M. 2011][Dini G. and Tiloca M. 2013][Tiloca M. and Dini G. 2016][Rafaeli S. and Hutchison D. 2003]. Specifying the exact rekeying mechanism to adopt is beyond the scope of this paper.

Finally, DISH allows nodes to join the network at *any* time, without particular additions to the standard joining procedure. That is, a joining node U interacts with the PAN Coordinator as usual, in order to receive a set of links that preserves collision-free communications in the network. Also, node U receives the security material to correctly execute the secure permutation of its communication pattern, as described in Section 6.2. That is, before joining the network at slotframe T_j, node U is provided with i) the shared permutation keys K^s and K^c; ii) the values z_j^s and z_j^c to initizalize the generator counters; and iii) the values z_0^s and z_0^c as the original initial states of the generators. In particular, node U considers $\{K^s, z_j^s, z_0^s\}$ and $\{K^c, z_j^c, z_0^c\}$ for permuting the timeslots and channel offsets, respectively. After that, node U completes the join process and starts using the assigned communication links.

6.4 Security analysis

In case DISH is adopted, every node in the network randomly changes its communication pattern, i.e. its N_V links, at each slotframe. Therefore, the adversary is no longer able to track her victim V, and thus cannot perform the selective jamming attack described in Section 5.

Hence, the only available strategy consists of jamming $N_J = N_V$ links, which are chosen by picking at random: i) N_V timeslots among the N_S timeslots in the slotframe; and ii) N_V associated channels among the N_C available channels. Then, at each slotframe, the adversary jams the N_J links selected at random (see Algorithm 3). Hereafter, we refer to this attack as *random jamming*.

Let us refer to l_J as one of the jammed links selected at random. In case the (originally intended) victim node V uses the link l_J, then the adversary corrupts a fraction of transmissions from node V and from all other nodes $U \neq V$ using the same jammed link l_J. On the other hand, if the link l_J is not used by the victim node V, the adversary ends up to jam transmissions from other nodes $U \neq V$ which use link l_J and whose associated receiver node is in the jammed area. It follows that the adversary corrupts a fraction of the transmissions from every *de-facto victim* node that uses any jammed link l_J, and whose associated receiver node is in the jammed area, which is the portion of the network within which no frames can be correctly received during the attack performance (see Section 4). The effectiveness of this random jamming attack carried out in the presence of DISH is discussed in more details in Section 7.

7 EVALUATION OF DISH

In order to evaluate the performance and effectiveness of DISH, we have used the open source implementation of TSCH [Contiki 2016] for the Contiki OS [Dunkels A., Grönvall B. and Voigt T. 2004], and implemented the Secure Link Permutation (Algorithm 2) and the

ALGORITHM 3: Random jamming attack against DISH.

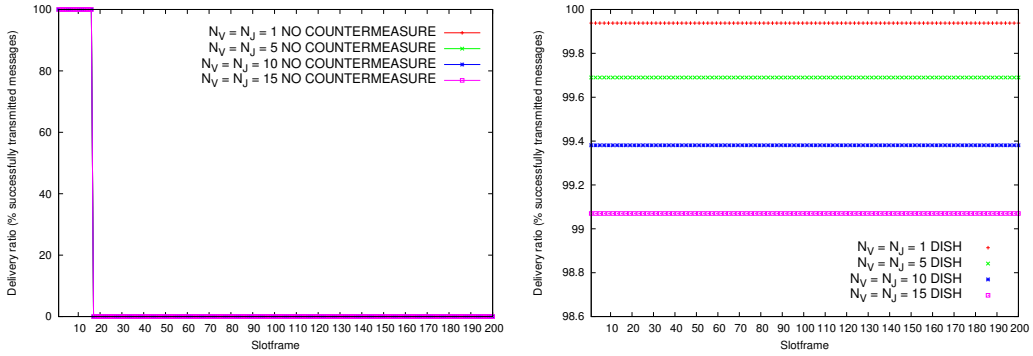
```
1  while true do
2      targets = {};
3      for (i = 0; i < N_J; i++) do
4          <randomly select timeslot s ∉ targets >;
5          targets = targets ∪ {s};
6      end
7      for each timeslot s in current_slotframe do
8          if s ∈ targets then
9              <randomly select channel c >;
10             <jam timeslot s on channel c >;
11         <move to the next timeslot>;
12     end
13     // End of current_slotframe
14 end
```

random jamming attack against DISH (Algorithm 3). Our results are derived by means of experimental tests on TelosB sensor nodes [Moteiv Corporation 2006].

In the following, we refer to a pair of nodes $\{V, R\}$. Specifically, we denote by V the victim sender node that, at each slotframe, uses N_V timeslots to transmit data messages to the recipient node R. While, in general, a node V can communicate with multiple recipient nodes, in our experiments we considered a single recipient node R, to simplify the collection and analysis of experimental results with no loss of generality. In our experiments, we referred to equally sized slotframes, whose timeslots have a duration set to $D_S = 15$ ms each. Furthermore, we refer to J as an external jammer node which is not associated to the IEEE 802.15.4e network and jams N_J timeslots at each slotframe. In particular, we assumed that $N_J = N_V$, i.e. the number of jammed timeslots is equal to the number of timeslots used by the victim node V for data transmission.

For each experiment, we performed 10 independent replications, and, for each replication, $100,000$ slotframes are considered. Besides, for each experiment, we considered different communication patterns, each consisting of randomly-generated link schedules. We averaged experimental results over all replications, and derived confidence intervals by using the independent replication method and 95% confidence level. We did not observe any particular difference when considering different nodes as V and R.

First, we evaluated how DISH results in a performance overhead which is small and affordable for sensor nodes (see Section 7.1). Then, we measured the message delivery ratio, defined as the percentage of messages sent by the victim node V and correctly received by the recipient node R during the jamming attack. Since we are considering an ideal channel, the message delivery ratio is expected to be 100%. In particular, we first referred to a typical configuration where $N_C = 16$ physical channels are available and each slotframe is composed of $N_S = 101$ timeslots [De Guglielmo D., Brienza S. and Anastasi G. 2016b] (see Section 7.2). Finally, we referred to a set of general configurations, where we considered different values of N_S, N_C and N_J (see Section 7.3).



(a) Delivery ratio with no countermeasure.

(b) Delivery ratio with DISH.

Fig. 5. Delivery ratio of victim node's messages.

7.1 Impact on performance

When DISH is used, sensor nodes do not have to exchange any additional message. Besides, DISH neither increases the size of exchanged messages, nor changes their structure or content. It follows that DISH does not display any communication overhead.

Furthermore, when using DISH, each sensor node performs $N_S + N_C$ encryptions at each slotframe, in order to generate as much pseudo-random numbers and in turn execute the DISH Secure Link Permutation algorithm. It follows that, in a typical setting where $N_S = 101$ and $N_C = 16$ [De Guglielmo D., Brienza S. and Anastasi G. 2016b], each node performs 117 encryptions at each slotframe. The considered TelosB sensor nodes are equipped with the Texas Instruments CC2420 chipset [Texas Instruments 2012], which provides AES encryptions in stand-alone mode, implemented in hardware. As reported in [Zhang F., Dojen R. and Coffey T. 2011] (Table 7), each AES encryption performed via hardware on the CC2420 chipset takes 0.3506 ms and results in an energy consumption of 26.82 μJ.

It follows that, at each slotframe, each node performs the 117 encryptions in 41.0202 ms. Since we have assumed a slotframe composed of $N_S = 101$ timeslots whose duration is $D_S = 15$ ms each, each slotframe has a total duration of 1.515 s. Hence, at each slotframe, performing the 117 encryptions takes 2.71% of the overall slotframe duration. Therefore, at each slotframe, each sensor node has plenty of time to perform the DISH Secure Link Permutation algorithm, without risking to not meet the deadlines of the timeslot utilization pattern, and thus without introducing any communication delay. We were, in fact, able to confirm this behavior when performing our experiments.

It also follows that, at each slotframe, each node performs the 117 encryptions at a total energy cost of 3.13794 mJ. When battery-powered, a TelosB sensor node uses 2 alkaline AA batteries, whose energy content is 9,360 J each [Wikipedia 2018], for a total energy content of 18,720 J. This means that, at each slotframe, performing the 117 encryptions reduces the fully-charged lifetime of a battery-powered sensor node only by the 0.000017%. Since such an energy overhead is small and affordable in extremely constrained TelosB sensor nodes, DISH is likely to result in smaller or even negligible energy overhead when used in recent sensor nodes equipped with more efficient hardware.

7.2 Effectiveness of DISH in a typical scenario

Figure 5 shows the message delivery ratio over time, for different values of N_V. We consider two different scenarios, namely: i) NO COUNTERMEASURE, where no countermeasure is used and the adversary performs the selective jamming attack described in Section 5; and ii) DISH, where the adversary performs the random jamming attack and our countermeasure described in Section 6 is used to contrast it. In both scenarios, the adversary starts performing the attack at slotframe 0. With reference to Figure 5a, we observe that, when no countermeasure is adopted, the delivery ratio is always 100% for the first 16 slotframes, during which the adversary only listens to the wireless medium and derives the victim's links. Then, starting from slotframe 16, the adversary performs the selective jamming attack described in Section 5 and entirely thwarts the victim's transmissions. That is, from then on, the delivery ratio is 0% at every slotframe.

Instead, as shown in Figure 5b, when using DISH to contrast the selective jamming attack, the experienced delivery ratio is always slightly below 100% and never decreases. That is, at every slotframe, almost all the data message sent by the victim node V are correctly received by the associated receiver R. Besides, the number of timeslots N_V considered by node V has only a negligible impact, i.e. the delivery ratio is 99.94% and 99.07% in case $N_V = 1$ or $N_V = 15$ are considered, respectively. This is consistent with the expected limited amount of messages that can be corrupted anyway, due to a lucky guess of the adversary or to transmission errors of the victim node. Therefore, results confirm that DISH effectively contrasts selective jamming attacks in IEEE 802.15.4e networks.

During the experiments, all the other sender nodes different than V experience a delivery ratio very close to 100% at every slotframe, both in the NO COUNTERMEASURE and in the DISH scenario. However, with respect to the NO COUNTERMEASURE scenario, in the DISH scenario we could observe a slight reduction in the fraction of corrupted messages experienced by other nodes different from V and using the victim's timeslots. This is because, in the presence of DISH, the adversary corrupts a significantly lower fraction of transmissions on those timeslots. Finally, with respect to the NO COUNTERMEASURE scenario, in the DISH scenario we also observed that nodes using timeslots different from the victim's ones experience a slightly higher fraction of corrupted messages. This is because, in the presence of DISH, the transmissions from such nodes might be corrupted by the adversary, if they occur during the jammed timeslot and on the channel offset value chosen at random.

7.3 Effectiveness of DISH in a general scenario

In the following, we consider a network scenario where our countermeasure is adopted, and evaluate the effectiveness of DISH for different values of N_J and N_C. In particular, we investigate how DISH performs in case the adversary randomly jams a number of timeslots $N_J \leq N_S$ higher than the number of victim's timeslots N_V, i.e. when $N_J \geq N_V$. Also, we investigate the case where a number of physical channels $N_C < 16$ is available, as, for instance, some of them are reserved or have been blacklisted. To this end, we derived an analytical expression of the delivery ratio DR_V.

Let us define by P_i the probability that the adversary performing a random jamming attack successfully jams i victim's timeslots during a slotframe T, with $0 \leq i \leq min(N_J, N_V)$. Also, we define $x_{min} = max(0, N_J - N_V)$, $x_{max} = min(N_S - N_J, N_J - i)$, and $y = (N_J - i - x)$. Then, the following claim holds.

THEOREM 7.1.

$$P_i = \frac{\binom{N_V}{i} \sum_{x=x_{min}}^{x_{max}} \binom{N_S-N_V}{x} \cdot {N_C}^x \cdot \binom{N_V-i}{y} \cdot (N_C-1)^y}{\binom{N_S}{N_J} \cdot {N_C}^{N_J}} \tag{4}$$

PROOF. See Appendix C. □

Therefore, the delivery ratio DR_V is equal to

$$DR_V = \frac{N_V - (\sum_{i=1}^{min(N_V,N_J)} i \cdot P_i)}{N_V} \cdot 100 \tag{5}$$

In order to validate analytical results obtained through Equation 5, we relied on simulation experiments and referred to the same methodology used to perform the experimental evaluation. Since probability P_i becomes computationally infeasible to compute for slotframes composed of $N_S = 101$ timeslots, we validated Equation 5 and produced a first set of results considering $N_S = 31$. Obtained analytical and simulation results almost overlap.

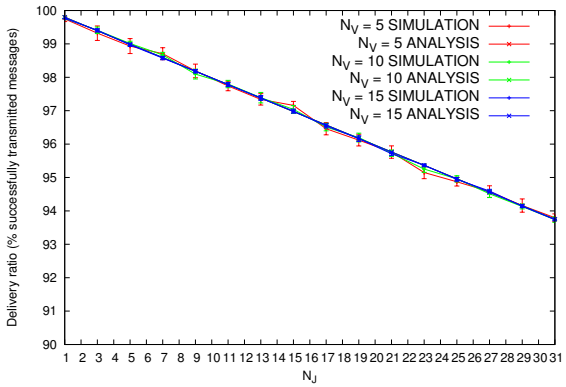


Fig. 6. Delivery ratio ($N_S = 31$, $N_C = 16$).

Figure 6 shows the message delivery ratio, for different values of N_J. First of all, we can see that the delivery ratio decreases for greater values of N_J, i.e. when the adversary jams a greater number of timeslots per slotframe. However, we observe that the curves associated to different values of N_V almost overlap. That is, when the victim node V uses a greater number of timeslots, the adversary jams a greater number of messages transmitted by V, but with no particular impact on the overall message delivery ratio. Furthermore, if the adversary jams only 1 timeslot per slotframe (i.e. $N_J = 1$), she has to guess one of the timeslots used by node V and the associated channel offset value. Thus, the resulting delivery ratio reaches 99.8%. On the other hand, if the adversary jams *all* timeslots at every slotframe, i.e. $N_J = N_S = 31$, then she only has to guess the victim's channel offset at each timeslot. Even in such (worst-)case, the delivery ratio reaches 93.75%. Note that, if N_J is comparable with the slotframe size N_S, the considered attack is actually a form of constant jamming, since the adversary jams every single timeslot, at every slotframe. This makes the adversary much easier to detect, and sensibly contributes to her energy consumption throughout the attack performance.

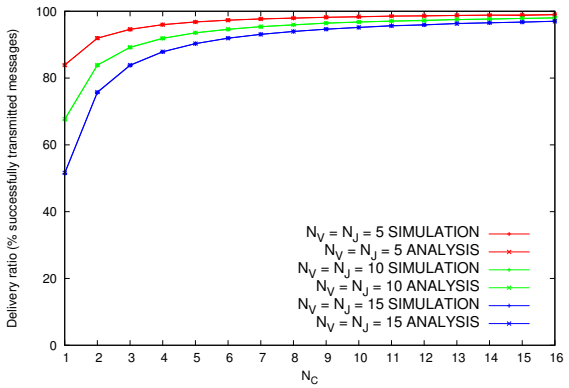


Fig. 7. Delivery ratio ($N_S = 31$).

Figure 7 shows the message delivery ratio, for different values of N_C. Also, we refer to different numbers of jammed timeslots N_J, and consider $N_J = N_V$. First of all, we can see that the delivery ratio increases for greater values of N_C. This is consistent with the fact that, if more channels are available, the adversary has less chances to guess the correct channel offset associated to a timeslot used by the victim node. Also, we can observe that, given N_C available channels, the delivery ratio is lower for greater values of N_J. This is consistent with the fact that, if more timeslots are jammed, the adversary has more chances to guess the timeslots actually used by the victim node at the current slotframe. However, the value of N_J impacts less and less as a greater number N_C of channels are considered. In fact, as discussed above, the adversary has less chances to guess the correct channel offset associated to a jammed timeslot, in case more channels are available, and thus she has less chances to successfully jam a victim's transmission. Furthermore, in the traditional setup where $N_C = 16$ channels are available, the delivery ratio is always greater than 96.97%. Instead, even in the uncommon (worst-)case when $N_C = 1$, i.e. only 1 channel is available, DISH makes it possible to achieve a delivery ratio equal to 51.61% (83.87%) when $N_J = 15$ ($N_J = 5$).

Finally, we present equivalent results with reference to a slotframe composed of $N_S = 101$ timeslots. Results are reported in Figures 8 and 9, and are obtained through simulation experiments, as it was computationally infeasible to compute probability P_i when $N_S = 101$.

Results reported in Figures 8 and 9 display the same trends observed in Figures 6 and 7, respectively, hence similar considerations hold. However, we can see that, given the presence of slotframes whose size is $N_S = 101$ timeslots, the experienced delivery ratio is even higher. In fact, the adversary has much less chances to guess the timeslots used by the victim node, as well as, of course, the associated channel offset values. In particular, Figure 8 shows a slight improvement for small values of N_J, i.e. the delivery ratio reaches 99.98% if the adversary jams only 1 timeslot per slotframe. Furthermore, Figure 9 shows a considerable improvement with respect to the scenario considered in Figure 7. That is, in the traditional setup where $N_C = 16$ channels are available, the delivery ratio is always greater than 98.99%. Instead, even in the uncommon (worst-)case when $N_C = 1$, i.e. only 1 channel is available, DISH makes it possible to achieve a delivery ratio equal to 85.12% (94.67%) when $N_J = 5$ ($N_J = 15$).

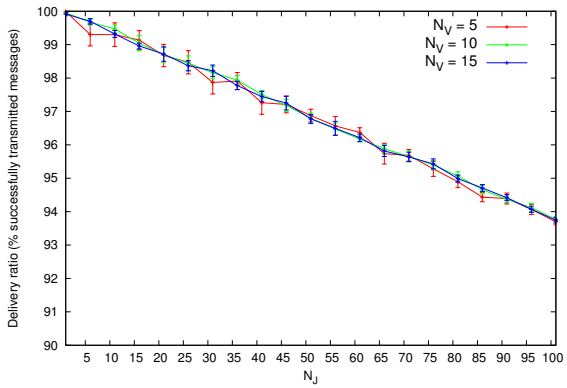


Fig. 8. Delivery ratio ($N_S = 101$, $N_C = 16$).

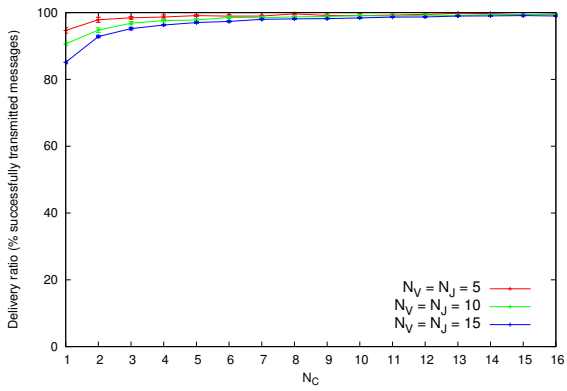


Fig. 9. Delivery ratio ($N_S = 101$).

8 CONCLUSION

In this paper, we have shown that the Time Slotted Channel Hopping (TSCH) mode of the IEEE 802.15.4e MAC amendment is particularly subject to selective jamming attack. In particular, we have highlighted how an adversary can easily and efficiently derive a victim node's communication pattern even though the IEEE 802.15.4e security services are adopted, and then completely thwart all victim's transmissions at each slotframe. Furthermore, we have proposed DISH, a countermeasure to the selective jamming attack according to which network nodes permute their own communication pattern at each slotframe in a completely distributed yet consistent way, so maintaining a collision-free link schedule and without requiring any additional message exchange. We have implemented DISH for the Contiki OS and tested its effectiveness on TelosB sensor nodes. Quantitative analysis for different network configurations shows that DISH effectively contrasts selective jamming with negligible penalties on performance. The proposed countermeasure has been conceived for TSCH. However, the same solution can be adapted with minor modifications to DSME (Deterministic and Synchronous Multi-Channel Extension), both in *channel adaptation* and *channel hopping* mode.

As to future works, we envision the following research directions. First, it would be interesting to assess the trade-off between security and network performance in case DISH pseudo-randomly permutes only the channel offset utilization pattern. In fact, keeping the timeslot utilization pattern fixed as originally scheduled would increase the effectiveness of the random jamming attack. On the other hand, it would also preserve the intent of scheduling policies designed to achieve limited and predictable end-to-end latencies in multi-hop networks. Second, DISH makes also the transmission of Beacons non-deterministic for non-members of the network. Hence, it would be interesting to assess how this affects new nodes attempting to join the network, as well as the overall duration and performance of the join process.

ACKNOWLEDGMENTS

The authors sincerely thank the anonymous referees and the associate editor for their insightful comments and suggestions. This project has been funded by the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no. 607109. This work has also been supported by the NSF grants CNS-1545037, DGE-1433659, CNS-154050, and NeTS-1818942, the EIT Digital High Impact Initiative project ACTIVE, the PRIN project TENACE (n. 20103P34XC) funded by the Italian Ministry of Education, University and Research, and the University of Pisa (PRA 2015 program). The authors would also like to thank Daniele Carmignani for his valuable help.

REFERENCES

Ashraf F., Hu Y.-C. and Kravets R. H. 2012. Bankrupting the jammer in WSN. In *Proceedings of the IEEE 9th International Conference on Mobile Adhoc and Sensor Systems*. 317–325.

Chiang J. T. and Hu Y.-C. 2011. Cross-layer jamming detection and mitigation in wireless broadcast networks. *IEEE/ACM Transactions on Networking* 19, 1 (2011), 286–298.

Contiki. 2016. EIT-ICT-RICH/contiki. https://github.com/EIT-ICT-RICH/contiki. (2016).

Daidone R., Dini G. and Tiloca M. 2011. On experimentally evaluating the impact of security on IEEE 802.15.4 networks. In *Proceedings of International Conference on Distributed Computing in Sensor Systems and Workshops*. 1–6.

Daidone R., Dini G. and Tiloca M. 2013. A solution to the GTS-based selective jamming attack on IEEE 802.15.4 networks. *Wireless Networks* 20, 5 (2013), 1223–1235.

De Guglielmo D., Brienza S. and Anastasi G. 2016a. IEEE 802.15.4e: a survey. *Computer Communications* 88 (2016), 1–24.

De Guglielmo D., Brienza S. and Anastasi G. 2016b. A model-based beacon scheduling algorithm for IEEE 802.15.4e TSCH networks. In *2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. 1–9.

Dini G. and Savino I. M. 2011. LARK: a Lightweight Authenticated ReKeying scheme for clustered wireless sensor networks. *ACM Transactions on Embedded Computing Systems* 10, 4, Article 41 (2011), 3 pages.

Dini G. and Tiloca M. 2013. HISS: a HIghly Scalable Scheme for group rekeying. *The Computer Journal* 56, 4 (2013), 508–525.

Dunkels A., Grönvall B. and Voigt T. 2004. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*. IEEE, 455–462.

IEEE Computer Society. 2011. *IEEE standard for local and metropolitan area networks, Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)*.

IEEE Computer Society. 2012. *802.15.4e-2012 - IEEE standard for local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer*.

Jones K., Wadaa A., Olariu S., Wilson L. and Eltoweissy M. 2003. Towards a new paradigm for securing wireless sensor networks. In *Proceedings of the 2003 workshop on New security paradigms*. ACM, 115–121.

Knuth D. E. 1998. *The art of computer programming, Volume 3: sorting and searching, 2nd Edition*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA.

Lazos L., Liu S. and Krunz M. 2009. Mitigating control-channel jamming attacks in multi-channel ad hoc networks. In *Proceedings of the second ACM conference on Wireless network security*. ACM, 169–180.

Lu Z., Wang W. and Wang C. 2014. Modeling, evaluation and detection of jamming attacks in time-critical wireless applications. *IEEE Transactions on Mobile Computing* 13, 8 (2014), 1746–1759.

Mansour I., Chalhoub G. and Quilliot A. 2011. Security architecture for wireless sensor networks using frequency hopping and public key management. In *Networking, Sensing and Control (ICNSC), 2011 IEEE International Conference on*. IEEE, 526–531.

Menezes A. J., van Oorschot P. C. and Vanstone S. A. 2001. *Handbook of applied cryptography*. CRC Press.

Moteiv Corporation. 2006. *Tmote iv low power wireless sensor module*.

Mustafa H., Zhang X., Liu Z., Xu W. and Perrig A. 2012. Jamming-resilient multipath routing. *IEEE Transactions on Dependable and Secure Computing* 9, 6 (2012), 852–864.

National Institute of Standards and Technology. 2001. *Federal Information Processing Standards, specification for the ADVANCED ENCRYPTION STANDARD (AES)*. National Institute of Standards and Technology.

Paar C. and Pelzl J. 2010. *Understanding cryptography*. Springer.

Popper C., Strasser M. and Čapkun S. 2010. Anti-jamming broadcast communication using uncoordinated spread spectrum techniques. *IEEE Journal on Selected Areas in Communications* 28, 5 (2010), 703–715.

Proaño A. and Lazos L. 2010. Selective jamming attacks in wireless networks. In *Proceedings of the 2010 IEEE International Conference on Communications (ICC 2010)*. IEEE Computer Society.

Proaño A. and Lazos L. 2012. Packet-hiding methods for preventing selective jamming attacks. *IEEE Transactions on Dependable and Secure Computing* 9, 1 (2012), 101–114.

Rafaeli S. and Hutchison D. 2003. A survey of key management for secure group communication. *ACM Computing Surveys* 35, 3 (2003), 309–329.

Richa A., Scheideler C., Schmid S. and Zhang J. 2013. An efficient and fair MAC protocol robust to reactive interference. *IEEE/ACM Transactions on Networking (ToN)* 21, 3 (2013), 760–771.

Schiller J. and Crocker S. 2005. *Randomness requirements for security*. Internet Engineering Task Force, Fremont, CA, USA.

Sokullu R., Dagdeviren O. and Korkmaz I. 2008. On the IEEE 802.15. 4 MAC layer attacks: GTS attack. In *Sensor Technologies and Applications, 2008. SENSORCOMM'08. Second International Conference on*. IEEE, 673–678.

Sokullu R., Korkmaz I. and Dagdeviren O. 2009. GTS attack: an IEEE 802.15.4 MAC layer attack in wireless sensor networks. *Int.l Journal On Advances in Internet Technologies* 2, 1 (2009), 104–114.

Spuhler M., Giustiniano D., Lenders V., Wilhelm M. and Schmitt J. B. 2014. Detection of reactive jamming in DSSS-based wireless communications. *IEEE Transactions on Wireless Communications* 13, 3 (2014), 1593–1603.

Stojanovski S. and Kulakov A. 2015. Efficient attacks in industrial wireless sensor networks. In *ICT Innovations 2014*. Advances in Intelligent Systems and Computing, Vol. 311. Springer International Publishing, 289–298.

Strasser M., Danev B. and Čapkun S. 2010. Detection of reactive jamming in sensor networks. *ACM Transactions on Sensor Networks* 7, 2 (2010), 16.

Texas Instruments. 2012. CC2420 2.4 GHz IEEE 802.15.4 / ZigBee ready RF transceiver. http://focus.ti.com/lit/ds/symlink/cc2420.pdf. (2012).

Tiloca M. and Dini G. 2016. GREP: a Group REkeying Protocol based on member join history. In *Proceedings of the twenty-first IEEE Symposium on Computers and Communications (ISCC 2016)*. IEEE, 326–333.

Tiloca M., De Guglielmo D., Dini G., Anastasi G. and Das S. K. 2017. JAMMY: a distributed and self-adaptive solution against selective jamming attack in TDMA WSNs. *IEEE Transactions on Dependable and Secure Computing* 14, 4 (July/August 2017), 392–405.

Tiloca M., De Guglielmo D., Dini G. and Anastasi G. 2013. SAD-SJ: a Self-Adaptive Decentralized solution against Selective Jamming attack in wireless sensor networks. In *Proceedings of the 18th IEEE International Conference on Emerging Technology & Factory Automation*. IEEE Computer Society, 1–8.

Wikipedia. 2018. Energy density. https://en.wikipedia.org/wiki/Energy_density. (2018).

Wilhelm M., Martinovic I., Schmitt J. B. and Lenders V. 2011. Short paper: reactive jamming in wireless networks: how realistic is the threat?. In *Proceedings of the fourth ACM conference on Wireless network security*. ACM, 47–52.

Wong C. K., Gouda M. and Lam S. S. 2000. Secure group communications using key graphs. *IEEE/ACM Transactions on Networking* 8, 1 (2000), 16–30.

Wood A. D., Stankovic J. A. and Zhou G. 2007. DEEJAM: Defeating Energy-Efficient Jamming in IEEE 802.15.4-based wireless networks. In *Proceedings of the 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*. IEEE Computer Society, 60–69.

Xu W., Ma K., Trappe W. and Zhang Y. 2006. Jamming sensor networks: attack and defense strategies. *IEEE Network* 20, 3 (2006), 41–47.

Xu W., Trappe W., Zhang Y. and Wood T. 2005. The feasibility of launching and detecting jamming attacks in wireless networks. In *Proceedings of the 6th ACM International Symposium on Mobile ad hoc Networking and Computing*. ACM,

46–57.

Xu W., Wood T., Trappe W. and Zhang Y. 2004. Channel surfing and spatial retreats: defenses against wireless denial of service. In *Proceedings of the 3rd ACM Workshop on Wireless Security (WiSe '04)*. ACM, 80–89.

Zhang F., Dojen R. and Coffey T. 2011. Comparative Performance and Energy Consumption Analysis of Different AES Implementations on a Wireless Sensor Network Node. *Internatioanl Journal of Sensor Networks* 10, 4 (October 2011), 192–201.

A OPTIMIZED PSEUDO RANDOM PERMUTATION

In Section 6, we considered the Knuth shuffle algorithm to perform a random permutation of timeslots and channel offets assigned to active sensor nodes at each slotframe. In the following, we present an optimized permutation function, which allows sensor nodes to store two vectors of N_V elements, rather than two vectors of N_S elements. Since the number of timeslots N_V considered by a given sensor node is typically much smaller than the slotframe size N_S, this makes it possible to considerably limit the memory occupancy on sensor nodes.

Let us consider a sensor node V and N_V associated links. Then, node V maintains a list of N_V communication directions, such that the i-th element of the list specifies whether V uses its i-th link to transmit or receive data. This list of communication directions remains unchanged at every slotframe. Also, node V maintains two vectors of N_V elements, namely X_V^s and X_V^c. In particular, X_V^s is used to permute timeslots, and includes only the indexes of the N_V timeslots considered by node V. Similarly, X_V^c is used to permute channel offsets, and includes only the channel offset values used in the N_V considered timeslots. Specifically, $X_V^c[i]$ indicates the channel offset value associated to timeslot $X_V^s[i]$, $i = \{0, 1, \ldots, (N_S - 1)\}$.

ALGORITHM 4: Optimized vector permutation.

```
1  unsigned* optimized_permute (unsigned* old_vector,
2                               unsigned vector_size,
3                               unsigned domain_size,
4                               unsigned K,
5                               unsigned z)
6  {
7  unsigned i, j, n;
8  unsigned new_vector[vector_size];
9  for (i = 0; i < vector_size; i++) do
10 |  new_vector[i] = old_vector[i];
11 end
12 for (i = domain_size; i > 0; i--) do
13 |  n = (random(K,z) % i);   // See Algorithm 1
14 |  for (j = 0; j < vector_size; j++) do
15 |  |  if (i == (new_vector[j] +1)) then
16 |  |  |  new_vector[j] = n;
17 |  |  else if (n == new_vector[j]) then
18 |  |  |  new_vector[j] = i - 1;
19 |  end
20 |  z = (z + 1);
21 end
22 return new_vector;
23 }
```

In order to compute its communication pattern for the next slotframe, node V relies on the Algorithm 4 shown above, and separately permutes the two vectors X_V^s and X_V^c at every slotframe. That is, node V invokes $optimized\_permute(X_V^s, N_V, N_S, K^s, z^s)$ in order to permute its N_V timeslots, and $optimized\_permute(X_V^c, N_V, N_C, K^c, z^c)$ in order to permute the

associated channel offset values. Then, the counters z^s and z^c are updated as $z^s = (z^s + N_S)$ and $z^c = (z^c + N_C)$. Once the two permutations have been completed, the i-th value in the permuted vector X^s_V and the i-th value in the permuted vector X^c_V indicate the timeslot and channel offset, respectively, that node V has to consider for the i-th link in the next slotframe.

B PROOF OF CLAIM 1

Claim 1. *The SLP algorithm maintains the Collision-Free property of the link schedule pattern, at every slotframe.*

Proof. Let us define $\mathcal{L} = (L_1, \ldots, L_N)$ where L_i is the set of non-interfering links that use timeslot s_i in the current slotframe. Consider a link $(u, t) \in L_i$ such that, to fix ideas, $X^s_u[i] = 1$, $X^s_t[i] = 2$, and $X^c_u[i] = X^c_t[i] = c$.

Let Π_1 be the same first permutation that u and t compute at the current slotframe (see Algorithm 2, lines 2-8). In particular, let us define $\bar{X^s_u}$ and $\bar{X^s_t}$ as the vectors resulting from applying Π_1 to X^s_u and X^s_t, respectively. Also, let us define $\bar{X^c_u}$ and $\bar{X^c_t}$ as the vectors resulting from applying Π_1 to X^c_u and X^c_t, respectively. Then, let us assume that Π_1 maps the i-th element into the j-th element. It follows that $\bar{X^s_u}[j] = 1$ and $\bar{X^s_t}[j] = 2$, that is, link (u, t) becomes active during timeslot s_j in the next slotframe. Also, it follows that $\bar{X^c_u}[j] = \bar{X^c_t}[j] = c$, that is, nodes u and t consistenly consider the same channel offset c when using link (u, t) in the next slotframe.

As all nodes compute the same permutation Π_1, then all links using timeslot s_i and channel offset c during the current slotframe will be active during timeslot s_j and consider channel offset c in the next slotframe. This means that L_i is permuted into L_j. More generally, this means that permutation Π_1 randomly permutes the elements in $\mathcal{L}$. As a consequence, every link becomes associated with a different set of non-interfering links. Thus, after permutation Π_1 has been performed, $\mathcal{L}$ still maintains the Collision-Free property.

After that, u and t compute the same permutation Π_2 of the first N_C natural numbers, i.e. $\{0, 1, \ldots, N_C - 1\}$ (see Algorithm 2, lines 10-15). Let us define Y as the original vector $\{0, 1, \ldots, N_C - 1\}$, and $\bar{Y}$ as the vector resulting from applying Π_2 to Y. Then, u and t consider the channel offset values associated to the timeslots active in the next slotframe. Specifically, u (t) separately considers all the elements of $\bar{X^c_u}$ ($\bar{X^c_t}$) whose value is $c \neq N_C$. Then, u (t) assigns to each of these elements the value c' such that $\bar{Y}[c] = c'$. That is, u and t execute a bijective function $f : C \rightarrow C$, $C = \{0, 1, \ldots, N_C - 1\}$, on all the channel offset values associated to links active in the next slotframe (see Algorithm 2, lines 16-22).

As all nodes compute the same permutation Π_2, then all the links that after permutation Π_1 refer to timeslot s_j and channel offset c, are updated in order to be active during timeslot s_j and considering channel offset c', in the next slotframe. More generally, this means that Π_2 randomly permutes the channel offset values, according to a random mapping based on a one-to-one correspondence, and separately for each timeslot used by at least one link in the next slotframe. Besides, Π_2 does not permute the elements in $\mathcal{L}$. That is, it does not permute vector X^s, and thus does not alter the timeslot schedule produced by permutation Π_1. Therefore, the SLP algorithm maintains the Collision-Free Property of the link schedule pattern, at every slotframe. □

C PROOF OF CLAIM 2

Claim 2.

$$P_i = \frac{\binom{N_V}{i} \sum_{x=x_{min}}^{x_{max}} \binom{N_S-N_V}{x} \cdot {N_C}^x \cdot \binom{N_V-i}{y} \cdot (N_C-1)^y}{\binom{N_S}{N_J} \cdot {N_C}^{N_J}} \tag{4}$$

Proof. Let us consider Equation 4 as the probability P_i that an adversary performing a random jamming attack scores i *hits*, with $0 \leq i \leq min(N_J, N_V)$, i.e. she successfully jams i victim's timeslots during a slotframe. We define: i) x as the number of times where the adversary scores a *miss*, i.e. she performs jamming by considering a pair {*timeslot*, *channel offset*} which is not used by the victim node; and ii) $y = (N_J - i - x)$. Hence, Equation 4 can be explained as follows.

The probability P_i is computed as the ratio between i) the number of attack configurations where i hints and at most $(N_J - i)$ misses are produced; and ii) the number of all possible attack configurations in the presence of an adversary that jams N_J timeslots.

As to the numerator, it is in turn composed of two elements. The first element $\binom{N_V}{i}$ before the summation represents all possible configurations where i hints are scored over the N_V victim's timeslots. The second element is the summation, and represents all possible configurations where at most $(N_J - i)$ misses x are scored. Specifically, the two factors $\binom{N_S-N_V}{x} \cdot {N_C}^x$ and $\binom{N_V-i}{y} \cdot (N_C-1)^y$ in the summation are the two contributions that concur to score the x misses.

That is, the first factor $\binom{N_S-N_V}{x} \cdot {N_C}^x$ represents all possible configurations where the adversary scores a miss, because she jams one of the $(N_S - N_V)$ timeslots not used by the victim node, regardless the considered channel offsets. In particular, the term ${N_C}^x$ represents all the possible sets of x channel offsets considered to jam x timeslots that are not used by the victim node. Then, all possible combinations of such x jammed timeslots over the $(N_S - N_V)$ timeslots not used by the victim node in the slotframe, i.e. $\binom{N_S-N_V}{x}$, are considered.

Instead, the second factor $\binom{N_V-i}{y} \cdot (N_C-1)^y$ represents all possible configurations where the adversary scores a miss, because she jams one of the $(N_V - i)$ timeslots used by the victim node, but on a different channel offset than the associated one. In particular, the term $(N_C-1)^y$ represents all the possible sets of $y = (N_J - i - x)$ channel offsets considered to jam y timeslots used by the victim node but on a different channel offset than the considered one, and thus not succesfully jammed. Then, all possible combinations of such y jammed timeslots over the $(N_V - i)$ timeslots used by the victim node in the slotframe and not succesfully jammed, i.e. $\binom{N_V-i}{y}$, are considered.

The values of x considered in the sum are in the range between $x_{min} = max(0, N_J - N_V)$ and $x_{max} = min(N_S - N_J, N_J - i)$. As to x_{min}, the number of misses x cannot be i) lower than 0; or ii) lower than $(N_J - N_V)$, i.e. N_V hits are scored if $N_J \geq N_V$ and the adversary successfully jams all the N_V victim's timeslots. As to x_{max}, the number of misses x cannot be i) greater than $(N_J - i)$, since the adversary jams N_J timeslots; or ii) greater than $(N_S - N_J)$, where N_S is the slotframe size in timeslots.

As to the denominator, the term ${N_C}^{N_J}$ represents all the possible sets of N_J channel offsets considered to jam N_J timeslots. Then, all possible combinations of N_J jammed timeslots over the total N_S timeslots in the slotframe, i.e. $\binom{N_S}{N_J}$, are considered. □

Received ; revised ; accepted