POSTER: Cracking the TSCH Channel Hopping in IEEE 802.15.4e

Xia Cheng
Department of Computer Science
State University of New York at Binghamton
Binghamton, New York, USA
xcheng12@binghamton.edu

Mo Sha∗
Department of Computer Science
State University of New York at Binghamton
Binghamton, New York, USA
msha@binghamton.edu

ABSTRACT
Industrial networks typically connect hundreds or thousands of sensors and actuators in industrial facilities, such as steel mills and oil refineries. Although the typical industrial applications operate at low data rates, they pose unique challenges because of their critical demands for reliable and real-time communication in harsh industrial environments. IEEE 802.15.4 based Wireless Sensor-Actuator Networks (WSANs) technology is appealing for use to construct industrial networks because it can be deployed and maintained inexpensively. Battery-powered wireless modules easily and inexpensively retrofit existing sensors and actuators in industrial facilities without running cabling for communication and power. To address the stringent real-time and reliability requirements, WSANs adopt a set of novel design choices such as employing the Time-Synchronized Channel Hopping (TSCH) technology that distinguish themselves from traditional Wireless Sensor Networks (WSNs) that require only best effort services. The equation-based channel hopping used in TSCH simplifies the network operations at the cost of security. Our case study shows that an attacker can reverse engineer the channel hopping sequence by silently observing the channel activities and then perform smart collision attacks. In this poster proposal, we describe our target problem and present our case study based on a publicly accessible implementation of TSCH.

KEYWORDS
Wireless Sensor-Actuator Networks; IEEE 802.15.4; Time-Synchronized Channel Hopping; Industrial Networks

ACM Reference Format:

1 INTRODUCTION
Industrial networks, the fundamental support of the industrial Internet of Things (IoT), typically connect hundreds or thousands of sensors and actuators in industrial facilities, such as steel mills, oil refineries, and infrastructures implementing complex monitoring and control processes. Although the typical industrial applications operate at low data rates, they pose unique challenges because of their critical demands for reliable and real-time communication in harsh industrial environments. Failure to achieve such performance can lead to production inefficiency, safety vulnerability, and financial loss. These requirements have been traditionally met by specifically chosen wired solutions, e.g., the Highway Addressable Remote Transducer (HART) communication protocol [5], where cables connect sensors and forward sensor readings to a control room where a controller collects data and sends commands to actuators. However, wired networks are usually costly to deploy and maintain in industrial environments and difficult to update to accompany new production requirements. IEEE 802.15.4 based Wireless Sensor-Actuator Networks (WSANs) technology is appealing for use to construct industrial networks because it does not require wired infrastructure and can be deployed and maintained inexpensively. Battery-powered wireless modules easily and inexpensively retrofit existing sensors and actuators in industrial facilities without running cabling for communication and power.

To address the stringent real-time and reliability requirements, industrial WSANs adopt a set of unique design choices such as employing the Time-Synchronized Channel Hopping (TSCH) technology in the Medium Access Control (MAC) layer that distinguish themselves from traditional Wireless Sensor Networks (WSNs) that require only best effort services [12]. Specifically, TSCH divides time into slots of fixed length that are grouped into a slotframe. Nodes are synchronized and share the notion of single or multiple slotframes that repeat over time. Frequency diversity is used to mitigate effects of multipath fading and to improve the robustness and network capacity. Channel hopping is achieved by sending successive packets on different frequencies. All nodes in the network compute the channel hopping sequence by following an equation. TSCH was amended into the IEEE 802.15.4e standard in 2012 [7] and adopted by the leading industrial WSAN standards (WirelessHART [6] and ISA100 [10]) and the one being standardized by IETF (6TiSCH [9]).

TSCH’s equation-based channel hopping simplifies the network operations at the cost of security. We find that an attacker can reverse engineer the channel hopping sequence by silently observing the channel activities and then perform smart collision attacks to those transmissions of interest. In a smart collision attack, the attacker overhears the channel usage, computes the transmission schedule, and anticipates the key message transmission time. After that, the attacker transmits a packet when a genuine packet is sent, which causes packet collision at the communication channel. We consider the smart collision attacks instead of the simple
jamming attacks, since blindly jamming a channel can be easily detected and localized by a wireless intrusion prevention system [14–16, 18, 19, 21]. Many countermeasures have been developed in the WSNs literature to minimize the damage, such as adjusting routing [1, 11, 20]. However, the existing approaches may fail if the transmission failures caused by the smart collision attacks only happen occasionally and are buried in the normal fluctuations of low-power links. In this poster proposal, we describe our target problem and present our case study based on a publicly accessible implementation of TSCH. The remainder of the poster proposal is organized as follows. Section 2 introduces the background of TSCH channel hopping. Section 3 presents our case study. Section 4 concludes the poster proposal.

2 BACKGROUND ON TSCH CHANNEL HOPPING

To provide time-deterministic packet deliveries and combat multi-path fading and signal interference, TSCH combines time-slotted MAC access, multi-channel communication, and channel hopping. TSCH divides time into slots of fixed length that are grouped in a slotframe. Each time slot is long enough to transfer both a data packet and an acknowledgement between a pair of network nodes. Nodes are synchronized and share the notion of single or multiple slotframes that repeats over time. Channel hopping is achieved by sending successive packets on different frequencies. The channel hopping sequence is fixed and known by all nodes in the network. Initially, at most 16 different channels are available for communication. Each channel is identified by a channel Offset, i.e., an integer value in the range $[0, 15]$. However, some of these frequencies could be blacklisted (because of low-quality communication) and, hence, the total number of channels $N_{channel}$ available for channel hopping may be less than 16. In TSCH, a link is defined as the pairwise assignment of a directed communication between nodes in a given time slot on a given channel offset. Hence, a link between communicating nodes can be represented by specifying the slot in the slotframe and the channel offset used by the nodes in that time slot. Let $[n, channelOffset]$ denote a link between two nodes. Then the communication frequency (channel) to be used for communication in time slot $n$ of the slotframe is derived as

$$f = F((ASN + channelOffset) \mod N_{channels})$$

(1)

where $ASN$ is the Absolute Slot Number, defined as the total number of time slots elapsed since the start of the network, and "\mod" is the modulo operator. The $ASN$ increments globally in the network, at every time slot, and is thus used by nodes as a time slot counter. The function $F$ consists of a look-up table containing the set of available channels. Eq. 1 can also be expressed as:

$$f = F([SN \cdot N + X + channelOffset] \mod N_{channels})$$

(2)

where $SN$ is the number of time slots in a slotframe ($ASN = SN \cdot N + X$), $N$ is the number of slotframes elapsed since the start of the network, and $X$ is the time slot offset in the slotframe.

3 CASE STUDY ON CRACKING ORCHESTRA

In this section, we present the security vulnerability of TSCH channel hopping in IEEE 802.15.4e by demonstrating how an attacker cracks the channel hopping sequence by silently observing the channel activities and predicts the future channel usage for attacks. The attacker is assumed to be a device which is capable of monitoring activities on all 16 channels in 2.4 GHz ISM band and has moderate computational capability (e.g., a Raspberry Pi 3 Model B [4] integrated with a Wi-Spy USB Spectrum Analyzer [13]). We perform the case study using a publicly accessible implementation of TSCH [2], which was developed to support the Orchestra system [3]. We run the experiments on our testbed consisting of 50 TelosB motes deployed in the second floor of the Engineering Building in our campus [17]. Figure 1 shows the testbed deployment.

Orchestra proposes an autonomous transmission scheduling method running on top of RPL [8] and TSCH networks. Each node computes its transmission schedule locally based on its routing state. All nodes running Orchestra hop the channel together according to the TSCH channel hopping method (using Eq. 2 to compute channel hopping sequence). Orchestra deploys three types of slotframes for different kinds of traffic: application, routing, and time synchronization. The number of slots in each type of slotframe is fixed at 47 ($SN_{app}$), 31 ($SN_{r}$), and 397 ($SN_{t}$). For the application and routing slotframes, the nodes are allowed to skip their scheduled slots when no application or routing packet is pending for transmission. The random usage of scheduled slots poses significant challenge on the channel hopping sequence.

We use four channels in the network (by default in Orchestra) and vary the data generation interval to examine our cracking method under different traffic loads. We perform eight sets of experiments with the data generation interval of 1s, 3s, 6s, 10s, 20s, 30s, 45s, and 60s. With each data generation interval, we randomly select a starting time to begin cracking. We next briefly present the basic idea on cracking the key parameters in Eq. 2.

- **Number of channels used in the network ($N_{channel}$)**: According to Eq. 2, all channels used in the network appears during every $SN \cdot N_{channel}$ slots. The time slot length $T$ and the maximum length of a slotframe $SN_{max}$ are usually specified by the upper layer standard (e.g., WirelessHART$^1$).

$^1$Even without specifications, the attacker can easily get $T$ by measuring two consecutive transmissions on different channels and use a large number for $SN_{max}$.
Table 1: Evaluation Result.

<table>
<thead>
<tr>
<th>Data Interval (s)</th>
<th>Cracking Time (s)</th>
<th>Cracking Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>278.79</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>281.55</td>
<td>100</td>
</tr>
<tr>
<td>6</td>
<td>288.42</td>
<td>100</td>
</tr>
<tr>
<td>10</td>
<td>327.74</td>
<td>93.3</td>
</tr>
<tr>
<td>20</td>
<td>303.18</td>
<td>93.3</td>
</tr>
<tr>
<td>30</td>
<td>274.41</td>
<td>93.3</td>
</tr>
<tr>
<td>45</td>
<td>327.74</td>
<td>93.3</td>
</tr>
<tr>
<td>60</td>
<td>351.29</td>
<td>86.7</td>
</tr>
</tbody>
</table>

\( N_{\text{channel}} \leq 16 \). The attacker can obtain \( N_{\text{channel}} \) by counting the channels appeared within the time window \( SN_{\text{max}} + 16 \times T \).

- **SN (number of slots in the slotframe):** According to Eq. 2, the channel must use the same channel in the time slot \( S_m \) and \( S_m+SN_{\text{channel}} \), \( S_m+1 \), and \( S_m+SN_{\text{channel}}+1 \). In other words, the channel hopping sequence used in the network repeats in every \( SN_{\text{channel}} \) time slots. Thus, the attacker can obtain the SN by observing the channel usage repetitions within the time window \( SN_{\text{max}} + 16 \). Please note that if \( SN \) and \( N_{\text{channel}} \) share one or more common factors, the repeating cycle will be the lowest common multiple of these two parameters.

- **Others:** The goal of cracking the channel hopping sequence by an attacker is to predict the future channel usage and then perform collision attacks. Thus there is no need for an attacker to obtain the actual values of \( N \) (the number of slotframes elapsed since the start of the network), \( X \) (the time slot offset in the slotframe), and \( \text{channelOffset} \) (the integer channel offset value). The attacker can assume the time slot \( S_m \) is the first slot in the slotframe and create a table pairing the time slot offset from \( S_m \) and the corresponding channel usage sequence observed.

Table 1 shows the time consumed by our implementation to derive the channel sequence and the prediction accuracy comparing to the ground truth. Our cracking method achieves 100% accuracy in five cases. The accuracy is lower when the data intervals are 20s, 45s, and 60s. In those cases where the data rates are low and the nodes skip many scheduled slots in the routing slotframe, the multiples of 31 are taken as \( SN \) of the routing slotframe.

Because of the fixed length of each slotframe of Orchestra, we are able to recognize the channel hopping features of each slotframe and extract them one by one to do cracking. However, for an unknown network, multiple slotframes increase the difficulty of observing the channel hopping repetition cycle. Moreover, the fact that nodes may or may not transmit packets in their scheduled slots brings in great flexibility, making it much more harder to pinpoint the repetitions of each slotframe. Our future work will focus on cracking the network running with multiple slotframes with unknown slotframe lengths and opportunistic transmissions. We will then explore potential countermeasures according to our observations.

4 CONCLUSIONS

To address the stringent real-time and reliability requirements posed by industrial applications, WSANs made a set of unique design choices such as employing TSCH that distinguish themselves from traditional WSNs that require only best effort services. The equation-based channel hopping in TSCH simplifies the network operations at the cost of security. Our case study shows that an attacker can reverse engineer the channel hopping sequences by silently observing the channel activities and then perform smart collision attacks to those transmissions of interest.

ACKNOWLEDGMENT

This work was supported by the NSF through grant CRII-1657275 (NeTS).

REFERENCES