Revealing Smart Selective Jamming Attacks in WirelessHART Networks

Xia Cheng, Junyang Shi, Mo Sha*, and Linke Guo

***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. According to the McKinsey report on future disruptive technologies, industrial IoT will contribute up to $47 trillion in added value globally by 2025 [2]. 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 [3], 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 [4], 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 [5]. 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 [6]. 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 [5] and error-prone configuration [7].

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 a step-by-step attack process. Specifically, this paper makes the following contributions:

Xia Cheng and Mo Sha are with the Knight Foundation School of Computing and Information Sciences at Florida International University, Miami, FL, 33199 USA (e-mail: xcheng@fiu.edu; msha@fiu.edu).

Junyang Shi is with Google. He contributed to this work while he was advised by Mo Sha in the Department of Computer Science at State University of New York at Binghamton, Binghamton, NY, 13902 USA (e-mail: jshi28@binghamton.edu).

Linke Guo is with the Department of Electrical and Computer Engineering, Clemson University, Clemson, SC, 29634 USA (e-mail: linkeg@clemson.edu).

*Corresponding author.

Part of this article was published in Proceedings of the INFOCOM [1].

Manuscript received Sep, 2021; revised Aug, 2022.

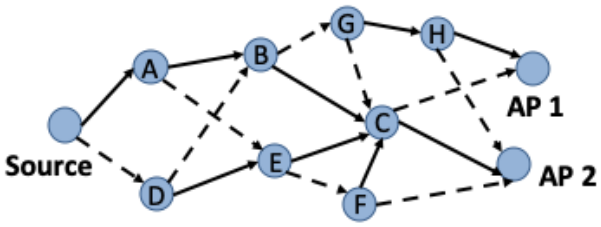


Fig. 1. A graph routing example. The solid lines represent the primary paths and the dashed lines represent the backup paths.

- We investigate the security vulnerability of WirelessHART networks and demonstrate how an attacker cracks the updating period and the link selection threshold used in a WirelessHART network;
- We model the effectiveness of smart selective jamming attacks and present a step-by-step attack process;
- We implement the attack process and test it on a physical testbed [8] with 50 devices that run the publicly accessible WirelessHART implementation [9]; Experimental results show that the smart selective jamming attacks can significantly compromise network reliability without triggering any network updates;
- We provide a set of insights on how we may secure WirelessHART networks against smart selective jamming attacks.

The remainder of this paper is organized as follows. Section II presents the background of WirelessHART networks. 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 and discusses our future work.

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. 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:

$$f = F[(ASN + C_{set}) \bmod S_{len}], \tag{1}$$

where ASN is the absolute slot number, defined as the total number of slots elapsed since the network started, C_{set} is the channel offset, which maps to one of available physical channels, F is the lookup table that maps each channel offset to its corresponding channel, and S_{len} is the length of a sequence of available physical channels.

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. 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). 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 DLPDU consists of address specifier, network ID, DLL payload, message integrity code (MIC), and other fields. DLPDU Specifier indicates the priority and data type of the message. There are four priority levels from high to low: Command, Process-Data, Normal, and Alarm. A network protocol data unit (NPDU) is carried in the DLL payload, which is composed of Graph ID, user data, and other fields. 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 are defined as link source/destination address, while the address of the device that originally generated the packet and the final destination address of the packet are defined as route source/destination address.

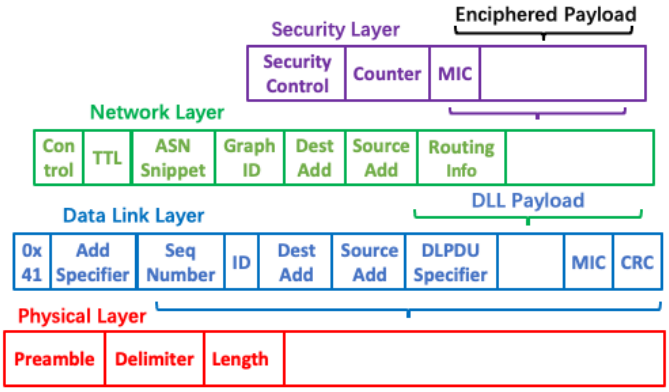


Fig. 2. DLPDU specified in the WirelessHART standard.

Each network device generates a health report periodically (e.g., one every 15 minutes) and transmits it to the network manager. The network manager can make use of such information to determine whether it should regenerate the routes and reschedule the transmissions. 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%) [10]. 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.

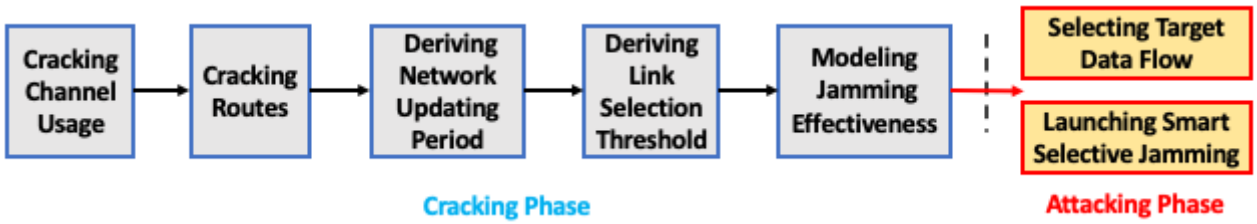


Fig. 3. Overview of smart selective jamming.

III. Threat Model

We consider a malicious device (attacker) in a WirelessHART network which is deployed in an open field or 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 [11] that integrates with a Wi-Spy USB Spectrum Analyzer [12]). 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.

TABLE I
Parameters needed for Smart Selective Jamming.

Parameter	Prerequisite	Credit
Channel Usage	None	[13]
Routes	Channel Usage	[14]
Updating Period	Channel Usage & Routes	Section IV-C
Link Threshold	Channel Usage & Routes & Period	Section IV-D

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 (see Section IV-B); (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). Table I summarizes the key parameters, which enable an attacker to launch the smart selective jamming attacks. 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). To maximize the damage to the network reliability, the attacker needs to carefully select a data flow as its target (see Section IV-G).

B. Cracking the Channel Usage and Routes

The attacker can use the method presented in the paper [13] to crack the channel usage. Here, we provide a brief summary of that method. The channel hopping sequences generated by

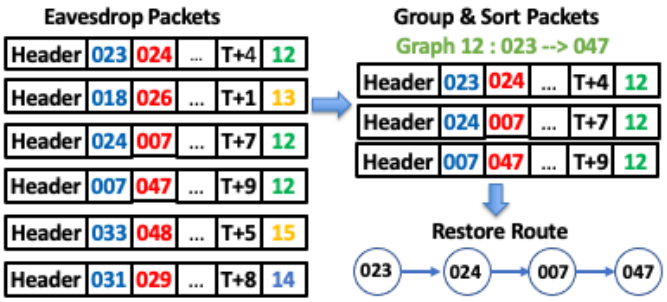


Fig. 4. Example of cracking the primary routing path.

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.

The attacker can use the method presented in the paper [14] 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.

Figure 4 shows an example on cracking the primary routing path. We assume that six packets have been eavesdropped by the attacker and three of them belongs to Graph 12 (the data flow from node 023 to node 047). The attacker sorts the packets according to their capture time $T+4$, $T+7$, and $T+9$, and identifies three links 023 $\rightarrow$ 024, 024 $\rightarrow$ 007, and 007 $\rightarrow$ 047. Thus, the primary routing path is 023 $\rightarrow$ 024 $\rightarrow$ 007 $\rightarrow$ 047. The attacker identifies the backup routes of node 023 by jamming the link 023 $\rightarrow$ 024 and repeats the process until obtaining all backup routes.

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 the most vulnerable link located on the primary routing path to make sure of routing changes. The most vulnerable 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.

Algorithm 1: Exploratory jamming to crack U_P

```
Output: U_P
Compute PRR of each first-hop link within its jamming
  range and select the one with the lowest PRR;
for i = 1;;i++ do
    if i%m != 0 then
        Jam the transmission over the selected link;
    end
    if Observe routing changes then
        Record the time and break;
    end
end
```

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. One of the primary design goals of the exploratory jamming is to trigger the routing update only once, which

minimizes exposure to the network manager that manages the victim network.

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 13 for evaluation results).

Algorithm 2: Exploratory jamming to crack PRR_T

```
Input  : PRR_test, R_jam, U_P, Link
Output: PRR_T
1  Initialize div according to Eq. 5;
2  for i = 1; i ≤ U_P; i++ do
3      if i == div then
4          Update the div according to Eq. 5;
5      end
6      if i > div 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
```

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 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 div 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 divth iteration, Algorithm 2 adjusts the div 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 div until the newly computed div 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 div 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 [15]. The number of estimated transmission failures (T_f) can be expressed as

$$T_f = FR_{pre} \times T_R, \tag{2}$$

where FR_{pre} 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.

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 5 shows an example execution of Algorithm 2, where PRR_{test} is 0.5. In the example, we assume that the

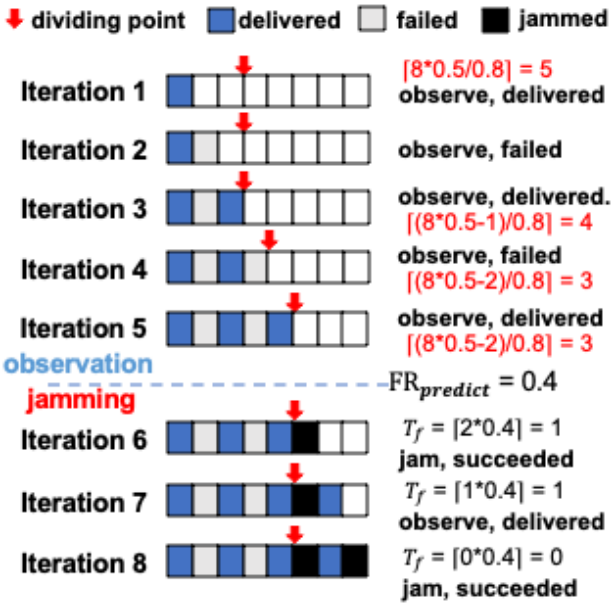


Fig. 5. Example of verifying PRR_{test} process, PRR_{test} is 0.5, U_P includes 8 slotframes and the jamming success ratio is 0.8.

network updating period includes eight slotframes and the attacker has an 80% jamming success ratio. Algorithm 2 first computes div 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 div 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 [9]. 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

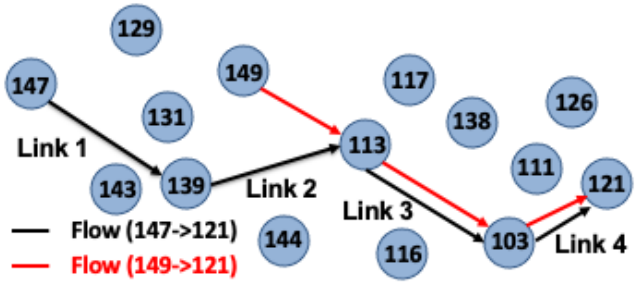


Fig. 6. Example of links shared by multiple data flows.

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 6 shows an example. The data flows 147 → 121 (the target) and 149 → 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 $f(i) =$

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

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 div[] according to Eq. 6;
2  for k = 1; k ≤ U_P; k++ do
3      if k == Σ_{i=1}^{n} div[] then
4          Update div[] according to Eq. 6;
5      end
6      if k > Σ_{i=1}^{n} div[] then
7          Sort links by their PRRs in descending order;
8          for j = 1; j ≤ n; j++ do
9              if Link_j is not jammed in last iteration
                 then
10                 Update FR_pre[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 $div[]$ 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 sub-period, the program keeps monitoring the transmission activities and adjusts the values of $div[]$ (line 4) in the iteration that is previously scheduled as div according to the sum of $div[]$ (line 3), until the sum of the updated $div[]$ 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_{pre} 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.

G. Selecting the Target Data Flow

Algorithm 4: Target Data Flow Selection Method

```
   Input : Flow[]
   Output: Target
1  for j = 1; j ≤ m; j++ do
2      for i = 1; i ≤ n; i++ do
3          Count the number of transmission failures
             caused by link fluctuation;
4          Calculate the upper bound of the PDR
             degradation on Flow[j] caused by jamming
             Link_i according to Eq.6;
5      end
6      Calculate the transmission failure ratio of Flow[j];
7      Estimate the upper bound of the PDR degradation
         on Flow[j] according to Eq.7;
8  end
9  for k = 1; k ≤ U_p; k++ do
10     for j = 1; j ≤ m; j++ do
11         if Flow[j] is transmitting a packet with the
             highest priority then
12             Select it as Target and break;
13         end
14     end
15     if Target is Null then
16         Sort Flow[] by the upper bound of the PDR
             degradation in descending order;
17         if A data flow with the highest PDR
             degradation then
18             Select it as Target;
19         end
20         else
21             Sort the flows with similar potential PDR
                 degradation values by the transmission
                 failure ratio in ascending order;
22             Select the data flow with the lowest
                 transmission failure ratio as Target;
23         end
24     end
25 end
```

After modeling the jamming effectiveness, the attacker begins to estimate the upper bound of the PDR degradation it can cause on each data flow within its overhearing range and selects the most important or most vulnerable one as its target. The attacker first identifies the priority levels of the packets that are transmitting over each data flow. If there exists a single data flow whose messages have the highest priority (e.g., packets indicated by "Command"), the attacker selects it as its target. If two or more data flows share the same priority, the attacker selects the data flow with the highest upper bound of the PDR degradation possibly caused by the smart selective jamming attacks as its target. If two or more data flows share similar PDR degradations, the data flow with the lowest transmission failure ratio is selected as the target. Algorithm 4 illustrates the target data flow selection process. The input of Algorithm 4 includes all data flows within the

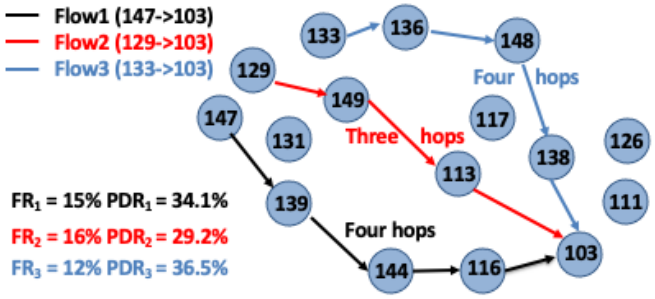


Fig. 7. Example of the target data flow selection process.

attacker's overhearing range $Flow[]$. Algorithm 4 first collects the transmission statistics of each data flow in a two-level nested loop (line 1 – 8). The outside loop traverses all data flows (from $Flow[1]$ to $Flow[m]$), while the inside loop traverses the links of a data flow (from $Link_1$ to $Link_n$). Algorithm 4 counts the number of transmission failures caused by the link fluctuation (line 3) and calculates the upper bound of the PDR degradation on this data flow caused by jamming each link separately according to Eq 6 (line 4). By adding up the transmission failures on each link, the transmission failure ratio caused by the link fluctuation on each data flow is calculated (line 6). Algorithm 4 then estimates the upper bound of the PDR degradation on each data flow according to Eq 7 (line 7). The outside loop (line 9 – 25) traverses all slotframes in the network updating period (from 1 to U_P). If a data flow is transmitting data with the highest priority among all data flows, this data flow is selected as the target to jam the transmissions of high importance (line 10 – 14). Otherwise, Algorithm 4 sorts all data flows by the upper bound of the PDR degradation in descending order and selects the data flow, which has the highest potential PDR degradation caused by jamming attacks, as the target (line 16 – 19). To avoid switching the target back and forth due to slight variations on the upper bound of the PDR degradation, the difference of the upper bound of the PDR degradation between the target data flow and any other data flow should be larger than a threshold. The threshold can be set equal to or larger than the variation of the upper bound of the PDR degradation on a data flow in two consecutive periods. We set this variation threshold to 2% in our implementation. If no data flow is selected because of those similar PDR degradation values, Algorithm 4 sorts all data flows with similar PDR degradation values by the transmission failure ratio in ascending order and selects the data flow with the lowest transmission failure ratio as the target (line 20 – 23). As discussed in Section IV-D, when the transmission failures caused by link fluctuation happen less frequently and more uniformly, the attacker is more likely to achieve the desirable jamming effectiveness by keeping the PRRs of all links equal or close to PRR_T. The attacker executes Algorithm 4 periodically to handle the network changes. We set the execution period to U_P in our implementation.

Figure 7 shows an example of the target data flow selection process. In this example, we assume that there are three data flows within the overhearing range of an attacker (Flow 1, Flow 2, and Flow 3). The primary routing paths of those

Fig. 8. Testbed consisting of 50 TelosB motes placed throughout 22 office and lab areas on the second floor of an office building. The device IDs range from 000 to 049.

data flows consist of four hops, three hops, and four hops, respectively. At the start of a network updating period, the attacker calculates the transmission failure ratio of each data flow (FR_1, FR_2, and FR_3) and then estimates the upper bound of the PDR degradation on each data flow (PDR_1, PDR_2, and PDR_3) according to Eq.6 and Eq.7. We assume that these three data flows share the same priority level. Therefore, the attacker begins to compare PDR_1, PDR_2, and PDR_3. Both PDR_1 and PDR_3 are greater than PDR_2, because Flow 1 and Flow 3 include one more hop on their primary routing paths. The difference between PDR_3 and PDR_1 is 2.4%, larger than the preset variation threshold (2%). Therefore, the attacker selects Flow 3 as its target and performs smart selective jamming attacks to the packet transmissions on Flow 3 in each slotframe of the current updating period. However, when Flow 1 or Flow 2 is transmitting a message with a higher priority level, the attacker changes its target temporarily and performs smart selective jamming attacks to the most important packet transmissions.

V. Evaluation

To demonstrate the threat, we first perform a series of 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 perform microbenchmark experiments to measure the time consumed by the target data flow selection method to identify the new target data flow. Next, we evaluate the jamming performance of the smart selective jamming attacks without enabling the target data flow selection method and compare it against five baselines. Finally, we evaluate the jamming performance of attacking the data flow provided by the target data flow selection method and compare it against the performance achieved by attacking other data flows. We perform all experiments on our testbed that consists of 50 TelosB motes [16] placed throughout 22 office and lab areas

TABLE II
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	4

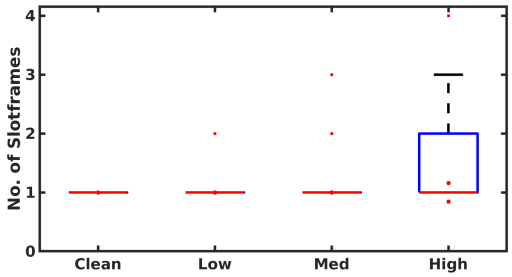
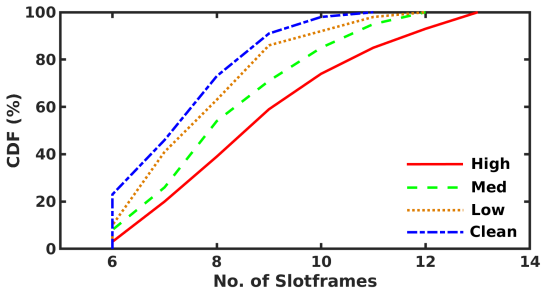


Fig. 9. Time consumed to crack the primary routing path under different conditions.

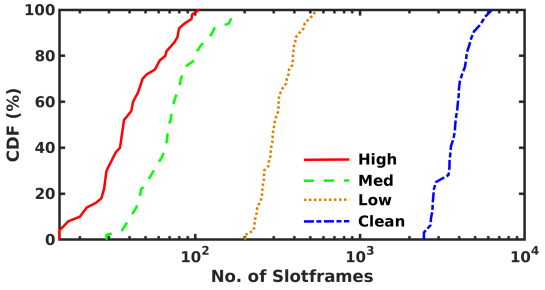
on the second floor of an office building [8]. Figure 8 shows the testbed deployment. We configure the network to have two access points and 48 field devices that operate on five different channels in all experiments. As Table II lists, we set up five data flows with different sources, destinations, data periods, and priorities for the experiments in Section V-A, V-B, and V-D. 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 [9], 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 of the attacking program in different environments, we create three different wireless conditions (i.e., low-interference, medium-interference, and high-interference) by using JamLab [17] to generate controlled interference with various strengths and disable JamLab to create the clean environment where the averaged PRRs of all links on the target data flow are above 98%. As a comparison, the averaged PRRs of all links on the target data flow range from 89% to 91% in the low-interference environment, the averaged PRRs vary between 80% and 83% under interference in the medium-interference environment, and the averaged PRRs range from 73% to 75% in the high-interference environment. We 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 9 plots the



(a) With exploratory jamming.



(b) Without exploratory jamming.

Fig. 10. Time consumed to crack the backup routes.

boxplots of the time consumed by the attacking program to eavesdrop on the transmissions and then crack the primary path. As Figure 9 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 10(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 slower when the environment is noisier, leading to an increase in the chance of transmission failures on both primary and backup routes. Therefore, the relay nodes fail to receive the packets more frequently in the noisy environments, which prevents the following nodes from being used. Under such scenarios, the attacker cannot perform exploratory jamming to those unused relay nodes and has to wait till the next slotframe. As a comparison, Figure 10(b) plots the CDF of the time consumed by the attacker to crack the backup routes without launching the exploratory jamming. As Figure 10(b) shows, the time consumption decreases significantly when the interference increases. It takes at least 2,678 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

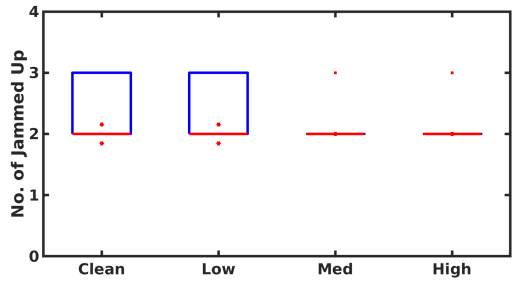


Fig. 11. Time consumed to crack the network updating period U_P.

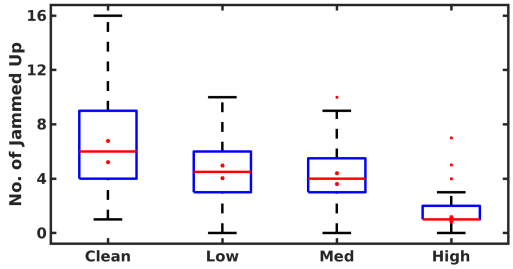


Fig. 12. Time consumed to crack the link selection threshold PRR_T.

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 11 plots the time consumption of cracking U_P under different wireless conditions when it is set to 51,200 time slots. As Figure 11 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

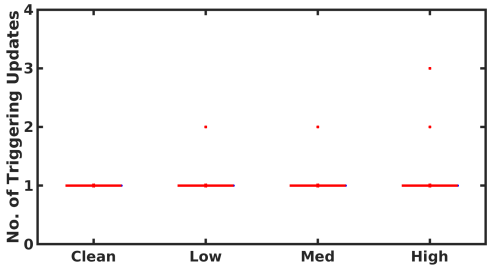


Fig. 13. Number of triggered network updates when cracking the link selection threshold.

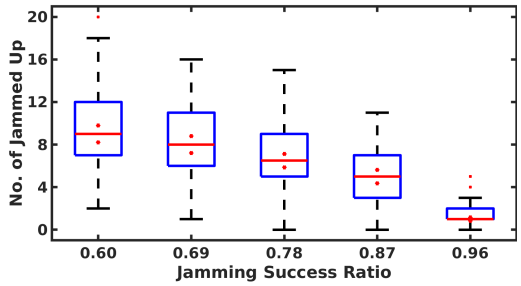


Fig. 14. Time consumption under different jamming success ratios.

TABLE III
DATA FLOWS SETUPS FOR TARGET DATA FLOW SELECTION.

Flow	Sensor	Actuator	Period	Priority
1	018	016	640ms	3
2	044	023	640ms	3
3	046	008	640ms	3
4	037	022	1280ms	3
5	027	035	1280ms	3
6	036	001	1280ms	3

of updating periods scheduled for exploratory jamming are 33, 26, 19, and 12, respectively. Figure 12 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 12 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 13 plots the number of network updates triggered by the exploratory jamming attack, which is one by design. It is very difficult for the network manager to detect the jamming attacks by observing an occasional network update. As Figure 13 shows, the chance of triggering additional network updates is very low under all wireless conditions. Therefore, 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 14 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 14 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. Selecting the Target Data Flow

In this set of experiments, we evaluate our target data flow selection method by measuring the time consumption of selecting a new target when the network topology or condition

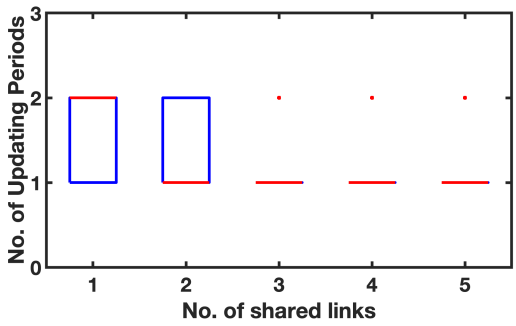


Fig. 15. Time consumption of selecting the new target data flow when we configure Flow 4 to share different numbers of links (from one to five) on its primary routing path with Flow 6's in a randomly selected time slot of an updating period. Flow 4 and 6 do not share any links on their primary routing paths before we make changes.

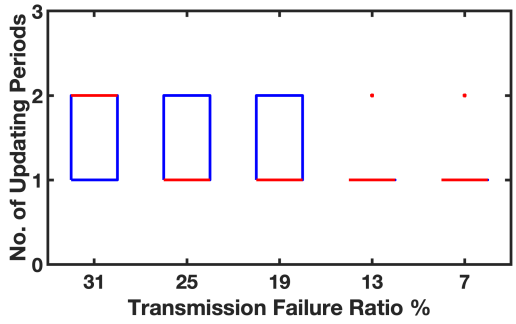


Fig. 16. Time consumed to identify the new target when we decrease the transmission failure ratio of Flow 6 from 31% 7%.

changes. We configure six data flows with the same priority but different sources, destinations, and data periods on our testbed, as Table III lists. We manually change the primary routing paths of different data flows and measure the time consumed by the attacker to identify new targets. Figure 15 plots the time consumption of identifying the new target flow when we configure Flow 4 to share different numbers of links (from one to five) on its primary routing path with Flow 6's in a randomly selected time slot of an updating period. We repeat each experiment 100 times. As Figure 15 shows, the attacking program spends less time on selecting the new target data flow when the number of shared links between Flow 6 and Flow 4 increases. For example, the median time consumption is two updating periods while only one link is shared, while the median values are one updating period while more links are shared. This is because sharing one link on their primary paths only introduces a very small change on the upper bound of the PDR degradation on Flow 6. Therefore, the attacking program needs more time to confirm that Flow 6 should be the new target. When more links are shared, it is easier for the attacking program to select the new target because the changes on the upper bound of the PDR degradation are much larger than than the preset threshold.

We also vary the transmission failure ratios of different data flows and examine their impacts on the time consumption. For instance, we use Jamlab to generate controlled interference with different signal strengths, which decreases the transmission failure ratio of Flow 6 from 36% to 31%, 25%, 19%, 13%, and 7%, respectively. Figure 16 plots the time consumption of selecting the new target data flow. As Figure 16 shows,

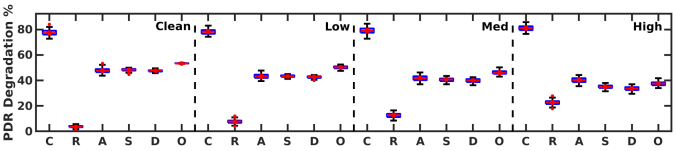


Fig. 17. PDR degradation caused by different attacking methods: C – Constant Jamming; R – Random Jamming; A – Smart Selective without Examination; S – Smart Selective without Estimation; D – Smart Selective Jamming; O – Optimal.

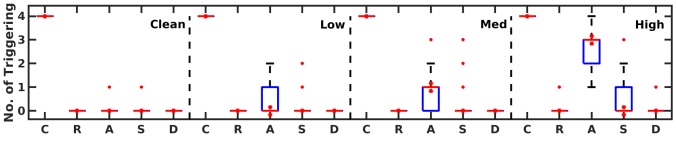


Fig. 18. Number of triggered network updates during attacks.

the attacking program needs one more updating period (the median value) to select the new target when the transmission failure ratio decreases from 36% to 31%. This is because it is more difficult for the attacking program to confirm that the increase of the upper bound of the PDR degradation is larger than the preset threshold when the transmission failure ratio caused by the link fluctuation varies in a small range. The results plotted in Figure 15 and 16 demonstrate that the target data flow selection method can efficiently select a new target when observing the change on the upper bound of the PDR degradation under different conditions

D. Jamming Performance without Target Data Flow Selection

In this set of experiments, we evaluate the overall performance of the smart selective jamming attacks on a given data flow 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 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 17 plots the PDR degradation caused by different jamming methods under different wireless conditions and Figure 18 shows the number of triggered network updates during attacks. As Figure 17 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 constant and random jamming, the smart selective jamming is much harder to be detected by the network manager, because the median value of triggered network updates is zero. Meanwhile, the damage introduced by the smart selective jamming is close to the one caused by 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

TABLE IV
COMPARISON OF ENERGY CONSUMPTION.

Method	Degradation	Packets	Energy Consumption
Constant	77%	256,000	88.2J
Random	4%	51,200	17.6J
Selective	49%	1352	2.2J
Optimal	52%	1248	2.0J

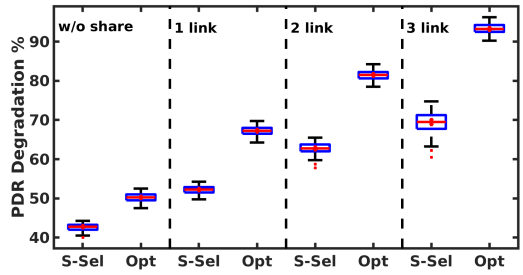


Fig. 19. PDR degradation when multiple data flows share links (S-Sel: Smart Selective Jamming, Opt: upper bound)

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 module and Estimation module of the smart selective jamming method effectively reduce the chance of being detected in noisy environments.

We also evaluate the energy efficiency of the smart selective jamming and compare it against other jamming methods. Table IV lists the number of jamming packets and the energy consumption within an updating period in the clean environment. The smart selective jamming introduces 49% PDR degradation by consuming only 2.2J, which is very close to the one caused by the optimal method. As a comparison, the constant jamming consumes 88.2J to generate 79% PDR degradation, while the random jamming provides 4% PDR degradation by consuming 17.6J. The results clearly show that the smart selective jamming is much more energy efficient than the traditional jamming methods.

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 19 presents the jamming performance achieved by Smart Selective Jamming Algorithm in the low-interference environment. As Figure 19 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

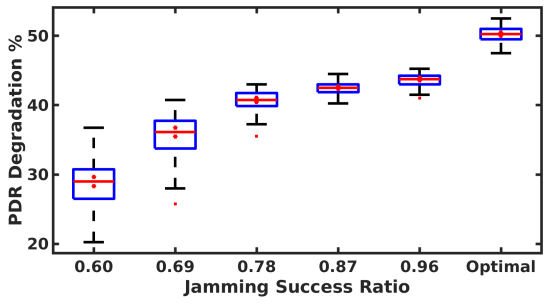


Fig. 20. PDR degradation under different jamming success ratios.

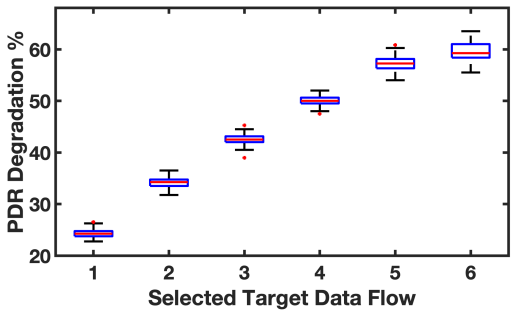


Fig. 21. PDR Degradation when attacking different data flows. The numbers of the links on the primary routes of six data flows are two, three, four, five, six, and six, respectively.

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 20 illustrates the jamming performance in the low-interference environment under different jamming success ratios. As Figure 20 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.

E. Jamming Performance with Target Data Flow Selection

In this set of experiments, we evaluate the overall performance of the smart selective jamming attacks when we enable the target data flow selection method and compare it against the performance when we randomly select a data flow as the target. The PDR degradation caused by the attacking program with the target data flow selection method is always the greatest in all our experiments. For example, Figure 21 presents the measured PDR degradation when we launch the smart selective jamming attacks to attack different data flows. The target data flow selection method selects Flow 6 as its target. As Figure 21 shows, the median PDR degradation caused by jamming the transmissions on Flow 6 is 60.7%, much higher than the damages caused when attacking other data flows. For example, the median PDR degradation is only 24.3% if the attacking program attacks Flow 1, while the median PDR degradation is 56.4% if the attacking program attacks Flow 5. This is because the upper bound of the PDR degradation on the target selected by the target data flow selection method is the highest and the actual performance achieved by launching the smart selective jamming attacks to such a target is the greatest. The experimental results confirm that the target data flow selection method can enhance the jamming performance of the smart selective jamming attacks.

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 [18], while random jamming aims to save energy but is hardly effective [19]. 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 [18]; therefore it is more energy efficient and more difficult to be detected [19]–[22]. For instance, Zhang et al. presented a reinforcement learning based algorithm that helps the attacker adapt its jamming methods to dynamic environments to improve the jamming performance [22]. On the other hand, many approaches have been proposed in the literature to detect jamming attacks [23]–[27] and many countermeasures have been developed to mitigate the jamming effects [28]–[30]. For instance, Zou et al. presented a jamming-resilient backbone construction algorithm [31]. Lu et al. studied modeling, evaluation, and detection of jamming attacks in wireless networks [26], [27]. D'Oro et al. proposed solutions to maximize the network performance when reactive jamming attacks are ongoing [32]. Navda et al. [33] and Liu et al. [34] suggested using channel hopping (frequency hopping) to increase resilience to jamming attacks. There also exist defense solutions designed for specific applications [35]–[38]. For instance, Proano et al. proposed to defend against selective jamming attacks that are launched by performing real-time packet classification at the physical layer by combining cryptographic primitives with physical-layer attributes [35]. Tiloca et al. developed a method that randomly permutes the time slots and channel utilization patterns for TSCH based wireless networks [36]. Samaddar et al. proposed a scheduling method that increase the randomness of the TSCH channel hopping sequence [37]. Pirayesh et al. developed a jamming-resilient receiver that mitigates the unknown interference using an optimized neural network to secure ZigBee communications [38]. This paper focuses on revealing the threat of smart selective jamming to WirelessHART networks and motivating the developments of new defense solutions, it is therefore complementary to the existing work.

Jamming attacks have also been studied in the context of Bluetooth, GPS, and cellular networks. For instance, Albazrqaoe et al. developed a novel dual-radio architecture where two Bluetooth-compliant radios coordinate with each other on learning the hopping sequence of indiscoverable Bluetooth networks [39], [40]. Dr. Chien developed an adaptive notch filter that is composed of a second-order infinite-impulse response filter with a lattice structure to detect, estimate, and block continuous jamming signals [41]. In recent years, the MIMO-based jamming mitigation techniques are applied in cellular networks [42]. For example, Akhlaghpasand et al. developed a framework that consists of a linear estimator and a bilinear equalizer to provide protection for massive MIMO systems in spatially correlated channels [43]. Vinogradova et al. proposed to employ the received signal projection onto the estimated signal subspace to nullify the jamming signal [44]. However, those anti-jamming methods are not directly applicable to WirelessHART networks. 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 data flow provided by the target data flow selection method, which are energy efficient and hardly detectable.

The WirelessHART standard offers multiple security features that protect the network against such attacks as denial of service (DoS), MAC spoofing, man in the middle (MITM), and authentication and encryption cracking. For instance, WirelessHART employs the AES 128-bit symmetric-key cryptography to protect the packet payload and uses the MIC and cyclic redundancy check (CRC) to detect errors. A series of enhancements has been developed to enhance the security of WirelessHART networks [45]–[48]. Unfortunately, the existing designs cannot prevent the attacker from launching the smart selective jamming attacks, which has been reported as a new, realistic threat to WirelessHART networks in this paper.

VII. Conclusions and Future Work

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 smart selective jamming attacks to the target provided by the target data flow selection method to degrade network performance without being detected. Compared to the constant jamming attacks and the random jamming attacks, the smart selective jamming attacks are energy efficient and hardly detectable, thus 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 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.

Our studies suggest two potential solutions to help WirelessHart Networks defend against smart selective jamming attacks. As discussed in Section IV-B, an attacker can derive the routing information of the victim network from the unencrypted fields stored in the packet headers. The attacker can crack both primary and backup routes using a small amount of time bounded by the data generation period, as plotted in Figure 9 and 10(a). Among all information carried by the packet header, Graph ID, original source address, and final destination address are the key in the cracking process. Without such information, it is very hard for an attacker to classify the packets and crack the routes. With the consideration of the encryption and decryption overhead, it is beneficial to only encrypt those three fields instead of all information in the packet header. Our studies also show that the health reports specified in the WirelessHART standard

do not carry the information, which can be used to relate a data flow's performance degradation to the links that cause it. This gives the attacker an opportunity to attack a data flow without triggering any alarms in the link level. As Figure 19 shows, the attacker introduces severe damages to the target data flow while keeping the PRRs of all links above their threshold PRR_T. Therefore, we suggest tagging the link statistics with Graph IDs in the health reports to help the network manager detect the selective jamming attacks. We leave the development of new defense solutions to secure WirelessHart networks based on the above-mentioned insights as our future work.

ACKNOWLEDGMENT

The work of Xia Cheng, Junyang Shi, Mo Sha was partially supported by the NSF through grants CNS-1657275, CNS-2046538, and CNS-2150010. The work of Linke Guo was partially supported by the NSF through grant IIS-1949640 and CNS-2008049.

REFERENCES

[1] X. Cheng, J. Shi, M. Sha, and L. Guo, "Launching Smart Selective Jamming Attacks in WirelessHART Networks," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*. IEEE, 2021, pp. 1–10.

[2] J. Manyika, M. Chui, J. Bughin, R. Dobbs, P. Bisson, and A. Marrs. (2013) Disruptive Technologies: Advances that will Transform Life, Business, and the Global Economy. [Online]. Available: http://www.mckinsey.com/

[3] HART. (2019) HART Communication Protocol and Foundation (Now the FieldComm Group). [Online]. Available: https://www.fieldcommgroup.org/technologies/hart

[4] WirelessHART, "WirelessHART," 2019. [Online]. Available: https://www.fieldcommgroup.org/technologies/wirelesshart

[5] 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," *Proceedings of the IEEE, Special Issue on Industrial Cyber Physical Systems*, vol. 104, no. 5, pp. 1013–1024, May 2016.

[6] Emerson. (2019) Emerson Wireless-technology. [Online]. Available: https://www.emerson.com/en-us/expertise/automation/industrial-internet-things/pervasive-sensing-solutions/wireless-technology

[7] J. Shi and M. Sha, "Parameter Self-Configuration and Self-Adaptation in Industrial Wireless Sensor-Actuator Networks," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*. IEEE, 2019, pp. 658–666.

[8] M. Sha. (2016) Testbed at the State University of New York at Binghamton. [Online]. Available: https://users.cs.fiu.edu/~msha/testbed.htm

[9] WCPS. (2018) Wireless Cyber-Physical Simulator (WCPS). [Online]. Available: http://wsn.cse.wustl.edu/index.php/WCPS:_Wireless_Cyber-Physical_Simulator

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

[11] Raspberry, "Raspberry Pi," 2019. [Online]. Available: https://www.raspberrypi.org/

[12] Wi-Spy, "Wi-Spy USB Spectrum Analyzer," 2020. [Online]. Available: http://www.wi-spy.co.uk/

[13] X. Cheng, J. Shi, and M. Sha, "Cracking the Channel Hopping Sequences in IEEE 802.15.4e-Based Industrial TSCH Networks," in *Internet of Things Design and Implementation (IoTDI)*. New York, NY, USA: ACM, 2019.

[14] ——, "Cracking Channel Hopping Sequences and Graph Routes in Industrial TSCH Networks," *ACM Transactions on Internet Technology*, vol. 20, no. 3, Jul. 2020.

[15] 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

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

[17] C. A. Boano, T. Voigt, C. Noda, K. Römer, and M. Zuniga, "Jamlab: Augmenting sensornet testbeds with realistic and controlled interference generation," in *Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks*. IEEE, 04 2011, pp. 175–186.

[18] W. Xu, W. Trappe, Y. Zhang, and T. Wood, "The Feasibility of Launching and Detecting Jamming Attacks in Wireless Networks," in *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, ser. MobiHoc '05. New York, NY, USA: ACM, 2005, pp. 46–57.

[19] K. Grover, A. Lim, and Q. Yang, "Jamming and Anti-jamming Techniques in Wireless Networks: A Survey," *Int. J. Ad Hoc Ubiquitous Comput.*, vol. 17, no. 4, pp. 197–215, Dec. 2014.

[20] 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, May 2016.

[21] M. Wilhelm, I. Martinovic, J. B. Schmitt, and V. Lenders, "Short Paper: Reactive Jamming in Wireless Networks How Realistic is the Threat?" in *Proceedings of the fourth ACM conference on Wireless network security*, ser. WiSec '11. New York, NY, USA: ACM, 2011, pp. 47–52.

[22] L. Zhang, F. Restuccia, T. Melodia, and S. M. Pudlewski, "Jam Sessions: Analysis and Experimental Evaluation of Advanced Jamming Attacks in MIMO Networks," in *Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. Mobihoc '19, New York, NY, USA, 2019, p. 61–70.

[23] 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, Mar. 2014.

[24] 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, Aug. 2010.

[25] M. K. Hanawal, D. N. Nguyen, and M. Krunz, "Jamming attack on in-band full-duplex communications: Detection and countermeasures," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 2016, pp. 1–9.

[26] 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, Aug. 2014.

[27] Z. Lu, W. Wang, and C. Wang, "From jammer to gambler: Modeling and detection of jamming attacks against time-critical traffic," in *2011 Proceedings IEEE INFOCOM*. Piscataway, NJ, USA: IEEE, April 2011, pp. 1871–1879.

[28] 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, Oct 2016.

[29] 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, July 2014.

[30] L. Zhang, Z. Guan, and T. Melodia, "Cooperative anti-jamming for infrastructure-less wireless networks with stochastic relaying," in *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, 2014, pp. 549–557.

[31] 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 *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019, pp. 1027–1035.

[32] S. D'Oro, E. Ekici, and S. Palazzo, "Rate Maximization under Reactive Jamming Attacks: Poster," in *Proceedings of the 17th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. MobiHoc '16, New York, NY, USA, 2016, p. 367–368.

[33] V. Navda, A. Bohra, S. Ganguly, and D. Rubenstein, "Using Channel Hopping to Increase 802.11 Resilience to Jamming Attacks," in *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*. Piscataway, NJ, USA: IEEE, May 2007, pp. 2526–2530.

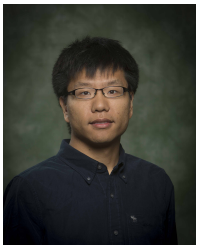
[34] A. Liu, P. Ning, H. Dai, and Y. Liu, "USD-FH: Jamming-resistant wireless communication using Frequency Hopping with Uncoordinated Seed Disclosure," in *The 7th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (IEEE MASS 2010)*. Washington, DC, USA: IEEE, Nov 2010, pp. 41–50.

[35] 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, Jan 2012.
[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 Transactions on Sensor Networks (TOSN)*, vol. 15, no. 1, Feb. 2019.
[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.
[38] H. Pirayesh, P. Kheirkhah Sangdeh, and H. Zeng, "Securing ZigBee Communications Against Constant Jamming Attack Using Neural Network," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4957–4968, 2021.
[39] W. Albazrqaoe, J. Huang, and G. Xing, "Practical Bluetooth Traffic Sniffing: Systems and Privacy Implications," in *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '16, 2016, p. 333–345.
[40] ——, "A Practical Bluetooth Traffic Sniffing System: Design, Implementation, and Countermeasure," *IEEE/ACM Transactions on Networking*, vol. 27, no. 1, pp. 71–84, 2019.
[41] Y.-R. Chien, "Design of GPS Anti-Jamming Systems Using Adaptive Notch Filters," *IEEE Systems Journal*, vol. 9, no. 2, pp. 451–460, 2015.
[42] H. Pirayesh and H. Zeng, "Jamming Attacks and Anti-Jamming Strategies in Wireless Networks: A Comprehensive Survey," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 2, pp. 767–809, 2022.
[43] H. Akhlaghpasand, E. Björnson, and S. M. Razavizadeh, "Jamming-Robust Uplink Transmission for Spatially Correlated Massive MIMO Systems," *IEEE Transactions on Communications*, vol. 68, no. 6, pp. 3495–3504, 2020.
[44] J. Vinogradova, E. Björnson, and E. G. Larsson, "Detection and mitigation of jamming attacks in massive MIMO systems using random matrix theory," in *2016 IEEE 17th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2016, pp. 1–5.
[45] S. Raza, A. Slabbert, T. Voigt, and K. Landernäs, "Security considerations for the WirelessHART protocol," in *Proceedings of the 14th IEEE International Conference on Emerging Technologies and Factory Automation*, ser. ETFA'09. Piscataway, NJ, USA: IEEE, 2009, pp. 242–249.
[46] L. Bayou, D. Espes, N. Cuppens-Boulahia, and F. Cuppens, "Security Issue of WirelessHART Based SCADA Systems," in *Risks and Security of Internet and Systems*. Cham: Springer International Publishing, 07 2015.
[47] ——, "Security Analysis of WirelessHART Communication Scheme," in *Foundations and Practice of Security*, vol. 10128. Cham: Springer International Publishing, 2017, pp. 223–238.
[48] C. Alcaraz and J. Lopez, "A Security Analysis for Wireless Sensor Mesh Networks in Highly Critical Systems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 40, no. 4, pp. 419–428, Jul. 2010.

Xia Cheng is a PhD student in the Knight Foundation School of Computing and Information Sciences at Florida International University. He received a M.Phil. degree from Tsinghua University in 2011 and a B.Eng. degree in Automation Engineering from Tsinghua University in 2006. His research focuses on industrial wireless networks and network security.

Junyang Shi is a software engineer at Google. He received his Ph.D. in Computer Science from State University of New York at Binghamton in 2021 and his B.S. degree in Electrical and Electronic Engineering from the Huazhong University of Science and Technology in 2016. His research focuses on industrial wireless networks and Internet of Things.

Mo Sha is an Associate Professor in the Knight Foundation School of Computing and Information Sciences at Florida International University (FIU). Before joining FIU, he was an Assistant Professor in the Department of Computer Science at State University of New York at Binghamton. His research interests include wireless networking, Internet of Things, applied machine learning, network security, and cyber-physical systems. He published more than 50 research papers, served on the technical program committees of 19 premier conferences, and reviewed paper for 22 journals. He received the NSF CAREER award in 2021, the NSF CRII award in 2017, and the Educator of the Year in Computer Science award and the Career Champion award at Binghamton University in 2018. He received his Ph.D. degree in Computer Science from Washington University in St. Louis in 2014, his M.Phil. degree from City University of Hong Kong in 2009, and his B.Eng. degree from Beihang University in 2007. He is a senior and lifetime member of ACM and a member of Sigma Xi.



Linke Guo is an Associate Professor in Holcombe Department of Electrical and Computer Engineering at Clemson University. He received his Ph.D. degree from University of Florida in 2014. Prior to his PhD, he received a M.S. degree from University of Florida in 2011 and a B.E. degree from Beijing University of Post and Telecommunications in 2008. His research interests include security and privacy in wireless network, Big Data, and Internet of Things.