# Jamming-Resilient Channel Hopping for WirelessHART Networks

Elisabeth K. A. Permatasari
Iowa State University, USA

Abusayeed Saifullah
Wayne State University, USA

Haibo Zhang
University of Otago, NZ

## ABSTRACT

Time Synchronized Channel Hopping (TSCH) is a key feature in existing industrial Internet of Things wireless standards such as WirelessHART to achieve reliable and real-time communication. However, the current TSCH strategy adopted in WirelessHART can produce strong repetitive channel usage, making it vulnerable to channel cracking and jamming attacks. Developing a jamming-resilient channel hopping strategy for WirelessHART networks poses significant challenges due to the need to ensure fast channel switching (within 0.192 ms), channel synchronization between senders and receivers, and interference-free concurrent transmissions. In this paper, we present a jamming-resilient channel hopping mechanism for WirelessHART that meets these requirements. Our approach leverages multi-level randomness to reduce repetitive channel hopping patterns, thereby enhancing resistance to jamming attacks. We have implemented our approach in Contiki-NG and evaluated it through both testbed experiments and simulations in Cooja. Our results demonstrate that this new channel hopping strategy markedly reduces the effectiveness of a jammer's ability to predict channel patterns, without increasing the channel hopping overhead compared to the current WirelessHART approach.

## CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Sensor networks*; • **Networks** → Network reliability.

## KEYWORDS

Industrial Internet of Things, WirelessHART, channel hopping, jamming resilience

**ACM Reference Format:**
Elisabeth K. A. Permatasari, Abusayeed Saifullah, and Haibo Zhang. 2025. Jamming-Resilient Channel Hopping for WirelessHART Networks. In *26th International Conference on Distributed Computing and Networking (ICDCN 2025), January 04–07, 2025, Hyderabad, India.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3700838.3700852

## 1 INTRODUCTION

As a driving force behind Industry 4.0, the Industrial Internet of Things (IIoT) has gained significant popularity in manufacturing and process control applications in recent years. Ensuring reliable and real-time communication between sensors and actuators is crucial for the proper functioning of these applications. Several wireless sensor-actuator network (WSAN) standards, such as WirelessHART [1], ISA100 [2], and WIA-PA [35], have been specifically developed for industrial control applications to provide reliable and real-time communication over unreliable wireless links in industrial environments. Among these standards, WirelessHART [1] is the predominant and widely adopted standard for process management worldwide [27]. To meet real-time and reliability requirements in highly unreliable industrial environment, WirelessHART leverages multi-channel Time-Division-Multiple-Access (TDMA), route diversity, channel blacklisting, and channel hopping [9].

Channel hopping is an important feature in WirelessHART for ensuring reliable communication and mitigating jamming attacks. Currently, WirelessHART implements time-synchronized channel hopping (TSCH) by letting each transmitter and receiver derive the channel for communication using a modular function based on a predefined channel sequence. However, the length of the predefined channel sequence is limited by the total number of active channels, which causes the modular function to produce a strong repetitive channel usage. As a result, a jammer may derive the channel usage pattern through a small effort by listening to the active links without knowing the original sequence. The adoption of harmonic superframe (a communication pattern that repeats after a certain interval) lengths further makes such a channel cracking easier. A recent study has shown that cracking channel hopping sequences can be performed easily by observing the channel usage alone [10]. Once the pattern of channel usage is figured out, a selective jamming attack can be launched easily [12], posing significant threats to industrial process control applications. Existing channel hopping strategies in other wireless domains primarily focus on un-slotted networks [4, 7, 13, 16, 20, 21, 23, 32] and cognitive radio networks [5, 6, 8, 22, 31]. In addition, several anti-jamming solutions addressing selective jamming attack for TSCH-based and WirelessHART networks have been explored in [19, 28–30, 36]. These strategies, however, are not specifically designed to meet the high reliability and real-time requirements of WirelessHART networks.

Designing a jamming-resilient channel hopping strategy for WirelessHART networks is indeed challenging as it must guarantee fast channel switching (within 0.192 ms as mandated by the WirelessHART standard [1]), channel synchronization between sender-receiver pairs, and interference-free concurrent transmissions in the network. In this paper, we propose a jamming-resilient channel hopping mechanism for WirelessHART networks that can meet these requirements and be applied to other TSCH based networks such as ISA100. The **key idea** of our approach is to exploit randomness at multiple levels to reduce the strong periodic pattern in channel hopping sequence, making it hard for a jammer to predict and crack the channel usage. Such a high degree of randomness together with the modular function can easily puzzle a jammer,


*ICDCN 2025, January 04–07, 2025, Hyderabad, India*

ACM ISBN 979-8-4007-1062-9/25/01
https://doi.org/10.1145/3700838.3700852

making it quite difficult to crack the channel hopping sequence within a feasible amount of time or effort. Specifically, our approach uses two channel sequences, where each sender-receiver pair uses an alternation sequence to dynamically determine the channel sequence to be used for their communication. Another level of randomness is introduced by using a pointer sequence to choose a random starting index for the active channel sequence.

While theory-based design will give more fundamental understanding, our design principle is to seamlessly integrate with the hopping mechanism in WirelessHART to maintain compatibility with existing systems while enhancing security. Compared to existing complex algorithms, our method is very simple yet effective, capable of online execution with very low memory requirement, making it well-suited for WirelessHART devices which are resource-constrained. We have implemented the proposed jamming-resilient channel hopping approach in Contiki-NG [25]. Testbed experiments have been conducted in an indoor deployment with 16 TelosB devices [3], and we also evaluate our scheme in large-scale networks through simulations in Cooja [26]. Both the experimental and simulation results show that our proposed approach can significantly lower a jammer's capability in making correct channel prediction.

The remainder of this paper is structured as follows. In Section 3, we review the related work. Section 2 describes the background and system model. Section 4 presents the proposed channel hopping technique. Sections 5 presents the experimental and simulation results. Finally, Section 6 concludes the paper.

## 2 BACKGROUND AND SYSTEM MODEL

WirelessHART adopts TDMA as its medium access control (MAC) protocol to ensure collision-free and deterministic communications. Time is slotted and synchronized. Each transmission and its associated acknowledgement (ACK) need one time slot. A communication pattern that repeats over time in the network is defined as *superframe*. Because of the use of TDMA, the knowledge of global clock becomes vital for each node to determine the start and the end of its communication time slot. Therefore, each node in the network maintains the *absolute slot number* (ASN), which is defined as the total number of time slots elapsed since the start of the network.

WirelessHART devices are built upon the IEEE 802.15.4 physical layer which provides channel diversity. Originally, there are 16 different channels available to support communications. Some of these channels may be blacklisted due to excessive noise. To improve communication reliability and mitigate jamming, WirelessHART adopts TSCH, in which each pair of communicating devices uses one channel in the current slot and hops to a different channel in the next slot. As stated in the standard, only one pair of devices communicates on a particular channel in a time slot due to the need to ensure interference-free concurrent transmissions [1]. Moreover, channel hopping in WirelessHART needs to be extremely lightweight as the standard mandates a maximum channel switching time of 0.192 ms [1].

TSCH in WirelessHART uses a random channel sequence shared among all nodes prior to network operation. Each communication link can be represented as a tuple $\{slot\_index, channel\_offset\}$, where $slot\_index$ is the slot index in a superframe in which the link is schedule for transmission and $channel\_offset$ is the allocated logical channel. To derive the hopping sequence, each node must keep track of the list of active channels ($A$). For a given $channel\_offset$, the index for the corresponding physical channel in $A$ ($ch\_id$) is computed using the following modular function:

$$ch\_id = (channel\_offset + ASN) \bmod s\_length \quad (1)$$

where $s\_length$ denotes the number of active channels. Once the node knows its channel index, the physical channel to be used for communication can be looked up from $A$ as follows:

$$ch = A[ch\_id]. \quad (2)$$

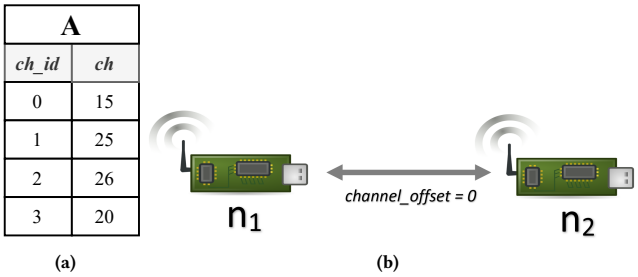



**Figure 1: The table in (a) represents channel mapping $A$ with 4-active channels. The left and right columns of the table are the channel index and the physical channel, respectively. (b) Simple network topology consisting 2 sensors, namely $n_1$ and $n_2$ and 1 link connecting $n_1$ and $n_2$ with $channel\_offset = 0$.**

Currently, TSCH uses a one-level random hopping sequence defined as a list of channels in which the order of channels does not follow any particular pattern. For example, a one-level random hopping sequence for 4-active channels can be written as 15, 25, 26, 20. The sequence is set up prior to the network execution; and every node in the network must know this sequence and store the information in a mapping table as illustrated in Fig. 1a. During network execution, each node will receive the following information: ASN, channel offset, and transmission schedule. Based on this information and channel mapping, a pair of communicating node computes their transmission channel using Eqs. (1) and (2). Suppose there are 2 active nodes in the network $n_1$ and $n_2$ and the link connecting $n_1$ and $n_2$ is assigned with a $channel\_offset = 0$ as shown in Fig. 1b. Without loss of generality, assume that the two nodes are scheduled for transmission every time slot. Based on the channel mapping, each node knows that the channel sequence length is 4. To derive the transmission channel when $ASN = 1$, each node first computes the channel index using Eq. (1).

$$ch\_id = (1 + 0) \bmod 4 = 1 \quad (3)$$

After the two nodes know the channel index, the physical channel is obtained by looking up the channel mapping table. For example, the physical communication channel derived from Eq. (3) is 25. For each successive packet transmission, the nodes hop to a new

channel by using the same computation. The channel hopping sequence computed based on the current method in WirelessHART is shown in Fig. 2.

| ***ch*** | 25 | 26 | 20 | 15 | 25 | 26 | 20 | 15 | 25 | 26 | 20 | 15 | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ***ch_id*** | 1 | 2 | 3 | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 | 0 | ... |
| **ASN** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | ... |

**Figure 2: one-level of randomness hopping sequence**

The use of function-based channel hopping incurs less overhead since each node can easily calculate its communication channel from the ASN and channel offset information only. However, since the predefined sequence length is limited by the number of active channels, the modular function produces a strong repetitive channel usage, as shown in Fig. 2. The main objective of our work is to develop a new hopping strategy in which its predefined channel sequence is difficult to be figured out by a jammer.

## 3 RELATED WORK

Channel hopping techniques have been extensively studied to enhance communication reliability in various wireless networks [34]. Based on the channel hopping sequence across nodes, there are two types: *symmetric sequences* where every node uses the same sequence, and *asymmetric sequences* where channel sequences may vary among nodes. In terms of the requirement on time synchronization, existing channel hopping mechanisms can be further divided into two classes: *asynchronous* or *un-slotted channel hopping (USCH)*, and *time-synchronized channel hopping (TSCH)*.

Existing channel hopping strategies proposed for cognitive radio networks (CRNs) primarily focus on using asymmetric sequence for both USCH and TSCH [5, 6, 8, 22, 31]. However, these strategies are not suitable for WirelessHART networks due to the dynamic nature of communication channels in CRNs. Since asymmetric channel hopping techniques require additional procedures for channel selection, additional overhead is introduced to the network making them not suitable for WirelessHART networks that have stringent real-time and reliability requirements.

USCH is commonly implemented in contention-based MAC such as CSMA/CA. Several channel hopping schemes have been designed for IEEE 802.11 based networks [4, 13, 21, 24, 33]. However, these schemes are not applicable to be used to WirelessHART networks since the reliability requirement cannot be guaranteed. Besides, these approaches introduce additional overhead for exchanging the channel sequences, making them less suitable for WirelessHART networks as WirelessHART devices are highly energy-constrained. Most existing channel hopping techniques for low-power wireless sensor networks relying on a dedicated channel for channel hopping synchronization also incurs much overhead on latency. Thus, none of these approaches can meet the reliability and real-time requirement in WirelessHART. Furthermore, WirelessHART requires that every transmission in a time slot happens on a different channel for enhanced reliability. Applying other existing channel hopping strategies including the one used in Bluetooth/BLE may not ensure this.

In the current standard, in order to enable high reliability, WirelessHART employs the symmetric and TSCH approach in which both the channel sequence and hopping function are predefined before network execution. Channel hopping in TSCH uses a modulo-based function. The recent work in [14] improves channel hopping strategy by adding mechanism for channel quality evaluation to increase reliability. Gunatilaka et al. have proposed to enhance WirelessHART network reliability through channel reuse and reactive link scheduling, employing the basic TSCH protocol [17, 18]. Notably, their work does not focus on developing new TSCH schemes. The study in [10] has presented the vulnerability of using modular function in TSCH. Cheng et al. in [11, 12] showed the crucial impact of detecting future channel usage, particularly for WirelessHART networks.

Several anti-jamming solutions have been proposed recently to disguise channel hopping pattern. R-TSCH [36] is an anti-jamming strategy for TSCH based network that tries to tackle the problem by generating random channel sequence using cryptographic key. Kim et al. proposed another way to reduce a jammer's capability of figuring out the channel hopping pattern by detecting the jammer's channel sequence and changing the communication channel sequence according to the jammer's sequence [19]. However, these solutions require a more advanced device capability and additional synchronization overhead to exchange the randomized sequence. Several anti-jamming techniques addressing WirelessHART networks, in particular, are proposed in [28–30] approaching the problem by randomizing the transmission slot for end-to-end data flow. Nonetheless, these strategies do not specifically remove the repetition pattern.

Our work is the first to present a resilient channel hopping scheme for WirelessHART networks without needing additional sequence synchronization overhead, complex device's capability, and scheduling modification. The scope of this work is limited to devising new hopping function for WirelessHART networks. Any random channel sequence generated through state-of-the-art pseudo random number generator (PRNG) can be applied to our hopping scheme as WirelessHART hopping sequence as long as both the sender-receiver pair use the same PNRG.

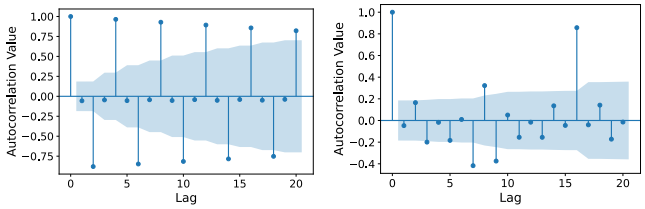
## 4 PROPOSED JAMMING-RESILIENT CHANNEL HOPPING

In this section, we first discuss the attack vulnerability of the current channel hopping technique used in WirelessHART, the attack model, and then present our jamming-resilient channel hopping scheme.

### 4.1 Attack vulnerability

In order to ensure that all devices know their communication time and channels, a tuple of time slot and channel offset $\{slot\_index,, channel\_offset\}$ is assigned to every device in the network. Under the current hopping technique, every receiver uses a unique channel which is not used by any other receivers across the network at any time slot. This approach can guarantee high reliability without inducing overhead in the network. However, the recent work in [10] observed that the existing technique is vulnerable to

jammers who can easily crack the channel hopping pattern without knowing the predefined sequence. Once the pattern is known, a severe selective jamming attack can be launched. This weakness mainly stems from the strong repetitive pattern of channel usage due to the short channel sequence length.




(a) 4 active channels (b) 16 active channels

**Figure 3: Comparison of autocorrelation graphs of hopping sequences generated from the current hopping technique.**

To show the strong repetitiveness of channel usage in the current approach, we perform an autocorrelation analysis on its channel hopping sequence. The autocorrelation analysis helps in identifying patterns within time series data. We generate a sequence using Eq. (1), where we set $channel\_offset = 0, ASN = \{0, 1, 2, \cdots, 20\}$, and $s\_length = \{4, 16\}$. Fig. 3a and 3b depict the autocorrelation results of the current hopping function when 4 and 16 active channels are used in the network, respectively. In this example, we select $s\_length = \{4, 16\}$ to show that less number of active channels produces a stronger repetition. In the autocorrelation analysis, *lag* is used to define the time gap or time period apart between values. Hence, the lag showing positive spike in autocorrelation value is the period when the sequence repeats. From Fig. 3a, when there are four active channels used in the network, the positive spike of autocorrelation value appears when the lag is a multiple of 4 such as 4, 8, 12, $\cdots$, which means that the hopping sequence repeats after 4 time (units) apart. A similar result can be observed from Fig. 3b where the positive spike of autocorrelation value appears when lag is a factor of 16. Both figures show strong repetitive pattern of channel usage.

## 4.2 Attack model

In this work, instead of jamming the network in all channels, we consider a jamming attack model for TSCH-based networks such as WirelessHART investigated in [10]. The straightforward jamming model requires more resource at the jammer while it is reported in [12] that the smart selective jamming makes the attack energy efficient and hardly detectable. Hence, we used this type of attack to evaluate the performance of our method. The attackers' main objective, in their scenario, is to crack the channel hopping sequence in order to launch selective jamming attack [12] reducing the network reliability. There are 4 steps that the jammers need to complete before the channel hopping is figured out as follows: (1) Grouping eavesdropped packets; (2) Identifying least common multiple (LCM) of the number of time slots in combined superframe and sequence length; (3) Identifying time slots of every transmission; and (4) Creating channel offset table. Among these steps, identifying the LCM is the core function to predict the future hopping pattern since the hopping sequence follows a periodic fashion. To successfully perform the jamming procedures, the jammer should be equipped with moderate computational abilities, capable of monitoring and generating signals across all channels in 2.4 GHz to intercept WirelessHART packets. It has energy-budget and needs to carefully spend energy for jamming.

Here we describe an example of an attack scenario where a jamming device is placed near an industrial plant with a WirelessHART network in operation. Suppose the jammer listens to a link connecting node $n_1$ and $n_2$ (as depicted in Fig. 1) with channel hopping pattern as described in Fig. 2. In this situation, the jammer has no prior knowledge of the sequence used by the WirelessHART nodes. Initially, the jammer will monitor all communication channels according to its own channel-hopping sequence and capture transmitted packets on the channels it is currently tuned to. For each intercepted packet, the sender-receiver information can be extracted from the packet's header, which is not encrypted. The jammer will also record the time slots and communication channels for each successfully intercepted packet. Subsequently, the jammer will create a jamming table that includes essential details such as sender and receiver identities, channel information, and approximate time slots of packet transmissions. By analyzing this data and calculating the least common multiple (LCM), the jammer can predict the timing and channel of future transmissions. Once these predictions are validated, the jammer will proceed with a selective jamming attack.

As identified in [10], WirelessHART's hopping sequence can be cracked more easily and quickly than that of Orchestra [15]. This happens because the lengths of superframes stated in WirelessHART standard follow a harmonic chain (e.g., 1, 2, 4, 8, $\cdots$), i.e. each superframe length is divisible by all smaller superframe lengths. The LCM of these values indicates the time period after which the entire schedule of transmissions repeats and is much smaller compared to the case when superframe lengths are relative prime numbers. Therefore, our work focuses on improving the resilience of channel hopping in WirelessHART so that the hopping pattern cannot be cracked within a feasible amount of time. Our work is different from [10] as their work shows the vulnerability by launching an attack while we propose to solve that vulnerability.

## 4.3 Jamming-Resilient Channel Hopping

Designing a jamming-resilient hopping strategy for WirelessHART network is challenging due to the following reasons:

- *A small number of available channels:* WirelessHART can support a maximum of 16 channels as specified in IEEE 802.15.4, and some channels can even be blacklisted due to low channel quality. Hence, the channel sequence used in the WirelessHART's default TSCH mechanism is short and thereby can not give much protection from channel cracking.
- *Fast channel switching with low overhead:* the WirelessHART standard mandates a maximum channel switching time of 0.192 ms. This stringent requirement makes it impossible to use different random sequences across the nodes or dynamic random sequences, since they cannot guarantee all pairs of communicating devices scheduled in the same slot

use different channels. Although most of the modern IoT devices are equipped with a standard pseudo-random number generator (PRNG), each sender-receiver pair must use the same random seed to generate the pseudo-random channel sequence so that they can hop to the same channel in each time slot. Using a synchronized random seed (e.g. ASN) for all concurrent transmission pairs can still generate channel hopping sequence with strong repeated patterns, whereas using different seeds for different transmission pairs requires each sender and its receiver to synchronize its own seed, which can result in significant overhead.

To design a jamming-resilient channel hopping scheme, we still adopt the principle in the current approach (i.e. modular function + predefined channel sequence) to ensure fast channel switching with low overhead, but explore multi-level randomness to reduce repetitive patterns in channel hopping sequence. In WirelessHART, A time slot can be either *dedicated* (where at most one transmission is scheduled to a receiver) or *shared* (where multiple nodes may contend to send to a common receiver). It is important to note that our proposed solution is applicable to shared links since it is a function-based channel hopping scheme that relies only on the ASN, channel offset, and the pre-assigned random sequences, not the type of links. Unlike the current technique that only employs one channel sequence of size limited to the number of available channels, our approach uses the following sequences:

- Channel sequence 0: $ch\_sequence_0$
- Channel sequence 1: $ch\_sequence_1$
- An alternation sequence: $alt\_sequence$
- A pointer sequence: $pointer\_sequence$

where $ch\_sequence_0$ and $ch\_sequence_1$ are two random sequences of active channels derived from the channel mapping, $alt\_sequence$ is a random list of 0 and 1 of a predefined length used to determine whether $ch\_sequence_0$ or $ch\_sequence_1$ is currently active for computing transmission channel, and $pointer\_sequence$ is a list of channel pointer indicating the starting index of the active channel sequence. When a pair of sender and receiver is about to communicate, they need to firstly determine which channel sequence ($ch\_sequence_0$ and $ch\_sequence_1$) will be used based on $alt\_sequence$. After the channel sequence is selected, the sender and receiver will calculate the pointer sequence and use the value in the pointer sequence to shift the selected channel sequence. Finally, the communication channel is computed from the shifted channel sequence. In-depth explanations of these procedures are provided in the following of this section.

The presence of multiple sequences will increase the repetition period of channel significantly and create a confusing channel usage pattern which forces the jammer to put a humongous amount of effort and time before making prediction. When our channel hopping is applied, a jammer would make wrong predictions of future channel usage, since the same channel may be repeated multiple times without representing a repetitive pattern of the sequence. In this section, we provide a detailed explanation of the proposed channel hopping technique, which includes two additional channel sequences, an alternation sequence, and a pointer sequence. This configuration serves as the lower bound for jamming resilience performance against selective jamming attacks. The upper bound on the number of sequences that can be incorporated into this technique depends on the number of available channels, as further detailed in Section 4.5. Our approach consists of the following three main procedures.

| | | | | |
|---|---|---|---|---|
| ***ch_sequence$_0$*** | 3 | 1 | 0 | 2 |
| ***ch_sequence$_1$*** | 1 | 0 | 3 | 2 |
| ***pointer_sequence*** | 2 | 1 | 3 | 0 |
| ***alt_sequence*** | 0 | 1 | 1 | 0 |

**(a) Sample random sequences**

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ***ch*** | 26 | 20 | 25 | 25 | 15 | 26 | 20 | 26 | 20 | 25 | 15 | 20 | 25 | 15 | 26 | 15 |
| ***ch_id*** | 2 | 3 | 1 | 1 | 0 | 2 | 3 | 2 | 3 | 1 | 0 | 3 | 1 | 0 | 2 | 0 |
| **ASN** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |

**(b) Generated sequence after performing shifting operation**

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ***ch_id*** | 2 | 3 | 1 | 1 | 0 | 2 | 3 | 2 | 3 | 1 | 0 | 3 | 1 | 0 | 2 | 3 |
| **ASN** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| ***ch_id*** | 2 | 1 | 0 | 0 | 3 | 2 | 1 | 2 | 1 | 0 | 3 | 1 | 0 | 3 | 2 | 0 |
| **ASN** | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |

**(c) Complete hopping sequence after shifting and alternation are applied**

**Figure 4: Major procedures to reduce repetitive nature in hopping sequence from Fig. 1**

*4.3.1 Generating Random Sequences.* We use the same topology and channel mapping provided in Fig. 1 to illustrate the procedure for generating the random sequences. Suppose we have a network that supports 4 different active channels ($s\_length = 4$), namely channel 15, 25, 26, 20 stored in a mapping $A$ (see Fig. 1a). Note that the channel sequence in array $A$ is set randomly. Assume that in our network, there are 2 wireless nodes $n_1$ and $n_2$ that are connected through a link with channel offset 0 as presented in Fig. 1b. Based on the available channel information, the following gives an example of the random sequences that can be set to both node $n_1$ and $n_2$ before the network execution:

- $ch\_sequence_0$ = [3, 1, 0, 2]
- $ch\_sequence_1$ = [1, 0, 3, 2]
- $alt\_sequence$ = [0, 1, 1, 0]
- $pointer\_sequence$ = [2, 1, 3, 0]

Each node keeps the sequences in a table depicted in Fig. 4a, and all sequences start with index 0.

*4.3.2 Shifting Channel Sequences.* To increase randomness, we apply random shifting between $ch\_sequence_0$ and $ch\_sequence_1$ using the $pointer\_sequence$. The random shift, denoted by $pointer$, is derived based on $ASN$ and the $pointer\_sequence$ as follows:

$$pointer\_id = \lfloor (ASN / s\_length) \rfloor \bmod s\_length \tag{4}$$

$$pointer = pointer\_sequence[pointer\_id] \quad (5)$$

For example, using the *pointer_sequence* in Fig. 4a and $ASN = 1$, we can compute the pointer index using Eq. (4) and derive the pointer value using Eq. (5) as follows:

$$pointer\_id = \lfloor (1/4) \rfloor \bmod 4 = 0 \quad (6)$$

$$pointer = pointer\_sequence[0] = 2 \quad (7)$$

As a result, when ASN = 1, the channel sequence is shifted by 2. As the main objective of channel hopping is to mitigate jamming attack and channel interference, maintaining the fairness of channel distribution in the network becomes critical. Therefore, the *pointer's* value must not be updated before all active channels are used for transmission. Since $s\_length$ denotes the total active channels, taking the floor of $ASN/s\_length$ in Eq. (4) guarantees that *pointer's* value remains the same before every channel is utilized.

Once the value of *pointer* is computed, the following functions are used to calculate the physical channel:

$$ch\_id = (channel\_offset + ASN + pointer) \bmod s\_length \quad (8)$$

$$ch = A[ch\_sequence_i[ch\_id]] \quad (9)$$

where $i \in \{0, 1\}$ denotes the id of the active channel sequence. Basically, Eq. (8) is similar to Eq. (1) that gives the current channel hopping function. The main difference between the two equations lies on the shifting technique which is shown as an addition function of the pointer's value.

To illustrate, Fig. 4b depicts the hopping sequence after the shifting technique is applied. Suppose $ch\_sequence_0$ is used currently. Assume that the network topology in Fig. 1b is deployed where $channel\_offset = 0$ is assigned to the link connecting 2 wireless nodes, $n_1$ and $n_2$. Previously, the pointer's value has been computed in Eq. (6) and Eq. (7) where $pointer = 2$ when $ASN = 1$. Now, to derive the transmission channel, Eq. (8) and Eq. (9) are used as follows:

$$ch\_id = (0 + 1 + 2) \bmod 4 = 3 \quad (10)$$

$$ch = A[ch\_sequence_0[3]] = A[2] = 26 \quad (11)$$

For the subsequent transmissions, the physical channels are computed in the same manner.

*4.3.3 Random Sequence Alternation.* To further puzzle a jammer in predicting our channel sequence, we introduce the concept of sequence alternation to randomly alternate between $ch\_sequence_0$ and $ch\_sequence_1$ based on the alternation sequence.

The alternation is done after all starting pointers have been applied to shift one of the channel sequences. Since the length of *pointer_sequence* and *channel_sequence* are the same which equals the $s\_length$; when transmission is scheduled in every slot, the alternation occurs after every $(s\_length)^2$ slots due to the use of two channel sequences. If the number of channel sequences ($i$) is greater than two, the alternation occurs after $(s\_length)^i$ slots. Unlike the other three random sequences, the size of alternation sequence is not limited to the number of active channels. Therefore, the cycle of hopping sequence could be enlarged dramatically by providing a large size of alternation sequence. The longer the length of the alternation sequence, the longer will be the time after which channel usage pattern repeats. The following function is used to compute the alternation index ($alt\_id$):

$$alt\_id = \lfloor (ASN/(s\_length)^2) \rfloor \bmod alt\_sequence\_size \quad (12)$$

where $alt\_sequence\_size$ represents the size of the alternation sequence. After $alt\_id$ is computed, the sequence index can be obtain by:

$$i = alt\_sequence[alt\_id] \quad (13)$$

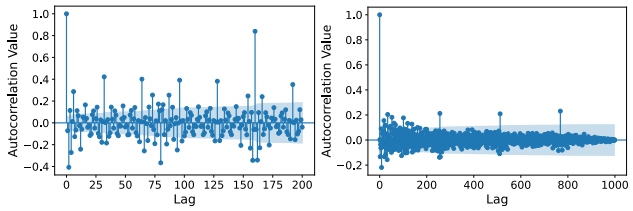
Fig. 4c illustrates the final hopping sequence after alternation technique is used. For example, applying shifting technique to the $ch\_sequence_0$ with all the available values from *pointer_sequence* in the second step produces a hopping sequence whose length is $(s\_length)^2$ (see Fig. 4b). Now, suppose another transmission is scheduled when ASN = 17. To determine which channel sequence to use, Eq. (12) and Eq. (13) are applied as follows:

$$alt\_id = \lfloor (17/(4)^2) \rfloor \bmod 4 = 1 \quad (14)$$

$$i = alt\_sequence[1] = 1 \quad (15)$$

Since the result of this computation is 1, $ch\_sequence_1$ is selected to derive the physical channel. As a result of adding shifting and alternation procedures for hopping sequence generation, a longer sequence can be produced. Obviously, the current function used in WirelessHART for hopping sequence has a period equal to $s\_length$ slots while our approach generates a hopping sequence with a period of $(alt\_sequence\_size)(s\_length)^2$ slots.

## 4.4 Analysis on Channel Usage Pattern




(a) 4 active channels (b) 16 active channels

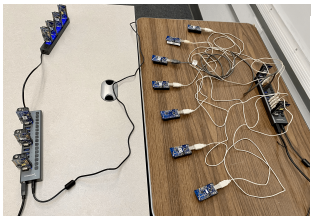
**Figure 5: Comparison of autocorrelation graphs of hopping sequences generated from our proposed method.**

We perform an autocorrelation analysis on the channel hopping sequence generated by our proposed method with the following setting: *channel offset = 0*, $ASN = \{0, 1, 2, ..., 200\}$ when $s\_length = 4$ and $ASN = \{0, 1, 2, ..., 1000\}$ when $s\_length = 16$. As shown in Fig. 5a and 5b, our proposed technique successfully produces channel sequence with 40x and more than 60x larger period for 4 and 16 active channels, respectively. In the case where 4 channels are used for transmission, the length of generated hopping sequence is approximately 160. This means that the whole sequence will repeat after 160 transmissions. Furthermore, when 16 active channels are used, the hopping channel will not repeat even after 1000 transmissions. From this result, we can confirm that our proposed technique can significantly reduce the repetitive nature of existing channel hopping technique and improve its resilience towards jamming attack.

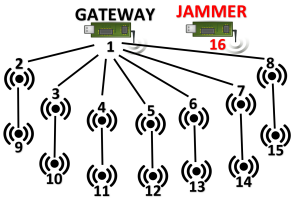
Under realistic traffic scenario where transmission is not scheduled in every slot, jammers often encounter difficulty in successfully intercepting transmitted data, leading to sparse datasets. From this fragmented data, accurately deriving the correct LCM to predict future channel usage becomes a formidable challenge. In [10], a *trial and error* learning is applied to tackle such condition. The advantage of employing our hopping scheme is that a channel from the available sequence appears multiple times without presenting the correct period of the hopping pattern. As we prolong the repetition period of the hopping pattern, a channel may be used for communication multiple times before the hopping cycle is repeated. As a result, the attacker will perform numerous attempts of trial-error inaccurately. Launching smart selective jamming with incorrect channel usage prediction could increase the risk of exposing the jamming attempt. Thus, despite the presence of periodicity in our channel sequence, the inherent complexity arising from sparse and incomplete data causes the jamming task to be significantly difficult and time-consuming for attackers. Besides, an attacker may not target explicitly our approach without knowing our three levels of randomness.

## 4.5 Enhancing the Robustness in Channel Hopping

Our approach can be made even more robust by adding more randomness through additional levels. Specifically, we can use more than two random channel sequences that is $ch\_sequence_i$ where $i = \{0, 1, 2, ..., s\_length\}$, and add a pointer sequence for each of these sequences ($pointer\_sequence_i$). Finally, we can use an alternate sequence of length greater or equals to the number of random channel sequences (i.e., $alt\_sequence\_size \geq i$) to choose a sequence from the $i$ sequences. By increasing the levels of randomness in our strategy, we theoretically introduce only a minor constant factor of computational complexity. Since only one communication channel is assigned to each sender-receiver pair, the addition of multiple sequences does not impact the network's interference immunity or timing performance. These factors are determined solely by the number of active channels used concurrently in a given time slot. Furthermore, the levels of randomness can be a tunable parameter which can be chosen based on our desired robustness or the severity of jamming.



(a) Real deployment

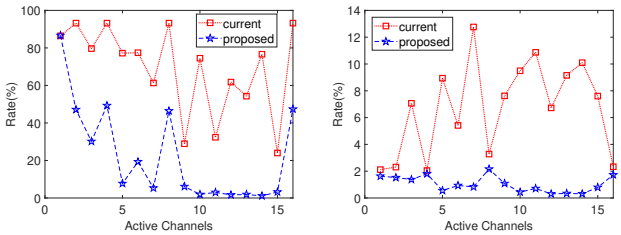



(b) Topology showing the routes to the Gateway

**Figure 6: Experiment deployment and topology**

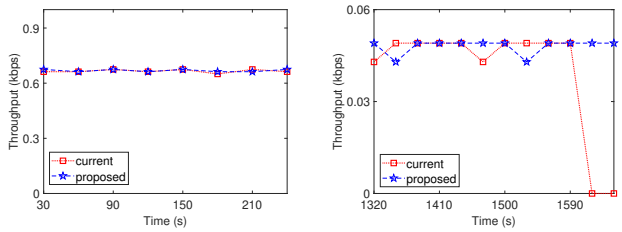
# 5 PERFORMANCE EVALUATION

In this section, we evaluate the robustness of our proposed jamming-resilient channel hopping strategy by comparing its performance with the current approach adopted in WirelessHART. Recall that WirelessHART mandates that no two nodes can hop to the same channel in a dedicated time slot. Since the methods discussed in [19, 28–30, 36] do not guarantee this requirement, we did not compare our solution with these state-of-the-art approaches. We first validate our scheme through testbed experiments and then evaluate its performance in large-scale networks through simulations.




(a) TPR (b) PGR

**Figure 7: Results in real-deployment network**




(a) Network throughput under jamming - free (b) Throughput per link under jamming attack

**Figure 8: Network performance under varying network environment in real-deployment network**

## 5.1 Experimental setup

We have implemented the proposed hopping mechanism in Contiki-NG [25] operating system, since Contiki-NG supports TSCH feature complying with the WirelessHART's physical layer. In our experiments, we used TelosB [3] as the WirelessHART's nodes, gateway, and jamming device because TelosB has the same physical layer as WirelessHART. We have used 14 TelosB devices as WirelessHART sensor nodes, one device as the gateway, and one device as a jammer. Fig. 6a shows the network deployed in an indoor environment. We adjusted the transmission power to form a two-hop network. Fig. 6b shows the routes from the sensor nodes to the gateway. In our experiments, each sensor generates a packet after every 2560 ms, the number of active channels was varied from 1 to 16, and $s\_length$ is set to 15. For the jamming strategy, we imitate the method presented in [10] in which the jammer aims to predict future communication channel by observing channel usage through snooping transmitted packet. The jammer will only

eavesdrop transferred packets in the network for a fixed time duration.

*Performance Metrics.* To evaluate the robustness of the proposed approach, the following evaluation metrics are used:

- **TPR**: true prediction to total prediction ratio, indicating the fraction of correct predictions out of the total predictions made by the jammer for future channel usage.
- **PGR**: correct prediction to ground truth ratio, defined as the fraction of jammer's total correct channel usage predictions over the total ground truth transmission channel usage (actual channel usage) in the network.
- **Throughput**: number of bits received per second.

## 5.2 Experimental Results

*5.2.1 Channel prediction.* Fig. 7 shows TPR and PGR under varying number of channels achieved by our channel hopping approach and the current one in WirelessHART. Fig. 7a shows that our approach could lower TPR of a jammer by more than 50% compared to the current strategy. A significant jamming-proof performance is achieved when the number of channels is relatively prime to the superframe's length and greater than 8. In addition, Fig. 7a shows that TPR decreases as the number of active channels increases and remains significantly lower compared to the current one. Furthermore, Fig. 7b shows that our strategy also lowers PGR on average. In summary, utilizing relatively prime numbers and more active channel enables more channel switching in the network, leading to significant increase of the jamming resilience.

*5.2.2 Throughput under Jamming.* We measure the network throughput in the gateway side every 30 seconds for both our channel hopping approach and the current approach. Our objective is to show the performance difference under jamming. First we compare both approaches under a jamming-free scenario as shown in Fig. 8a. As expected, both approaches yield almost the same throughput. We consider a jamming scenario where a jammer can launch selective jamming attack after determining a link's channel. After observing transmitted packets in one link for less than 30 minutes, the jammer could predict future channel usage in that particular link and jam that link. Fig. 8b shows that the throughput sharply drops to 0 kbps after channel detection and the start of jamming. Under our approach, the throughput remains steady as the jammer has not been able to determine the channel usage.

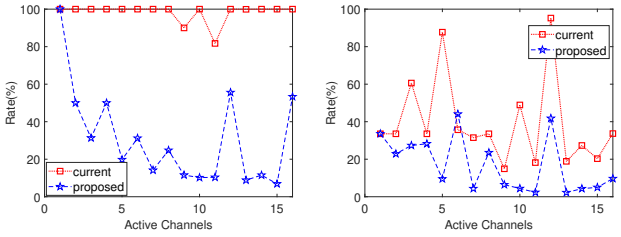



**Figure 9: Results in single-hop sparse network**

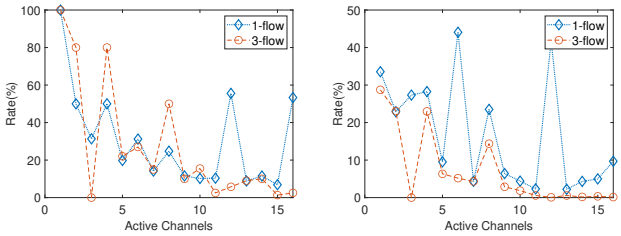



**Figure 10: Multiple data flows in single-hop sparse network**

## 5.3 Simulation results

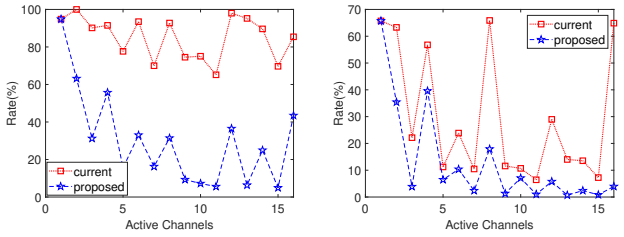
To evaluate the robustness of our jamming-resilient channel hopping strategy in large-scale networks, we perform simulations in Cooja [26] - the default simulator in Contiki. We consider two types of networks, namely sparse and dense networks. The sparse network consists of a few links and nodes. We consider less than 10 nodes. The sparse network schema allows jammer to eavesdrop more packets in the network through limited number of links. Through this simulation, we wish to investigate the resilience of our proposed technique when jammer has such capabilities. To further observe our jamming-resilient hopping technique, we run simulations in dense network scenario which involves many links and hundreds of nodes in the network. Unlike the sparse network, jammer can only get a smaller fraction of transmitted packets from each link. For each type of networks, we further consider two network typologies, namely *single-hop* and *multi-hop* networks. A single-hop network represents an easier scenario for a jammer to crack the channel usage as there is only one active link at a time (as each device including the gateway has a single half-duplex radio). A multi-hop network represents a relatively more difficult scenario for a jammer to crack the channel usage as there may be many concurrent transmissions in a time slot. We set one gateway and one jammer in every simulation while we vary the number of sensor nodes. *s_length* is varied from 1 to 16 unless stated otherwise. Jamming simulation for both the baseline (current approach) and proposed technique are done for the same duration of time since we only consider the jamming performance under the same amount of jamming energy to observe and crack channel usage. We evaluate channel predictability using the same evaluation metrics described in Section 5.

*5.3.1 Results under Sparse Networks.* We consider a sparse network that consists of 3 nodes; 1 sensor node and 1 gateway, and 1 jammer. In the first scenario, we schedule each sensor node to generate one data flow every 320 ms in a *superframe* whose length is 32 time slots. Subsequently, we consider the scenario with multiple concurrent flows, where each sensor generates one packet either every 320 ms, 640 ms, or 1280 ms.

Fig. 9 shows the results in terms of the success rate in channel cracking. It can be seen that the jamming success rate is significantly lower by more than 60% when our strategy is applied compared to the current approach (see Fig. 9a). Our approach causes

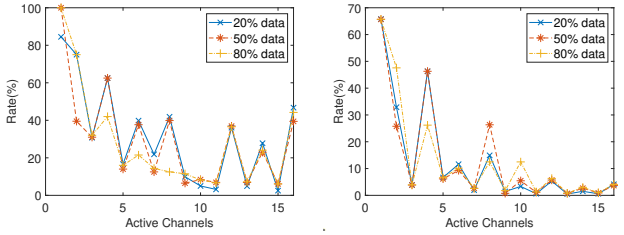
jammer to miscalculate the repetition period. As *s_length* is increased, TPR keeps decreasing. Furthermore, we observe that applying our strategy decreases the PGR on average as presented in Fig. 9b. Applying our approach forces the jammer to predict considerably large repetition period. As the predicted period becomes larger, PGR significantly decreases.

Fig. 10a shows the results with flows generated by three nodes every 320, 640, and 1280 ms, respectively. It can be seen that a significant resilience difference happens when *s_length* is greater than 8. As shown, the jammer can hardly predict the future channel usage when each node generates 3 data flows, where PGR $\approx$ 0%, compared to that of single-data flow. This indicates that adding more data flow to the network increases jamming difficulty, particularly when *s_length* $> 8$.




**(a) TPR** **(b) PGR**

**Figure 11: Results in multi-hop sparse network**

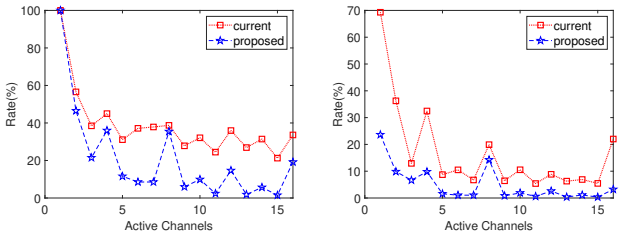



**(a) TPR** **(b) PGR**

**Figure 12: Multiple training data in multi-hop sparse network**

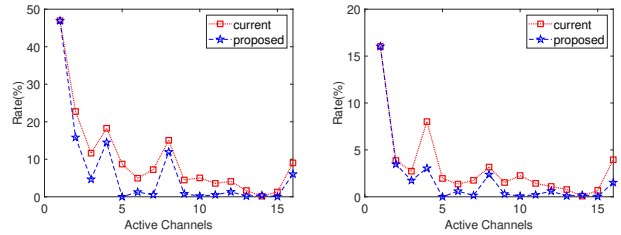
We further evaluate the resilience performance by adding more nodes and links to form a multi-hop network. Here, we consider a sparse network that consists of 6 nodes; 4 sensor nodes and 1 gateway, and 1 jammer. Fig. 11a shows that our approach dramatically improves the jamming difficulty by more than 60%. Similar to the single-hop network, the most noticeable difference between the current hopping strategy and the proposed method can be observed when *s_length* $> 8$. From the PGR in Fig. 11b, when the number of active channels are 8, 12, 16, the jammer can only make less than 20% correct predictions. To summarize, as *s_length* and number of nodes increase in the network, the pattern of channel hopping sequence becomes confusing to the jammer, reducing its capabilities of making correct prediction.

Subsequently, we analyze the jamming difficulty based on the size of training data. The training data is represented in the form




**(a) TPR** **(b) PGR**

**Figure 13: Results in single-hop dense network**




**(a) TPR** **(b) PGR**

**Figure 14: Results in multi-hop dense network**

of percentage. For example, 20% data means that 20% portion data of the total observation is used for the jammer to snoop the channel usage. This means that as we increase the percentage, we use more observation data for training. The channel prediction under different size of observation data is presented in Fig. 12. In general, it is shown that increasing the amount of training data does not enhance jammer capability of predicting channel usage. Instead, we found interesting findings from this simulation scenario. When *s_length* $\in \{4, 6, 8\}$, Fig. 12a shows approximately 20% drops of jammer's correct prediction. This means that, in some cases, using more observation data for making prediction would confuse the jammer, leading to making incorrect prediction. This happens when the jammer is not able to derive the correct channel usage repetition as a result of adding multilevel randomness to the hopping method. Fig. 12b demonstrates that a significant drop in PGR happens while the number of active channels is prime or relative prime to the superframe's length, such as 3, 5, 7, 9, 11, 13, and 15. The use of prime or relatively prime numbers enables more channel switching in the network. From these results, we can conclude that the utilization of prime or relative-prime to the size of superframe numbers as the *s_length* will increase the jamming difficulty.

*5.3.2 Results under Dense Networks.* In this setting, we deploy one WirelessHART gateway node and one jammer node in the network while we increase the number of sensor nodes. Each sensor generates a data packet every 520 ms.

Fig. 13 shows the results in single-hop dense network setting with 100 sensor nodes. It can be seen from Fig. 13a that the proposed technique is able to considerably reduce TPR (by approximately 50%) compared to that of current hopping method. Fig. 13b shows PGR under both approaches. As conclusion, our proposed technique has shown satisfactory jamming-proof performance in

dense network setting compared to the existing technique when *s_length* is set to prime or relative-prime numbers.

We further evaluate our scheme in a multi-hop network consisting of 200 nodes. As the number of nodes is significantly higher while the number of jammer is fixed (at one), we can expect a reduced performance of jammer to eavesdrop packets. The prediction results are shown in Fig. 14. It can be seen that our approach outperforms the current strategy of channel hopping in terms of both TPR and PGR. As we increase *s_length*, the general pattern shows that jamming the channel sequence becomes extremely difficult, especially when $s\_length > 8$. To conclude, our approach is able to give protection to jamming attack as the repetitive nature is reduced by adding randomness to the hopping technique.

## 6 CONCLUSION

WirelessHART is a prominent standard for industrial Internet of Things applications providing feasibility for reliable and real-time communication. The current channel hopping strategy for ensuring reliable communication in WirelessHART networks exhibits strong repetitive channel usage patterns, making it vulnerable to channel cracking and jamming attacks. Designing a jamming-resilient channel hopping strategy for WirelessHART networks is challenging as it must guarantee fast channel switching ($\leq 0.192$ ms), channel synchronization between sender-receiver pairs, and interference-free concurrent transmissions. In this paper, we have proposed a jamming-resilient channel hopping mechanism that meets these requirements for WirelessHART networks. Our approach significantly reduces the strong repetitive pattern in channel sequence by adding randomness at multiple levels. We have implemented the proposed approach and evaluated through testbed experiments using a 16-node network and also through large scale simulations in Cooja. The results show that our proposed approach can significantly lower a jammer's capability in making correct channel prediction without increasing the channel hopping overhead from the current approach.

## ACKNOWLEDGEMENT

The work was supported by the US National Science Foundation through grants CNS-2301757, CAREER- 2306486, CNS-2306745, and by the US Office of Naval Research through grant N00014-23-1-2151.

## REFERENCES

[1] 2007. WirelessHART. https://fieldcommgroup.org/technologies/hart.
[2] 2009. ISA100: Wireless Systems for Automation. http://www.isa.org/MSTemplate.cfm?MicrositeID=1134&CommitteeID=6891.
[3] 2016. CC2420 RF-Transceiver. http://www.ti.com/lit/ds/symlink/cc2420.pdf.
[4] Jia Bai, Emeka Eyisi, Yuan Xue, and Xenofon Koutsoukos. 2010. Dynamic Tuning Retransmission Limit of IEEE 802.11 MAC Protocol for Networked Control Systems. In *CPSCom'10*.
[5] Kaigui Bian and Jung-Min Park. 2011. Asynchronous channel hopping for establishing rendezvous in cognitive radio networks. In *INFOCOM'11*. 236–240.
[6] Kaigui Bian, Jung-Min Park, and Ruiliang Chen. 2011. Control Channel Establishment in Cognitive Radio Networks using Channel Hopping. *IEEE Journal on Selected Areas in Communications* 29, 4 (2011), 689–703.
[7] Joris Borms, Kris Steenhaut, and Bart Lemmens. 2010. Low-overhead dynamic multi-channel MAC for wireless sensor networks. In *EWSN'10*. Springer, 81–96.
[8] Guey-Yun Chang, Wen-Hung Teng, Hao-Yu Chen, and Jang-Ping Sheu. 2014. Novel Channel-Hopping Schemes for Cognitive Radio Networks. *IEEE Transactions on Mobile Computing* 13, 2 (2014), 407–421.
[9] D Chen, M Nixon, and A Mok. 2010. *WirelessHART™Real-Time Mesh Network for Industrial Automation*. Springer.
[10] Xia Cheng, Junyang Shi, and Mo Sha. 2019. Cracking the channel hopping sequences in IEEE 802.15.4e-based industrial TSCH networks. In *IoTDI'19*.
[11] Xia Cheng, Junyang Shi, and Mo Sha. 2019. Cracking the Graph Routes in WirelessHART Networks. In *AsiaCCS'19*.
[12] Xia Cheng, Junyang Shi, Mo Sha, and Linke Guo. 2021. Launching Smart Selective Jamming Attacks in WirelessHART Networks. In *INFOCOM'21*.
[13] Hon Sun Chiu, Kwan L. Yeung, and King-Shan Lui. 2009. J-CAR: an efficient joint channel assignment and routing protocol for IEEE 802.11-based multi-channel multi-interface mobile ad hoc networks. *Trans. Wireless. Comm.* 8, 4 (2009).
[14] Peng Du and George Roussos. 2012. Adaptive time slotted channel hopping for wireless sensor networks. In *CEEC'12*. 29–34.
[15] Simon Duquennoy, Beshr Al Nahas, Olaf Landsiedel, and Thomas Watteyne. 2015. Orchestra: Robust mesh networks through autonomously scheduled tsch. In *Sensys'15*. 337–350.
[16] Prabal Dutta, Stephen Dawson-Haggerty, Yin Chen, Chieh-Jan Mike Liang, and Andreas Terzis. 2010. Design and Evaluation of a Versatile and Efficient Receiver-Initiated Link Layer for Low-Power Wireless. In *SenSys '10*. 1–14.
[17] Dolvara Gunatilaka and Chenyang Lu. 2018. Conservative Channel Reuse in Real-Time Industrial Wireless Sensor-Actuator Networks. In *ICDCS'18*. 344–353.
[18] Dolvara Gunatilaka and Chenyang Lu. 2020. REACT: an Agile Control Plane for Industrial Wireless Sensor-Actuator Networks. In *IoTDI'20*. 53–65.
[19] Yongchul Kim and Jungho Kang. 2017. Efficient Anti-Jamming Technique Based on Detecting a Hopping Sequence of a Smart Jammer. *IOSR Journal of Electrical and Electronics Engineering* 12 (2017), 118–123.
[20] Youngmin Kim, Hyojeong Shin, and Hojung Cha. 2008. Y-MAC: An Energy-Efficient Multi-channel MAC Protocol for Dense Wireless Sensor Networks. In *IPSN'08*. 53–63.
[21] M. Lacage, M. H. Manshaei T., and Turletti. 2004. IEEE 802.11 rate adaptation: a practical approach. In *MSWiM '04*. 126–134.
[22] Zhiyong Lin, Hai Liu, Xiaowen Chu, and Yiu-Wing Leung. 2011. Jump-stay based channel-hopping algorithm with guaranteed rendezvous for cognitive radio networks. In *INFOCOM'11*. 2444–2452.
[23] Tie Luo, Mehul Motani, and Vikram Srinivasan. 2009. Cooperative Asynchronous Multichannel MAC: Design, Analysis, and Implementation. *IEEE Transactions on Mobile Computing* 8, 3 (2009), 338–352.
[24] V. Navda, A. Bohra, S. Ganguly, and D. Rubenstein. 2007. Using Channel Hopping to Increase 802.11 Resilience to Jamming Attacks. In *INFOCOM'07*. 2526–2530.
[25] George Oikonomou, Simon Duquennoy, Atis Elsts, Joakim Eriksson, Yasuyuki Tanaka, and Nicolas Tsiftes. 2022. The Contiki-NG open source operating system for next generation IoT devices. *SoftwareX* 18 (2022), 101089.
[26] Fredrik Osterlind, Adam Dunkels, Joakim Eriksson, Niclas Finne, and Thiemo Voigt. 2006. Cross-Level Sensor Network Simulation with COOJA. In *LCN'06*. 641–648.
[27] Abusayeed Saifullah, Dolvara Gunatilaka, Paras Tiwari, Mo Sha, Chenyang Lu, Bo Li, Chengjie Wu, and Yixin Chen. 2015. Schedulability analysis under graph routing in WirelessHART networks. In *Real-Time Systems Symposium, 2015 IEEE*. IEEE, 165–174.
[28] Ankita Samaddar and Arvind Easwaran. 2023. Online Distributed Schedule Randomization to Mitigate Timing Attacks in Industrial Control Systems. *ACM Transactions on Embedded Computing Systems* 22, 6 (2023), 1–39.
[29] Ankita Samaddar, Arvind Easwaran, and Rui Tan. 2020. A schedule randomization policy to mitigate timing attacks in WirelessHART networks. *Real-Time Systems* 56, 4 (2020), 452–489.
[30] Ankita Samaddar, Arvind Easwaran, and Rui Tan. 2020. SlotSwapper: a schedule randomization protocol for real-time WirelessHART networks. *SIGBED Rev.* 16, 4 (2020).
[31] C.F. Shih, T. Y. Wu, and W. Liao. 2010. DH-MAC: A Dynamic Channel Hopping MAC Protocol for Cognitive Radio Networks. In *ICC'10*. 1–5.
[32] Lei Tang, Yanjun Sun, Omer Gurewitz, and David B. Johnson. 2011. EM-MAC: A Dynamic Multichannel Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *MobiHoc '11*.
[33] A. Tzamaloukas and J.J. Garcia-Luna-Aceves. 2000. Channel-hopping multiple access. In *ICC'00*, Vol. 1. 415–419 vol.1.
[34] Thomas Watteyne, Ankur Mehta, and Kris Pister. 2009. Reliability through Frequency Diversity: Why Channel Hopping Makes Sense. In *PE-WASUN '09*. 116–123.
[35] Tang Zhong, Cheng Mengjin, Zeng Peng, and Wang Hong. 2010. Real-time communication in WIA-PA industrial wireless networks. In *ICCSIT'10*, Vol. 2. IEEE, 600–605.
[36] Dimitrios Zorbas, Panayiotis Kotzanikolaou, and Christos Douligeris. 2018. R-TSCH: Proactive jamming attack protection for IEEE 802.15. 4-TSCH networks. In *ISCC'18*. IEEE, 00766–00771.