Launching Smart Selective Jamming Attacks in WirelessHART Networks

Xia Cheng*, Junyang Shi*, Mo Sha*, Linke Guo+
*Department of Computer Science, State University of New York at Binghamton
+Department of Electrical and Computer Engineering, Clemson University
*{xcheng12,jshi28,msha}@binghamton.edu +linkeg@clemson.edu

IEEE INFOCOM 2021 - IEEE Conference on Computer Communications | 978-1-6654-0325-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/INFOCOM42981.2021.9488805

***Abstract*—As a leading industrial wireless standard, WirelessHART has been widely implemented to build wireless sensor-actuator networks (WSANs) in industrial facilities, such as oil refineries, chemical plants, and factories. For instance, 54,835 WSANs that implement the WirelessHART standard have been deployed globally by Emerson process management, a WirelessHART network supplier, to support process automation. While the existing research to improve industrial WSANs focuses mainly on enhancing network performance, the security aspects have not been given enough attention. We have identified a new threat to WirelessHART networks, namely smart selective jamming attacks, where the attacker first cracks the channel usage, routes, and parameter configuration of the victim network and then jams the transmissions of interest on their specific communication channels in their specific time slots, which makes the attacks energy efficient and hardly detectable. In this paper, we present this severe, stealthy threat by demonstrating the step-by-step attack process on a 50-node network that runs a publicly accessible WirelessHART implementation. Experimental results show that the smart selective jamming attacks significantly reduce the network reliability without triggering network updates.**

***Index Terms*—WirelessHART Networks, Selective Jamming, Industrial Wireless Sensor-Actuator Networks, Denial-of-Service Attack**

I. Introduction

Industrial Internet of Things (IoT) is revolutionizing the process industries and promises to be one of the largest potential economic effects in the future. Industrial networks connect sensors and actuators in industrial facilities, such as oil refineries, steel mills, and manufacturing plants, and serve as the communication infrastructures for various industrial IoT applications. Most industrial IoT applications have stringent demands for reliable and real-time communication in harsh industrial environments. Failure to meet such demands may lead to production inefficiency, financial loss, and safety threats. Traditionally, specifically chosen wired solutions, such as the highway addressable remote transducer (HART) communication protocol [1], have been designed to meet those stringent demands. Cables connect sensors and forward sensor readings to a control room where a controller makes control decisions, then sends commands to actuators. However, wired networks are often costly to deploy and maintain in harsh environments and difficult to reconfigure to accommodate new application requirements.

To reduce the cost and enhance the flexibility, industrial wireless sensor-actuator network (WSAN) technology has been developed and serves as a cost-effective way to connect sensors, actuators, and controllers in industrial facilities. Battery-powered wireless modules have been designed to easily and inexpensively retrofit existing sensors and actuators in industrial facilities without the need to run cables for communication and power. To meet the stringent reliability, real-time, and low-power requirements, the industrial WSAN standards, such as WirelessHART [2], make a set of specific design choices including employing the IEEE 802.15.4 physical layer, the time slotted channel hopping (TSCH) technology, and reliable graph routing that distinguish themselves from traditional wireless sensor networks (WSNs) designed for best-effort services [3]. Over the last decade, a large number of wireless networks that implement those standards have been deployed in industrial facilities. For instance, Emerson process management, one of the leading WirelessHART network suppliers, has deployed 54,835 WirelessHART networks globally and gathered 19.7 billion operating hours of experience [4]. A decade of real-world deployments has demonstrated the feasibility of using WirelessHART networks to achieve reliable low-power wireless communication in industrial facilities and exposed many limitations such as poor scalability [3] and error-prone configuration [5].

While the existing research to improve industrial WSANs focuses mainly on enhancing network performance, the security aspects have not been given enough attention. After careful analysis of the WirelessHART standard and extensive experimentation, we have identified a new threat to WirelessHART networks, namely smart selective jamming attacks, where the attacker first cracks the channel usage, routes, and parameter configuration of the victim network and then jams the transmissions of interest on their specific communication channels in their specific time slots. Compared to the constant jamming and random jamming, the smart selective jamming attacks are energy efficient and hardly detectable, and therefore pose a more severe, stealthy threat to WirelessHART networks. In this paper, we present this severe, stealthy threat by demonstrating the step-by-step attack process on a 50-node network [6] that runs a publicly accessible WirelessHART implementation [7]. Experimental results show that the smart selective jamming attacks significantly reduce the network reliability without triggering network updates.

The remainder of this paper is organized as follows. Section II presents the background of WirelessHART networks.

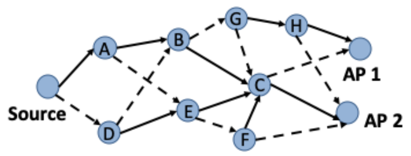


Fig. 1. A graph routing example. The solid lines represent the primary paths and the dashed lines represent the backup paths.

Section III introduces our threat model. Section IV demonstrates the step-by-step attack process. Section V describes our experimental studies. Section VI reviews related work. Section VII concludes the paper.

II. Background on WirelessHART Networks

A WirelessHART network is composed of a gateway, multiple access points, and a set of field devices (sensors and actuators) that form a multi-hop mesh network. A centralized network manager, a software module that runs on the gateway, is responsible for the network management, such as collecting link statistics, generating routes and transmission schedule, and maintaining the network operation. WirelessHART adopts the IEEE 802.15.4 physical layer and employs the TSCH technology in the MAC layer. TSCH divides time into slices of fixed length that are grouped into a slotframe. All devices in the network are time synchronized and share the notion of a slotframe that repeats over time. Channel hopping is used to mitigate effects of multipath fading and improve the robustness and the network capacity. Under TSCH, a pair of communicating devices uses the following function to determine their communication channel:

$$f = F[(ASN + ChannelOffset) \bmod S_{length}] \quad (1)$$

where F is a lookup table that stores a sequence of available physical channels. $ChannelOffset$ is used to identify each channel. S_{length} is the length of the channel sequence ($S_{length} \geq N_c$). N_c is the number of available physical channels. ASN is the Absolute Slot Number, defined as the total number of slots that has elapsed since the network started.

WirelessHART supports both source and graph routing. For each data flow, source routing provides a single route between source and destination, while graph routing provides a primary path and a series of backup routes to enhance the network reliability by taking advantage of route diversity. Figure 1 shows the example routes between the Source node and two access points (AP 1 and AP 2). A packet may take backup routes (through nodes D, E, F, G, or H) to reach AP 1 or AP 2 if it fails on the primary routing path (through nodes A, B, and C). The graph routing specified by WirelessHART requires each node to have at least two outgoing routes. A data-link protocol data unit (DLPDU) is used to carry the routing information and provides means for reliable communication in the data-link layer (DLL). As Figure 2 shows, a WirelessHART DLPDU consists of sequence number, network ID, destination and source addresses, DLL payload, message integrity code (MIC), and other fields. A network protocol data unit (NPDU) is carried in the DLL payload, which is composed of Graph ID, user data, and other fields.

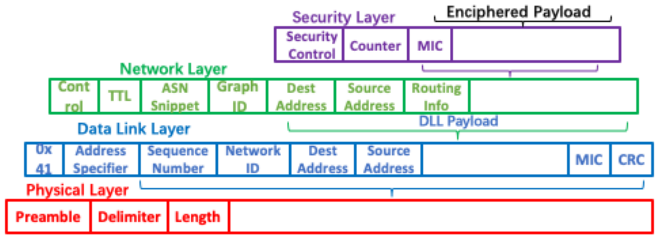


Fig. 2. DLPDU specified in WirelessHART.

WirelessHART does not require the devices to encrypt the DLPDU and NPDU headers due to the overhead concern. The source and destination addresses of a communicating link (defined as link source/destination address) are stored in the DLPDU header, while the address of the device that originally generated the packet and the final destination address of the packet (defined as route source/destination address) are stored in the NPDU header.

The network manager is responsible for collecting network diagnostic information and adjusting the routes and transmission schedule accordingly. Each network device generates a health report periodically (e.g., one every 15 minutes) and transmits it to the network manager. The health reports are composed of three types of response protocol data units (PDUs) for Command 787, 779 and 780. Command 787 reports on all detected neighbors that do not have links to this device. Command 779 is used to return the device health, such as battery status. Command 780 carries the statistical data for linked neighbors, such as the number of packets transmitted to and received from each neighbor. The network manager can make use of such information to determine whether it should regenerate the routes and reschedule the transmissions. As devices join or leave the network, the network manager updates its internal model of the WirelessHART network and uses this information to update the routes and transmission schedule. Although the WirelessHART standard provides an example of the generation interval of health reports (15 minutes), it leaves vendors to decide the actual value used in their networks. WirelessHART also leaves vendors to decide how to adjust routing based on the statistic information gathered from health reports. For instance, Emerson recommends that wireless field devices used for control and high speed monitoring have a higher packet reception ratio (PRR) threshold (70%) than general monitoring devices (60%) [8], while ASEA Brown Boveri (ABB) suggests the PRR threshold should be higher than 50% to establish a robust network [9].

In addition to health reports, each network device maintains a PathFailureTimer for each routing path, which is reset to a constant (PathFailInterval) when a DLPDU from that neighbor is received. When the PathFailureTimer for a neighbor reaches zero, a Path-Down alarm is generated and sent to the network manager. The value of PathFailInterval is left for vendors to decide.

III. Threat model

We consider a malicious device (attacker) in a WirelessHART network which is deployed in an open field or

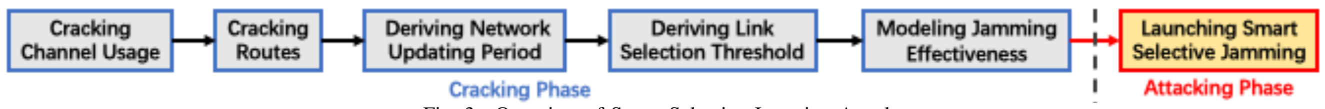

Cracking Channel Usage
Cracking Routes
Deriving Network Updating Period
Deriving Link Selection Threshold
Modeling Jamming Effectiveness
Launching Smart Selective Jamming
Cracking Phase
Attacking Phase

Fig. 3. Overview of Smart Selective Jamming Attack.

facility (e.g., an oil drilling plant) to support industrial wireless monitor and control applications.

Attacker's Objective. The intention of the attacker is to reduce the network reliability (i.e., the packet delivery ratio (PDR) of a target data flow) as much as possible by launching selective jamming attacks without being detected. To achieve this objective, the attacker must address the following four challenges:

- **Deployment-specific parameters:** There exist several important parameters including the link selection threshold for routing and the network updating period, which the WirelessHART standard allows vendors to decide. Moreover, a vendor may use different values for the same parameter in different deployments. The attacker must derive those values by observing the network behavior at runtime.
- **Fluctuation of low-power wireless links:** Unexpected transmission failures caused by the normal low-power link fluctuations may expose the attacker when performing attacks. The attacker must consider the network dynamics and adjust its attacks based on its runtime observations on the network condition.
- **Uncertainty of jamming effectiveness:** Many factors play an important role in the jamming effectiveness, such as the locations of the attacker, benign transmitter, and victim, the attacker's signal strength at the victim, and the timing when the jamming signal reaches the victim. The attacker must profile its jamming effectiveness and consider that when performing attacks.
- **Limited power supply:** The malicious device has limited power supply and cannot perform computation-intensive tasks, such as cracking information from data protected by the Advanced Encryption Standard (AES) 128-bit encryption.

Attacker's Resource. The attacker is assumed to be a device that has moderate computational capability and is capable of monitoring the transmission activities on each channel (the transmissions of DLPDU packets and their acknowledgments) and generating signals on each channel in the 2.4 GHz ISM band (e.g., a Raspberry Pi 3 Model B [10] that integrates with a Wi-Spy USB Spectrum Analyzer [11]). The attacker is powered by batteries or energy harvesting and deployed or airdropped into the WirelessHART network. We assume that the attacker does not have any prior knowledge on the deployment-specific parameters used in the WirelessHART network and can only gather information from the unencrypted packet headers transmitted in the network.

IV. Smart Selective Jamming Attacks

In this section, we provide a step-by-step presentation on the attack process.

A. Overview

To achieve the attacker's objective presented in Section III, the smart selective jamming attack consists of two phases: cracking phase and attacking phase. The attacker gathers the needed information by eavesdropping on transmissions in the network and performing exploratory jamming attacks in the cracking phase and launches the attacks in the attacking phase. Figure 3 shows the five steps in the cracking phase: (1) The attacker cracks the TSCH channel hopping sequences by silently observing the channel activities; (2) With the cracked channel usage information, the attacker cracks the routes by analyzing the eavesdropped transmissions (see Section IV-B); (3) With the cracked channel usage and routing information, the attacker launches exploratory jamming attacks to crack the network updating period by observing the time interval between two consecutive routing changes (see Section IV-C); (4) In an updating period, the attacker launches exploratory jamming attacks to identify the link selection threshold for routing (see Section IV-D); and (5) The attacker models its jamming effectiveness upon the previous exploratory jamming attacks (see Section IV-E). With the information gathered in the cracking phase, the attacker launches the smart selective jamming attacks to the target data flow (see Section IV-F).

The cracking method presented in the paper [12] can be used to complete the first step in the cracking phase. Here, we provide a brief summary of that method. The channel hopping sequences generated by the network devices when using Eq. 1 show a strong cyclic pattern. The attacker can identify the channel usage repetition cycle by observing the channel usage of a link. After deriving the channel usage repetition cycle, the attacker can identify the time slots that are scheduled for transmissions in each cycle and then create a table for each link. The table pairs each slot with a scheduled transmission to a communicating channel based on the observed channel activities. With those tables, the attacker can predict the channel used by each link in each time slot in the future. In addition, the attacker can derive the number of time slots in a slotframe and the number of active channels, and also synchronize its clock with the victim network.

B. Cracking the Routes

The attacker can use the method presented in the paper [13] to derive both source and graph routes from the eavesdropped packet headers. Here we present the method that cracks graph routes. The attacker can follow the same method to crack source routes by skipping the step of cracking backup routes. To crack the graph routes, the attacker can follow the following four steps:

1) **Eavesdropping on the on-air packets**: The attacker eavesdrops on each packet and records its capture time;

2) **Grouping and sorting the eavesdropped packets**: The attacker separates the eavesdropped packets into different groups by the Graph IDs stored in their packet headers and then sorts the packets in each group according to their capture time;
3) **Identifying the primary route:** As each DLPDU header stores the source and destination addresses of the communicating link, the attacker identifies all relay nodes located on the primary routing path by checking the sorted packets one by one until the link destination address is the same as the route destination address;
4) **Identifying the backup routes:** The attacker identifies the backup routes by selectively jamming each link on the primary routing path.

C. Deriving the Network Updating Period

After obtaining the routing information, the next step is to derive the network updating period U_P. As discussed in Section II, the network manager examines link statistics periodically and generates new routes and transmission schedule when needed. U_P is the time interval between two consecutive examinations. The network manager removes a link from routing if its PRR is below the preset PRR threshold PRR_T. U_P and PRR_T are deployment-specific parameters, which are not transmitted over the network. Therefore, the attacker cannot get them directly from the standard or information stored in the packet headers. However, the attacker can detect U_P by measuring the time duration between two consecutive routing changes. In a stable network, the attacker is likely to observe an U_P that is larger than its actual value because the network manager may skip network updates if no change is needed. To ensure the correctness of derived U_P, the attacker must perform exploratory jamming attacks on a best-suited link located on the primary routing path to make sure of routing changes. The best-suited link must be the first hop of a data flow since the data source node always transmits packets following its schedule and a relay node may skip a transmission if it fails to receive the packet correctly. It is beneficial for the attacker to select the weakest first-hop link (with the lowest PRR) among all data flows because the received signal strength (RSS) at the receiver of that link must either be low or close to the interference-plus-noise floor. When performing exploratory jamming, the attacker records the time when the link is removed from routing. The attacker repeats the above process again and obtains U_P by measuring the time duration between two consecutive routing changes.

To reduce the chance of being detected, the attacker must avoid destroying a link completely because a link failure triggers the Path-Down alarm (presented in Section II), which significantly increases the chance of exposing the attacker. Algorithm 1 illustrates the algorithm of launching exploratory jamming to crack U_P. The attacker executes Algorithm 1 twice when cracking U_P. We set m to three in our implementation, because attacking two-thirds of transmissions with a jamming success ratio of 60% reduces the PRR of a link by at least 40%, which is enough to trigger a routing update while keeping the link alive.

```
Algorithm 1: Exploratory jamming to crack $U_P$
  Output: $U_P$
1 Compute PRR of each first-hop link within its jamming
    range and select the one with the lowest PRR;
2 for $i = 1;; i++$ do
3     if $i\%m$ != 0 then
4         Jam the transmission over the selected link;
5     end
6     if Observe routing changes then
7         Record the time and break;
8     end
9 end
```

D. Deriving the Link Selection Threshold

As discussed in Section II, the network manager uses only the links with the PRRs larger than PRR_T for routing. If a route has a degraded link performance ($PRR < PRR_T$), it will be removed from routes. To make the jamming attacks stealthy, the attacker must crack PRR_T and attack the target data flow without triggering network updates by keeping the PRRs of all links above PRR_T. To crack PRR_T, the attacker gradually reduces the PRR of a link by launching exploratory jamming with a progressive increase in intensity in a series of network updating periods. The attacker starts from the lowest PRR observed in the routes and tests each possible value of PRR_T in descending order until triggering a network update. Ideally, the attacker should trigger the network update only once when cracking PRR_T. However, the fluctuation of low-power wireless link performance and imperfect jamming effectiveness may in reality trigger more network updates. Therefore, the attacker must use a carefully designed method to launch exploratory jamming. To reduce the chance of triggering additional network updates, the attacker should launch exploratory jamming to the most stable link in the network. Attacking a stable link also reduces the chance of triggering the Path-Down alarm. Here, we illustrate a method that cracks PRR_T without triggering more than one network update (see Figure 10 for evaluation results).

Algorithm 2 shows the process of testing whether a possible value of PRR_T (PRR_{test}) is the actual value in an updating period. Algorithm 2 has two modules: the *Estimation* module and *Examination* module. The Estimation module divides a network updating period into two sub-periods: observation sub-period and jamming sub-period. In the observation sub-period, the attacker silently observes the channel activities, counts the number of transmission failures, and updates the length of the jamming sub-period based on the runtime observations. In the jamming sub-period, the Examination module decides which transmissions should be jammed to ensure the resulting PRR is equal to PRR_{test}. The input of Algorithm 2 consists of four parameters: the PRR value that

Algorithm 2: Exploratory jamming to crack PRR_T

```
Input  : $PRR_{test}, R_{jam}, U_P, Link$
Output: $PRR_T$
1  Initialize the $dividingpoint$ according to Eq. 5;
2  for $i = 1; i \le U_P; i++$ do
3      if $i == dividingpoint$ then
4          Update the $dividingpoint$ according to Eq. 5;
5      end
6      if $i > dividingpoint$ then
7          Jam the current transmission on $Link$ if there
            are enough transmissions to compensate $T_f$;
8      end
9  end
10 if Observe the removal of target route from routing
    then
11     $PRR_T$ = $PRR_{test}$, then break;
12 end
13 else
14     if A PRR lower than $PRR_{test}$ observed then
15         Set $PRR_{test}$ to it;
16     end
17     else
18         Reduce $PRR_{test}$ by a preset testing step;
19     end
20 end
```

is currently in testing PRR_{test}; the jamming success ratio R_{jam}; the network updating period U_P; and the target link ($Link$). Algorithm 2 first computes the variable $dividingpoint$ that divides a network updating period into the observation and jamming sub-periods by using Eq. 5 (line 1) with the assumption that there is no transmission failure caused by link fluctuation in the observation sub-period. The loop (line 2 – 9) traverses all slotframes in the updating period (from 1 to U_P). In the $dividingpoint$th iteration, Algorithm 2 adjusts the $dividingpoint$ based on Eq. 5 if some transmission failures caused by link fluctuation are observed in the observation sub-period (line 3 – 5). Algorithm 2 keeps adjusting $dividingpoint$ until the newly computed $dividingpoint$ is equal to the previous value, and then starts the jamming sub-period. If the transmission failures caused by link fluctuation are uniformly distributed in the updating period, the above process of adjusting $dividingpoint$ based on runtime observations in the observation sub-period is a guarantee that the PRR in this network updating period is equal to PRR_{test}. In reality, the transmission failure caused by link fluctuation may not follow the uniform distribution. If the transmission failures happen more frequently in the jamming sub-period, the resulting PRR will be smaller than PRR_{test}, which makes the cracked PRR_T inaccurate. To address this issue, the attacker can employ a time series forecasting algorithm to estimate the transmission failures which will happen in the jamming sub-period and keep adjusting the estimation based on the actual observations in the jamming sub-period to ensure there are enough transmissions to compensate for unexpected failures. In our implementation, we use the Holt-Winters method that is one of the most effective time series forecasting algorithms [14]. The number of estimated transmission failures (T_f) can be expressed as

$$T_f = FR_{predict} \times T_R \tag{2}$$

where $FR_{predict}$ is the transmission failure ratio that is predicted by the Holt-Winters method and T_R is the number of the transmissions left in the remaining updating period.

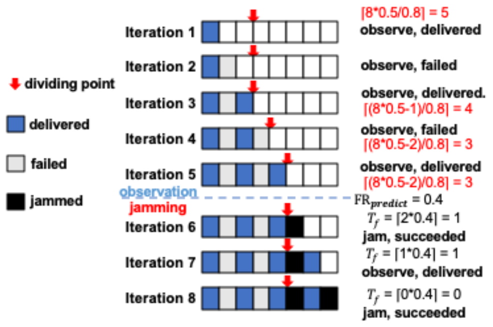


Fig. 4. Example of verifying PRR_{test} process, PRR_{test} is 0.5, U_P includes 8 slotframes and the jamming success ratio is 0.8.

Algorithm 2 first assumes that the current transmission can be jammed successfully and there will be T_f transmission failures caused by link fluctuation in the remaining network updating period. Algorithm 2 decides to jam a transmission if there are enough transmissions to compensate T_f (the resulting PRR is higher than PRR_{test}) (line 6 – 8). If Algorithm 2 observes a route removal, it gets PRR_T (line 10 – 12). If not, Algorithm 2 sets PRR_{test} to the next value (line 17 – 19) or a lower PRR which is owned by a route (line 14 – 16).

Figure 4 shows an example execution of Algorithm 2, where PRR_{test} is 0.5. In the example, we assume that the network updating period includes eight slotframes and the attacker has an 80% jamming success ratio. Algorithm 2 first computes $dividingpoint$ and finds that the jamming sub-period should have five slotframes. Let us assume that the attacker observes one transmission failure in the first three iterations. Therefore, Algorithm 2 adjusts the length of jamming sub-period to four in the end of Iteration 3. In Iteration 4, the attacker observes a transmission failure and adjusts the length of jamming sub-period to three. In Iteration 5, because the updated $dividingpoint$ is equal to the previous one (three), Algorithm 2 starts the jamming sub-period. In Iteration 6, Algorithm 2 estimates that there will be one possible transmission failure caused by link fluctuation based on Eq. 2, therefore it decides to jam the current transmission. In Iteration 7, Algorithm 2 decides to skip the attack on the current transmission because if the transmission failure happens in the last slotframe, the PRR will be 0.375 which is lower than PRR_{test}. In Iteration 8, Algorithm 2 decides to jam again to ensure that the resulting PRR is equal to PRR_{test}.

E. Modeling the Jamming Effectiveness

The attacker can model the jamming effectiveness based on the observations in the previous exploratory jamming attacks. Our analysis is based on a publicly accessible WirelessHART implementation, which employs three transmission attempts for each packet [7]. The first two attempts go through the primary route and the last attempt uses backup routes. To analyze the upper bound of jamming effectiveness, we assume that the primary routing path of the target data flow has n links and the attacker always successfully jams the third transmission attempt through the backup routes.

To simplify our explanation, we first assume that the attacker has a 100% jamming success ratio and the target data flow does not share routes with other data flows, and will drop these two assumptions later. The attacker first estimates the upper bound of the PDR degradation, which it can possibly cause on the target data flow by jamming an individual link i. Under graph routing, a packet uses the backup routes if the first two attempts through the primary routing path fail. To avoid triggering network updates, the attacker must keep the PRR of the victim link not less than PRR_T. Thus,

$$PRR = \frac{T_i - FD_i - Ji}{T_i + FD_i + FS_i + Ji} \geq PRR_T \tag{3}$$

where FD_i denotes the number of packets that fail in both two attempts on the primary routing path, FS_i denotes the number of packets that fail on the first attempt but succeed on the second attempt, J_i denotes the number of jammed packets in the jamming sub-period, and T_i denotes the total number of packets which are scheduled for transmission in the updating period.

When the PRR is equal to PRR_T, J_i achieves the maximum value. Accordingly, the upper bound of the PDR degradation on the target data flow that is possibly caused by jamming an individual link i (J_i/T_i) is

$$\frac{1 - PRR_T - (1 + PRR_T)FD_i/T_i - PRR_T FS_i/T_i}{1 + PRR_T} \tag{4}$$

In reality, the attacker cannot achieve a 100% jamming success ratio. The attacker must reserve more slotframes in the jamming sub-period to compensate jamming failures. The number of packets (the length of the jamming sub-period) scheduled for performing jamming is

$$\frac{(1 - PRR_T)T_i - (1 + PRR_T)FD_i - PRR_T FS_i}{R_{jam}(R_{jam} + PRR_T)} \tag{5}$$

The attacker can compute its jamming success ratio $Rjam$ by comparing the number of scheduled transmissions and the number of acknowledgments after jamming.

In most WirelessHART networks, multiple data flows exist and share one or more routes. Figure 5 shows an example. The data flows 147 $\rightarrow$ 121 (the target) and 149 $\rightarrow$ 121 share the routes between node 113 and 103 and between node 103 and 121. To continue our analysis on the upper bound of jamming performance, let us assume that all TS_i packets are transmitted successfully on route i for all data flows except the target data flow within the network updating period. Eq. 4 can be revised as $f(i) =$

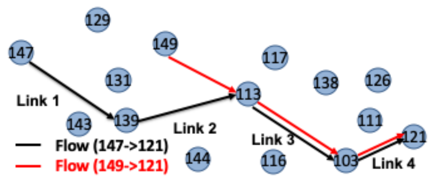


Fig. 5. Example of links shared by multiple data flows.

$$\frac{(1 - PRR_T)(1 + \frac{TS_i}{T_i}) - (1 + PRR_T)\frac{FD_i}{T_i} - PRR_T\frac{FS_i}{T_i}}{1 + PRR_T} \tag{6}$$

According to Eq. 6, the upper bound of the PDR degradation is significantly increased if the target link is shared by multiple data flows.

The upper bound of the PDR degradation on the target data flow which is possibly caused by jamming all n links is

$$PDR = \sum_{i=1}^{n} f(i) \tag{7}$$

F. Launching Smart Selective Jamming Attacks

```
Algorithm 3: Smart Selective Jamming
   Input  : $PRR_T$, $U_P$, $R_{jam}$
1  Initialize $dividingpoint[]$ according to Eq. 6;
2  for $k = 1; k \leq U_P; k++$ do
3      if $k == \sum_{i=1}^{n} dividingpoint[]$ then
4          Update $dividingpoint[]$ according to Eq. 6;
5      end
6      if $k > \sum_{i=1}^{n} dividingpoint[]$ then
7          Sort links by their PRRs in descending order;
8          for $j = 1; j \leq n; j++$ do
9              if $Link_j$ is not jammed in last iteration
                 then
10                 Update $FR_{predict}[j]$;
11                 Jam the current transmission if there are
                    more transmissions to compensate $T_f$;
12             end
13         end
14     end
15 end
```

With the cracked information, the next step is to launch the selective jamming attacks. Algorithm 3 presents how the attacker selects the target links and their transmissions for jamming by employing the *Estimation* module and *Examination* module. The input includes PRR_T, U_P and R_{jam}. Algorithm 3 first creates an array $dividingpoint[]$ that stores the initialized value of the dividing point of each link on the primary routing path according to Eq. 6 without considering the transmission failures caused by link fluctuation (line 1). The outside loop (line 2 – 15) traverses all slotframes in the network updating period (from 1 to U_P). In the observation

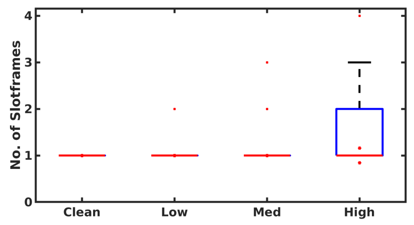
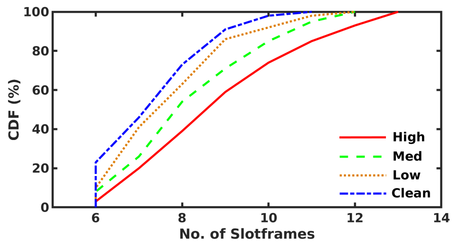
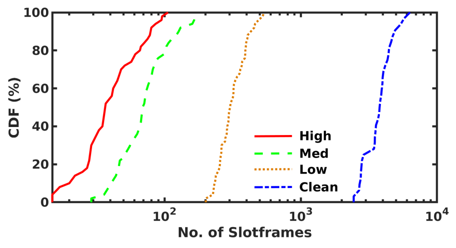


Fig. 6. Time consumed to crack the primary routing path under different conditions.



(a) With exploratory jamming.



(b) Without exploratory jamming.

Fig. 7. Time consumed to crack the backup routes.

TABLE I
DATA FLOWS SETUPS.

Flow	Sensor	Actuator	Period	Priority
1	044	046	640ms	1
2	047	008	640ms	2
3	036	004	1280ms	3
4	037	033	1280ms	4
5	027	035	1280ms	5

sub-period, the program keeps monitoring the transmission activities and adjusts the values of $dividingpoint[]$ (line 4) in the iteration that is previously scheduled as $dividingpoint$ according to the sum of $dividingpoint[]$ (line 3), until the sum of the updated $dividingpoint[]$ is equal to the previous one. Then, the PRRs of the links are updated and sorted in descending order (line 7) during the jamming sub-period. While traversing available links on the primary routing path from the link with the highest PRR (line 8), the attacker skips jamming a link if it was jammed in the last iteration (line 9) to avoid triggering the Path-Down alarm. Otherwise, Algorithm 3 makes the jamming decision by applying the same Examination module used in Algorithm 2 (line 11). In our implementation, we also use the Holt-Winters method to predict $FR_{predict}$ for each link and adopt a conservative policy to make sure the PRR of each link is always above PRR_T by taking more than T_f transmission failures into account.

V. Evaluation

To demonstrate the threat, we first perform microbenchmark experiments to measure the time consumed to crack the routes, network updating period, and link selection threshold, and examine the chance of triggering network updates. We then evaluate the performance of the smart selective jamming attacks and compare it against five baselines. We perform all experiments on our testbed that consists of 50 TelosB motes [15] placed throughout 22 office and lab areas on the second floor of an office building [6]. We configure the network to have two access points and 48 field devices that operate on five different channels in all experiments. As Table I lists, we set up five data flows with different sources, destinations, data periods, and priorities. Each time slot lasts $10ms$. Our testbed runs a publicly accessible WirelessHART implementation, which adopts the IEEE 802.15.4 physical layer, TSCH, and graph routing that employs three transmission attempts for each packet [7], while the attacking program runs on a Raspberry Pi with a 1.2GHz 64-bit quad-core processor and 1.0 GB memory. To examine the performance in different environments, we create four different wireless conditions (clean, low-interference, medium-interference, and high-interference) by using JamLab [16] to generate controlled interference with different signal strengths and repeat experiments 100 times in each environment.

A. Cracking the Routes

In the first set of experiments, we configure the attacking program to start cracking after eavesdropping on the transmissions during a certain number of slotframes and measure the number of eavesdropped slotframes consumed by the cracking program to crack the routes. The primary path used by the target (Flow 2) consists of seven nodes and six links. Our attacking program first identifies the primary routing path and then detects the backup routes by launching exploratory jamming to each link located on the primary routing path. Our attacking program achieves 100% success rate of cracking the routes under all wireless conditions. Figure 6 plots the boxplots of the time consumed by the attacking program to eavesdrop on the transmissions and then crack the primary path. As Figure 6 shows, the attacker can gather enough information to crack the primary routing path with a median value of one slotframe in the clean, low-interference, and medium-interference environments and up to four slotframes in the high-interference environment. This is because there is a high chance of using the primary routing path to deliver packets when the interference is weak, which makes the cracking easy. With the presence of strong interference, it takes longer for the attacker to identify the entire primary routing path because frequent failures on a link located on the primary routing path prevent the exposure of the following links.

Figure 7(a) plots the cumulative distribution function (CDF) of the time consumed by the attacker to crack the backup routes by launching the exploratory jamming to the links located on the primary routing path. The cracking process finishes within 13 slotframes under all wireless conditions. The cracking speed is slightly lower when the environment is noisier. This is because the transmission failures of the links on the primary path prevent the following links from being used in the noisy environments. As a comparison, Figure 7(b) plots the CDF of the time consumed by the attacker to crack the backup routes without launching the exploratory jamming. As Figure 7(b) shows, the time consumption decreases significantly when the interference increases. It takes at least 2,678

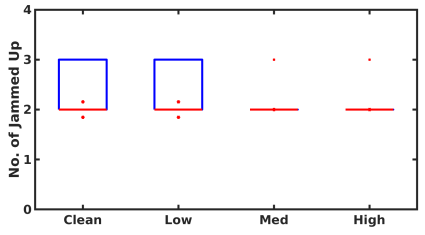


Fig. 8. Time consumed to crack the network updating period U_P.

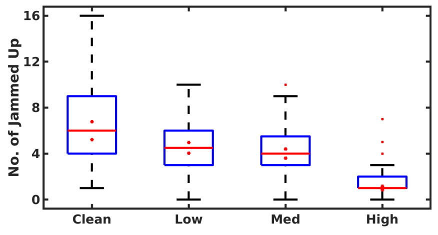


Fig. 9. Time consumed to crack the link selection threshold PRR_T.

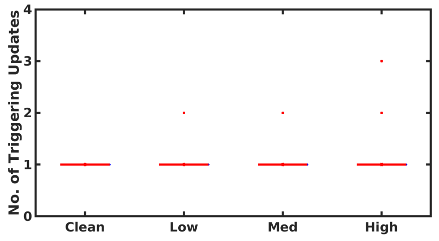


Fig. 10. Number of triggered network updates when cracking the link selection threshold.

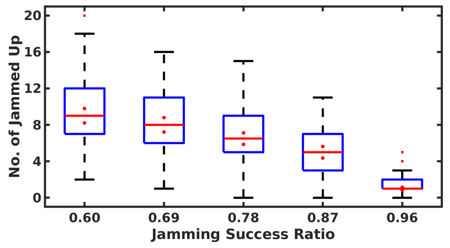


Fig. 11. Time consumption under different jamming success ratios.

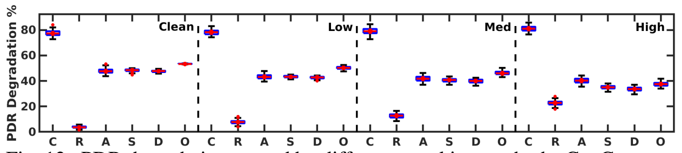


Fig. 12. PDR degradation caused by different attacking methods: C – Constant Jamming; R – Random Jamming; A – Smart Selective w/o Examination; S – Smart Selective w/o Estimation; D – Smart Selective Jamming; O – Optimal.

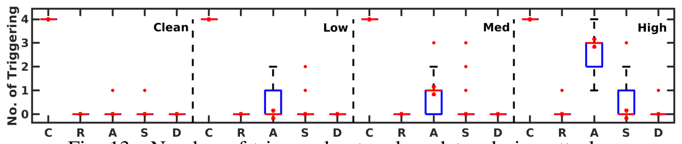


Fig. 13. Number of triggered network updates during attacks.

slotframes for the attacker to identify all backup routes in the clean environment, while it takes up to only 106 slotframes in the high-interference environment. This is because the backup routes are heavily used when the ambient environment is noisy. The long tails indicate that it may take a long time for the attacker to crack the routes if the attacker only observes silently, which emphasizes the importance of launching the exploratory jamming to speed up the cracking process.

B. Cracking U_P and PRR_T

In the second set of experiments, we launch the attacking program to crack the network updating period U_P and link selection threshold PRR_T and measure the time consumption and the chance of being detected. We observe 100% cracking accuracy for both U_P and PRR_T in all environments. Figure 8 plots the time consumption of cracking U_P under different wireless conditions when it is set to 51,200 time slots. As Figure 8 shows, the attacking program needs at least two updating periods (median value) to derive the value of U_P. It needs one more updating period in the clean and low-interference environments because it is more difficult for the attacker to trigger the routing updates by launching exploratory jamming when the environment is clean.

We set PRR_T to 60%. The lowest PRR values observed by the attacker in the clean, low-interference, medium-interference, and high-interference environments are 92%, 85%, 78%, and 71%, respectively. The corresponding number of updating periods scheduled for exploratory jamming are 33, 26, 19, and 12, respectively. Figure 9 plots the time consumed to crack PRR_T beyond the scheduled updating periods. The attacker has a jamming success ratio of 0.87 and reduces the testing PRR by 1% every time when launching exploratory jamming. As Figure 9 shows, the time consumption decreases when the interference increases. The median time consumption is $6U_P$, $5U_P$, $4U_P$, and $2U_P$ in the clean, low-interference, medium-interference, and high-interference environments, respectively. This is because the attacking program must use a larger jamming sub-period in the cleaner environment, which results in the increase of jamming failures and the difficulty of keeping the PRR within the expected range.

Figure 10 shows the number of network updates triggered by the exploratory jamming attack, which is one by design. As Figure 10 shows, the chance of triggering additional network updates is very low under all wireless conditions, which shows the exploratory jamming is hardly detectable.

To study the impact of the jamming success ratio, we repeat the experiments when the attacking program has different jamming success ratios by varying its transmission power. Figure 11 plots the time consumed to crack PRR_T beyond the scheduled updating periods in the low-interference environment when the jamming success ratios are 0.60, 0.69, 0.78, 0.87, and 0.96, respectively. As Figure 11 shows, the median time consumption of cracking PRR_T decreases significantly when the jamming success ratio increases. The median time consumption decreases from nine updating periods at 0.60, to six updating periods at 0.78, and then to one updating period at 0.96. These results show that the attacker can quickly crack the threshold PRR_T if it has a high jamming success ratio.

C. Jamming Performance

In this set of experiments, we evaluate the overall performance of the smart selective jamming attacks and compare it against five baselines: constant jamming; random jamming; smart selective jamming without its Estimation module; smart selective jamming without its Examination module; and the

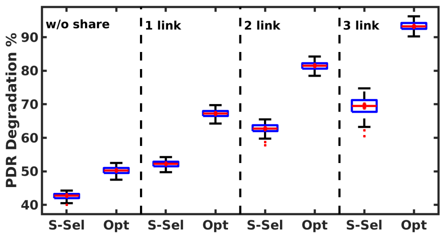


Fig. 14. PDR degradation when multiple data flows share links (S-Sel: Smart Selective Jamming, Opt: upper bound)

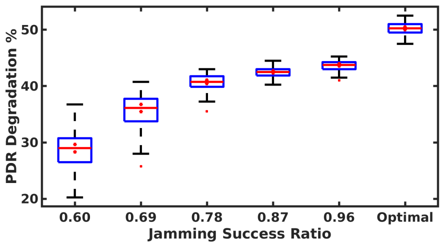


Fig. 15. PDR degradation under different jamming success ratios.

optimal method. Please note that the optimal method is based on backward data analysis using Eq. 7 and only for the purpose of comparison. We configure the attacking program to attack Flow 3 with four links on its the primary routing path and set U_P to 51,200 time slots, and PRR_T to 0.70.

Figure 12 plots the PDR degradation caused by different jamming methods under different wireless conditions and Figure 13 shows the number of triggered network updates during attacks. As Figure 12 shows, constant jamming introduces the largest damage (77% PDR degradation). However, it has the highest chance of being detected because it triggers 4X more network updates. Random jamming triggers fewer network updates, but it provides the smallest damage to the network. Compared to the constant and random jamming, the smart selective jamming has the lowest chance of being detected (with the smallest number of triggered network updates) while introducing significant damage close to the optimal method. The median PDR degradations are 49%, 43%, 39%, and 33% in the clean, low-interference, medium-interference, and high-interference environments, respectively. These results confirm the correctness of our analysis (Eq. 5). The upper bound of the PDR degradation caused by jamming decreases, while the transmission failure caused by link fluctuation due to interference increases. By comparing the performance of A, S, and D, the Examination and Estimation modules effectively reduce the chance of being detected in noisy environments.

To study the impact of shared links, we configure the victim data flow to use links shared with other data flows and repeat the experiments by varying the number of shared links. Figure 14 presents the jamming performance achieved by Smart Selective Jamming Algorithm in the low-interference environment. As Figure 14 shows, while the target data flow shares more links, the median value of the PDR degradation increases significantly, from 43% (w/o sharing link) to 68% (sharing three links). These increments accord with the results computed according to Eq. 6. The successful transmissions of other data flows compensate for the transmission failures due to jamming and cover up the jamming attacks.

We also repeat the experiments when the attacking program has different jamming success ratios. Figure 15 illustrates the jamming performance in the low-interference environment under different jamming success ratios. As Figure 15 shows, the jamming performance experiences an increase while we enhance the jamming success ratio. The median value of the PDR degradation increases from 29% (0.60) to 41% (0.78), reaches 46% (0.96). With a smaller chance of jamming success, it is difficult for the attacking program to achieve the expected number of jammed packets in the jamming sub-period, even if we adjust the size of the jamming sub-period according to the jamming success ratio.

VI. Related Work

Jamming attacks have been extensively studied in the literature of wireless mesh network and WSNs. Simply jamming a channel or the whole spectrum continuously, namely constant jamming, is energy inefficient and can be easily detected and located [17], while random jamming aims to save energy but is hardly effective [18]. Compared to constant and random jamming, selective (reactive) jamming stays quiet when the channel is idle but starts transmitting as soon as it senses activity on the channel [17]; therefore it is more energy efficient and more difficult to be detected [18], [19], [20], [21]. On the other hand, many approaches have been proposed in the literature to detect jamming attacks [22], [23], [24], [25], [26] and many countermeasures have been developed to mitigate the jamming effects [27], [28], [29], [30], [31], [32], [33]. There also exist defense solutions designed for specific applications [34], [35], [36], [37]. In this paper, we present a specific kind of selective jamming to WirelessHART networks, namely smart selective jamming attack, which aims to reduce the network reliability without being detected. This paper starts by investigating the security vulnerability of WirelessHART networks and then demonstrates that the attacker can crack the channel usage, routes, and parameter configuration of the victim network, and launch the smart selective jamming attacks to the target flows, which are energy efficient and hardly detectable.

VII. Conclusions

Our studies show that the attacker can reverse engineer the channel usage and graph routes of the victim WirelessHART network by silently observing the transmission activities, crack the victim network's parameter configurations with exploratory jamming attacks, and then perform selective jamming attacks to degrade network performance without being detected. In this paper, we present this severe, stealthy threat by demonstrating the step-by-step attack process on a 50-node network that runs a publicly accessible WirelessHART implementation.

Acknowledgment

This work was partially supported by the NSF through grants CRII-1657275 (NeTS), CNS-1947065, and CNS-2008049.

REFERENCES

[1] HART Communication Protocol and Foundation (Now the FieldComm Group). [Online]. Available: https://fieldcommgroup.org/

[2] WirelessHART, "WirelessHART." [Online]. Available: https://fieldcommgroup.org/technologies/hart/hart-technology

[3] C. Lu, A. Saifullah, B. Li, M. Sha, H. Gonzalez, D. Gunatilaka, C. Wu, L. Nie, and Y. Chen, "Real-Time Wireless Sensor-Actuator Networks for Industrial Cyber-Physical Systems," *IEEE Special Issue on Industrial Cyber Physical Systems*, vol. 104, no. 5, pp. 1013–1024, 2016.

[4] Emerson. [Online]. Available: https://www.emerson.com/en-us/expertise/automation/industrial-internet-things/pervasive-sensing-solutions/wireless-technology

[5] J. Shi and M. Sha, "Parameter Self-Configuration and Self-Adaptation in Industrial Wireless Sensor-Actuator Networks," in *INFOCOM*, 2019.

[6] "Testbed at the State University of New York at Binghamton." [Online]. Available: http://www.cs.binghamton.edu/\%7emsha/testbed

[7] "Wireless Cyber-Physical Simulator (WCPS)." [Online]. Available: http://wsn.cse.wustl.edu/index.php/WCPS:_Wireless_Cyber-Physical_Simulator

[8] System Engineering Guidelines IEC 62591 WirelessHART. [Online]. Available: https://www.emerson.com/

[9] ABB Wireless Industrial Network User Manual. [Online]. Available: https://search-ext.abb.com/library/Download.aspx?DocumentID=3BNP102912&LanguageCode=en&DocumentPartId=&Action=Launch

[10] "Raspberry Pi." [Online]. Available: https://www.raspberrypi.org/

[11] "Wi-Spy USB Spectrum Analyzer." [Online]. Available: http://www.wi-spy.co.uk/

[12] X. Cheng, J. Shi, and M. Sha, "Cracking the Channel Hopping Sequences in IEEE 802.15.4e-Based Industrial TSCH Networks," in *IoTDI*, 2019.

[13] ——, "Cracking Channel Hopping Sequences and Graph Routes in Industrial TSCH Networks," *ACM Transactions on Internet Technology*, vol. 13, no. 7, pp. 3481–3495, 2020.

[14] Holt-Winters Forecasting Method. [Online]. Available: https://www.ons.gov.uk/ons/guide-method/user-guidance/index-of-services/index-of-services-annex-b--the-holt-winters-forecasting-method.pdf

[15] TelosB Datasheet. [Online]. Available: https://insense.cs.st-andrews.ac.uk/files/2013/04/tmote-sky-datasheet.pdf

[16] C. A. Boano, T. Voigt, C. Noda, K. Römer, and M. Zuniga, "JamLab: Augmenting Sensornet Testbeds with Realistic and Controlled Interference Generation," in *IPSN*, 2011.

[17] W. Xu, W. Trappe, Y. Zhang, and T. Wood, "The Feasibility of Launching and Detecting Jamming Attacks in Wireless Networks," in *MobiHoc*, 2005.

[18] K. Grover, A. Lim, and Q. Yang, "Jamming and Anti-jamming Techniques in Wireless Networks: A Survey," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 17, no. 4, pp. 197–215, 2014.

[19] S. Fang, Y. Liu, and P. Ning, "Wireless Communications under Broadband Reactive Jamming Attacks," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 3, pp. 394–408, 2016.

[20] M. Wilhelm, I. Martinovic, J. B. Schmitt, and V. Lenders, "Short Paper: Reactive Jamming in Wireless Networks How Realistic is the Threat?" in *WiSec*, 2011.

[21] L. Zhang, F. Restuccia, T. Melodia, and S. M. Pudlewski, "Jam Sessions: Analysis and Experimental Evaluation of Advanced Jamming Attacks in MIMO Networks," in *MobiHoc*, 2019.

[22] M. Spuhler, D. Giustiniano, V. Lenders, M. Wilhelm, and J. B. Schmitt, "Detection of Reactive Jamming in DSSS-based Wireless Communications," *IEEE Transactions on Wireless Communications*, vol. 13, no. 3, pp. 1593–1603, 2014.

[23] M. Strasser, B. Danev, and S. Čapkun, "Detection of Reactive Jamming in Sensor Networks," *ACM Transactions on Sensor Networks*, vol. 7, no. 2, pp. 16:1–16:29, 2010.

[24] M. K. Hanawal, D. N. Nguyen, and M. Krunz, "Jamming Attack on In-band Full-duplex Communications: Detection and Countermeasures," in *INFOCOM*, 2016.

[25] Z. Lu, W. Wang, and C. Wang, "Modeling, Evaluation and Detection of Jamming Attacks in Time-Critical Wireless Applications," *IEEE Transactions on Mobile Computing*, vol. 13, no. 8, pp. 1746–1759, 2014.

[26] Z. Lu, W. Wang, and C. Wang, "From Jammer to Gambler: Modeling and Detection of Jamming Attacks Against Time-critical Traffic," in *INFOCOM*, 2011.

[27] A. Sheikholeslami, M. Ghaderi, H. Pishro-Nik, and D. Goeckel, "Energy-Efficient Routing in Wireless Networks in the Presence of Jamming," *IEEE Transactions on Wireless Communications*, vol. 15, no. 10, pp. 6828–6842, 2016.

[28] K. Firouzbakht, G. Noubir, and M. Salehi, "On the Performance of Adaptive Packetized Wireless Communication Links Under Jamming," *IEEE Transactions on Wireless Communications*, vol. 13, no. 7, pp. 3481–3495, 2014.

[29] L. Zhang, Z. Guan, and T. Melodia, "Cooperative Anti-jamming for Infrastructure-less Wireless Networks with Stochastic Relaying," in *INFOCOM*, 2014.

[30] Y. Zou, D. Yu, L. Wu, J. Yu, Y. Wu, Q. Hua, and F. C. M. Lau, "Fast Distributed Backbone Construction Despite Strong Adversarial Jamming," in *INFOCOM*, 2019.

[31] S. D'Oro, E. Ekici, and S. Palazzo, "Rate Maximization Under Reactive Jamming Attacks: Poster," in *MobiHoc*, 2016.

[32] V. Navda, A. Bohra, S. Ganguly, and D. Rubenstein, "Using Channel Hopping to Increase 802.11 Resilience to Jamming Attacks," in *INFOCOM*, 2007.

[33] A. Liu, P. Ning, H. Dai, and Y. Liu, "USD-FH: Jamming-resistant Wireless Communication Using Frequency Hopping with Uncoordinated Seed Disclosure," in *MASS*, 2010.

[34] A. Proano and L. Lazos, "Packet-Hiding Methods for Preventing Selective Jamming Attacks," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 1, pp. 101–114, 2012.

[35] R. Algin, H. O. Tan, and K. Akkaya, "Mitigating Selective Jamming Attacks in Smart Meter Data Collection Using Moving Target Defense," in *Q2SWinet*, 2017.

[36] M. Tiloca, D. D. Guglielmo, G. Dini, G. Anastasi, and S. K. Das, "DISH: DIstributed SHuffling against Selective Jamming Attack in IEEE 802.15.4e TSCH Networks," *ACM Trans. Sen. Netw.*, vol. 15, no. 1, 2018.

[37] A. Samaddar, A. Easwaran, and R. Tan, "SlotSwapper: A Schedule Randomization Protocol for Real-Time WirelessHART Networks," *ACM SIGBED Review*, vol. 16, no. 4, pp. 32–37, 2020.