SlotSwapper: A Schedule Randomization protocol for Real-Time WirelessHART Networks

Ankita Samaddar, Arvind Easwaran, Rui Tan
Nanyang Technological University, Singapore
{ankita003,arvinde,tanrui}@ntu.edu.sg

ABSTRACT

Industrial process control systems are time-critical systems where reliable communications between sensors and actuators need to be guaranteed within strict deadlines to maintain safe operation of all the components of the system. WirelessHART is the most widely adopted standard which serves as the medium of communication in industrial setups due to its support for Time Division Multiple Access (TDMA) based communication, multiple channels, channel hopping, centralized architecture, redundant routes and avoidance of spatial re-use of channels. However, the communication schedule in WirelessHART network is decided by a centralized network manager at the time of network initialization and the same communication schedule repeats every hyper-period. Due to predictability in the time slots of the communication schedule, these systems are vulnerable to timing attacks which eventually can disrupt the safety of the system. In this work, we present a moving target defense mechanism, the SlotSwapper, which uses schedule randomization techniques to randomize the time slots over a hyper-period schedule, while still preserving all the feasibility constraints of a real-time WirelessHART network and makes the schedule uncertain every hyper-period. We tested the feasibility of the generated schedules on random topologies with 100 simulated motes in Cooja simulator. We use schedule entropy to measure the confidentiality of our algorithm in terms of randomness in the time slots of the generated schedules.

Keywords

WirelessHART, schedule, randomization, entropy

1. INTRODUCTION

Time-critical systems such as the industrial process control systems are real-time cyber-physical systems (CPS) that monitor and control the production lines in a manufacturing plant. The number of devices in such setup keeps increasing. To support more devices and to cope up with frequent changes in the network topology due to addition (removal) of devices to (from) the network, a switch of the communication infrastructure from wired networks to wireless networks is desirable. Among the existing wireless sensor network (WSN) standards, WirelessHART is best suited for the industrial process control systems due to its reliable TDMA-based schedule, centralized architecture, multi-channel support, channel hopping, redundancy in routes, and avoidance of spatial re-use of channels.

RTN'19, July 2019, Stuttgart, Germany

Although the use of wireless brings flexibility and adaptability to the communication infrastructure, it increases the threats of cyber attacks. Some recent sophisticated attacks against critical infrastructures such as Stuxnet [1] and Dragonfly [2] have alerted us to the shaky protection of the conventional air gap solution. The main components of a WirelessHART network are the sensors, actuators, Gateway, a network manager, and multiple access points (AP). Each communication between these devices are real-time flows with fixed periods and deadlines. To make the flows schedulable, the schedule in a WirelessHART network is predetermined by the centralized network manager at the time of network initialization. The same schedule is repeated over every hyper-period (*i.e.*, lowest common multiple of the periods of all the flows in the network), until there is any change in the network topology, such as addition/removal of new/existing devices to/from the network. The repetitive execution of the deterministic flow schedule in a WirelessHART network over every hyper-period makes these systems vulnerable to timing attacks. Such repetition greatly helps the attacker to analyze the eavesdropped traces and infer the schedule. With the inferred schedule, the attacker can further launch various strategic destructive attack steps. For instance, the attacker can selectively jam the transmissions from/to a certain critical sensor/actuator which can eventually breach the safety of the system.

In this work, we aim at reducing the predictability of the time slots in the communication schedule of a real-time WirelessHART network. We propose a moving target defense (MTD) mechanism, the *SlotSwapper*, that randomizes the time slots in the communication schedule over every hyper-period, satisfying all the feasibility constraints of a real-time WirelessHART network as follows— (1) deadlines of all real-time flows in the network are to be satisfied, (2) the hop sequences associated with each flow are to be preserved and (3) no conflicting transmissions in the network are allowed. From our analysis, the attacker who can monitor the wireless transmissions needs at least two hyper-periods to infer the schedule. Randomizing the schedule over every hyper-period renders the attacker's inference futile, thereby greatly improving the confidentiality of the WirelessHART network's operations. More varied are the slots in a schedule, more difficult it is for the attacker to predict them. Hence, the measure of uncertainty in the time slots of a schedule can be expressed in terms of the amount of randomness in the time slots over the hyper-period schedules generated by our algorithm. We re-defined *schedule entropy* [3] as a metric to measure the uncertainty in predicting the time slots. We illustrated the feasibility of our proposed algorithm on random topologies with 100 simulated nodes in Contiki Cooja [4]. To the best of our knowledge, this is the first work on randomization to reduce the determinism of the time slots of a hyper-period schedule in real-time WirelessHART networks.

2. RELATED WORK

Two notable works in the literature which adopt randomization techniques in the context of real-time processor scheduling are *taskshuffler* [3] and SPARTA [5]. [3] presents a schedule randomization protocol, the *taskshuffler*, that shuffles a set of fixed priority real-time tasks on a uniprocessor system. [5] proposes SPARTA, a scheduler to randomize the leakage points in the schedule protecting the system from Differential Power Analysis (DPA) attacks. However, both of these works are on uniprocessor system. Our problem is even harder than multi-processor scheduling. m channels and n real-time flows of our network can be mapped to m processors and n real-time tasks respectively. However, the conflicting transmissions among the flows impose additional constraint in our network which makes our problem even harder than multi-processor scheduling.

Due to support for TDMA schedule in WirelessHART networks, these networks are vulnerable to selective jamming attacks [6]. [7,8] survey various possible jamming attacks and the key ideas of existing security mechanisms against such attacks in WSNs. [9] proposes various types of side-channel attacks and their respective countermeasures in WSN. The countermeasures against jamming attacks can be provided from physical-layer solutions as in [7,10] or cyber-space solutions such as [11,12]. [13] presents the steps of an attacker to launch jamming attacks in industrial process control systems. Recent works such as [14] and [15] provide countermeasures against timing attacks in single and multi-channel WSN respectively by permuting the slot utilization pattern at the node level over a super-frame to randomize the schedule. However, the flows considered in these works are not associated with deadlines, hence, randomization of slot utilization pattern at the node level makes the flows schedulable. Our problem is more complex. Each flow in our network is a real-time flow with a strict deadline. Permuting the time-slots at each node does not guarantee deadline satisfaction of all the real-time flows in our network, hence, existing solutions in [14] and [15] are not applicable.

3. WIRELESSHART BACKGROUND

The WirelessHART protocol, being compliant with IEEE 802.15.4, is the first open wireless communication standard for measurement and control in network and process industry [16]. A WirelessHART network consists of a Gateway, multiple field devices, APs and a centralized network manager which are connected via wireless mesh networks. The network manager, connected to the Gateway, is responsible for managing the devices, scheduling, creating the routes and optimizing the network. The field devices are wireless sensors and actuators which can either transmit or receive in a particular time slot. Also, in a time slot, a receiver can receive from exactly one sender. Multiple APs are connected to the Gateway via wired connections to provide redundant paths between the Gateway and the network devices. The key features of the WirelessHART network for which it is suitable for process industries include

TDMA: For reliable collision-free communications in a WirelessHART network, time is globally synchronized and slotted into 10ms time slots within which a network device sends a packet and receives its corresponding acknowledgment.

Channel and route diversity: WirelessHART supports a maximum of 16 channels [17] at a frequency band of 2.4 GHz. To avoid interference from neighboring wireless systems, it adopts channel hopping in every time slot. A channel is blacklisted if it suffers from external interference. WirelessHART allows route diversity by transmitting a packet multiple times via multiple paths over different channels.

Avoidance of spatial re-use of channels: To avoid interference and to increase reliability, WirelessHART avoids spatial re-use of channels [17]. The physical channel assigned to a link in a particular time slot is given by [17], $Ch_p = (ASN + Ch_l) \bmod m$, where ASN represents Absolute Slot Number and increases at every slot, Ch_l and Ch_p are the logical and physical channels assigned to a node, m denotes the number of channels in the network.

A WirelessHART network is represented as a graph $G = (V, E)$, where V is the set of nodes which are the sensors, actuators and Gateway; E is the set of edges or links between the devices. An edge $e = u \to v$, $u, v \in V$, is part of G, if and only if device u can reliably communicate with device v. In a transmission along an edge $u \to v$, the transmitting node, u, is the *sender* and the receiving node, v, is the *receiver* of the transmission.

Definition 1: *Two transmissions along edges $u \to v$ and $w \to x$, where $u, v, w, x \in V$, are said to be **conflicting transmissions**, if both of them have the same sender or the same receiver, i.e., if $(u = w) \lor (v = w) \lor (u = x) \lor (v = x)$. For each edge $u \to v \in E$, there exists a set of conflicting transmissions in G. To keep track of the conflicting transmissions in G, we store an adjacency list known as the **Conflict List**. Each index i in the list corresponds to an edge in E and the list corresponding to i stores the list of edges which generate conflicting transmissions with i.*

An end-to-end communication between a sensor and an actuator occurs in two phases: a *sensing phase* and a *control phase* during which the communications are between the sensors and the Gateway and between the Gateway and the actuators respectively.

4. SYSTEM MODEL

Our system model consists of a WirelessHART network $G = (V, E)$ and n end-to-end flows $\mathcal{F} = \{\mathcal{F}_1, \mathcal{F}_2, \dots \mathcal{F}_n\}$. Each flow $\mathcal{F}_i \in \mathcal{F}$ periodically generates a packet at the source node $s_i \in V$ with period p_i. The packet passes via Gateway and reaches the destination node $d_i \in V \setminus \{s_i\}$ within deadline δ_i. We assume that our flows are of implicit deadline, *i.e.*, $\delta_i \leq p_i$. A packet is scheduled in more than one routes between the source and destination for reliability.

Definition 2: *The **release time** (r_{ij}) of the j^{th} instance of flow $\mathcal{F}_i$ $(j \geq 1)$ is the time at which the j^{th} instance of $\mathcal{F}_i$ is released at the source node s_i. r_{ij} is defined as*

$$r_{ij} = (j - 1) \cdot p_i. \tag{1}$$

Definition 3: *The **number of hops** in a route of a flow $\mathcal{F}_i$ is the number of intermediate devices between the source (s_i) and the destination (d_i) in the route of $\mathcal{F}_i$.*

Definition 4: *Given a graph G with m channels and a set of flows $\mathcal{F}$, a **feasible schedule** $\mathcal{S}$ is a sequence of transmissions over the slots in $\mathcal{S}$ along the edges in G. Each transmission is a mapping of a flow to a channel in a slot satisfying the following conditions:*

1. ***No transmission conflict:*** *Two transmissions along $u \rightarrow v$ and $w \rightarrow x$ can be scheduled in the same time slot t, if $u \rightarrow v$ and $w \rightarrow x$ are non-conflicting transmissions;*

2. ***No collision:*** *If $u \rightarrow v$ uses channel y and $w \rightarrow x$ uses channel z in the same time slot t, then $y \neq z$, $\forall y, z \in [1, m]$;*

3. ***No deadline violation:*** *If a flow $\mathcal{F}_j$, $1 \leq j \leq n$, has h hops, then all the h hops of $\mathcal{F}_j$ are to be scheduled within the deadline δ_j;*

4. ***Flow sequence preservation:*** *If a flow $\mathcal{F}_j$ has h hops, then the k^{th} hop $(1 < k \leq h)$ cannot be scheduled until all the previous $k-1$ hops are scheduled.*

We assume that the network manager blacklists those channels from the network in which the probability of successful transmission is less than a certain threshold [18]. Therefore, the number of packet drops in the network can be neglected. At the time of network initialization, the network manager decides the *schedule* depending on the number of available channels, the topology of the network and available routes for each flow [17], [19]. Given a graph G, a set of n flows $\mathcal{F}$ over G and m channels, the network manager runs any scheduling algorithm $\mathbb{A}$ that generates a schedule $\mathcal{S}$ satisfying all the conditions of Definition 4. The network manager then informs all the network devices about the allocated slots in which they can transmit (receive) messages from specific neighbors. The network devices become active only in those slots in which they can transmit (receive) messages. The same schedule repeats every hyper-period.

5. THREAT MODEL

The main objective of the adversary is to select a critical sensor or an actuator as the victim node in the network and predict the time slots in which the victim node sends (receives) packets to (from) its neighboring nodes by observing the traffic in the network. Our adversary model is based on the following assumptions:-

1. The adversary is aware of the network parameters such as the number of channels adopted by the network.
2. The adversary is equipped with multiple antennae, hence, he is capable of listening to all 16 channels in 2.4 GHz ISM band in the network.

Based on the above assumptions, the adversary has the following capabilities:

Capability 1: The adversary can target a specific node (sensor or actuator) as the victim node in the network and monitor all communications associated with that node. After analyzing the traffic for a sufficiently long period of time, the adversary can predict the time slots in which the victim node communicates with its neighbors.

Capability 2: Due to repetitive nature of the communication schedule, the adversary can estimate the hyper-period of the schedule. The adversary can use this estimate in the subsequent hyper-periods to infer the communication time slots of the victim node.

Capability 3: The adversary can reverse engineer the channel hopping sequences by silently observing the channel activities in the network [20].

With the above three capabilities, the adversary can execute further destructive attack steps. For instance, the adversary can target specific transmissions from (to) certain critical sensors (actuators) and can selectively jam the targeted transmissions in specific time slots, thereby causing disruptive effect on the system. Due to repetitive nature of the hyper-period schedules, same flow gets transmitted in the same time slot over every hyper-period. Hence, selectively jamming the predicted channel in specific time slots over every hyper-period results in jamming the targeted flow with probability 1. Different from the constant jamming attack that jams all the transmissions, selective jamming is more stealthy as it allows the attacker to strategically target certain critical sensors and/or actuators within their proximity with much lower radio transmission power. This reduces the overhead and cost for the attacker to implement the jamming attack [21]. In contrast, random jamming that does not infer the schedule and jams in randomly selected slots is much less effective [22].

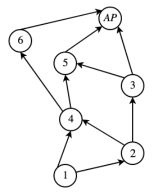

Figure 1: A network graph with six nodes and one AP

		1	2	3	4	5	6	7	8
$\mathcal{S}_1$	ch_1	$1-2$ (F_1)	—	—	$2-3$ (F_3)	$4-5$ (F_2)	—	—	$3-AP$ (F_3)
	ch_2	$4-5$ (F_2)	—	$5-AP$ (F_2)	—	$2-3$ (F_1)	$3-AP$ (F_1)	$5-AP$ (F_2)	—
$\mathcal{S}_2$	ch_1	—	$1-2$ (F_1)	—	$5-AP$ (F_2)	$3-AP$ (F_1)	$4-5$ (F_2)	—	$5-AP$ (F_2)
	ch_2	$2-3$ (F_3)	$4-5$ (F_2)	$2-3$ (F_1)	—	—		$3-AP$ (F_3)	—

Table 1: Two schedules $\mathcal{S}_1$ and $\mathcal{S}_2$ over 8 time slots with three flows $\mathcal{F}_1$, $\mathcal{F}_2$, $\mathcal{F}_3$ where $s_1 = 1$, $s_2 = 4$, $s_3 = 2$ and $d_1 = d_2 = d_3 = AP$.

Attack consequences: Selectively jamming the transmissions from a critical sensor node results in blocking the sensor data to reach the Gateway. As a result, proper control commands cannot be delivered to the actuators which in turn may result in degraded performance of the system. Also, selectively jamming the control commands to reach the actuators may hamper the safety of the system.

Motivation of our work: The main objective of our work is to develop a MTD technique, the *SlotSwapper*, that randomizes the communication time slots over every hyper-period schedule such that the schedule changes before the attacker can estimate it. We present a motivating example to illustrate how the threat can be addressed by randomizing the time slots in every hyper-period schedule.

Example 1: *Consider the network graph shown in Figure 1 with two channels, three flows, F_1, F_2 and F_3 where the sources are $s_1 = 1$, $s_2 = 4$, $s_3 = 2$; the destinations are $d_1 = d_2 = d_3 = AP$; the periods and the dealines are $p_1 = p_3 = \delta_1 = \delta_3 = 8$, $p_2 = \delta_2 = 4$ respectively. Consider $\mathcal{S}_1$ in Table 1 to be the hyper-period schedule over the flows. Consider node 1 to be the victim node. In the traditional TDMA-based real-time WirelessHART network, the network starts with schedule $\mathcal{S}_1$ which repeats every 8 time slots. An attacker listening to the channels in the network will find nodes 1 and 2 communicating every 8 time slots. In particular, to identify this repetitive pattern, the attacker needs to listen to the network for at least two hyper-periods, i.e., 16 time slots. The attacker can launch selective jamming* *attack earliest in the 17th slot. With our proposed MTD technique, a new schedule is followed in each hyper-period,* i.e., *if $\mathcal{S}_1$ is followed in the first eight slots, then $\mathcal{S}_2$ will be followed in the next eight slots and so on. However, there is no communication between nodes 1 and 2 in slot 1 in $\mathcal{S}_2$,* i.e., *the communicating time slots in two consecutive hyper-periods are different. To identify the repetitive patterns in the schedule, the attacker needs to monitor the communications for at least two hyper-periods. Hence, by changing the schedule every hyper-period, the system will change at a faster pace compared to the learning pace of the attacker, rendering further strategic destructive attack steps (e.g., selective jamming) infeasible.*

6. PROPOSED MTD TECHNIQUE

Our proposed MTD technique, the *SlotSwapper*, consists of two main phases— (1) An offline schedule generation phase (2) an online schedule selection phase. *Sched_Gen*() considers an initial hyper-period schedule $\mathcal{B}$ for a set of n flows $\mathcal{F}$ over a graph G, and generates a new feasible schedule $\mathcal{S}'$ by randomizing the slots in $\mathcal{B}$. However, randomization of time slots in $\mathcal{B}$ is to be done in such a way that all the conditions of generating a *feasible schedule* (Definition 4) are obeyed. To reduce the repeatability of time slots in $\mathcal{B}$, we propose to run *Sched_Gen*() K times (K is a large number) in offline mode and generate a set of feasible hyper-period schedules $\mathbb{S}$. We suggest to select a schedule uniformly at random every hyper-period from $\mathbb{S}$ and execute that schedule over that hyper-period.

Algorithm 1: *SlotSwapper*

```
1 $\mathbb{S} = \{\mathcal{B}\}$;// a base scehdule
2 for $i$=1,2 upto $K$ do
3     $\mathbb{S} = \mathbb{S} \cup$ Sched_Gen();
4 $\mathcal{S}$ = Select a random schedule from $\mathbb{S}$ every hyper-period ;
```

Offline Randomized Schedule Generator : Algorithm 2 presents an overview of *Sched_Gen*(). Table 2 summarizes the notations used in the algorithm. We present an example to illustrate the steps of *Sched_Gen*().

G	a network graph over V nodes and $\lvert E\rvert$ edges
$\mathcal{F}$	a set of n flows defined over G
m	number of channels in the network
hp	hyper-period of n flows
$\mathcal{B}$	a base schedule consisting of mapping of a channel in a slot to a flow over one hp
C	Conflict List corresponding to the network graph G
$\mathcal{S}'$	a copy of the base schedule $\mathcal{B}$
hop_list	a dictionary to store hop number to slot mapping of all the flow instances in $\mathcal{F}$
edge_list	a dictionary to map channel to edge in a particular slot in $\mathcal{S}'$.

Table 2: List of notations used in the algorithm.

Example 2: *Consider the same setting as in Figure 1 and Example 1. Let $\mathcal{S}_1$ in Table 1 be the base schedule. Let us consider the 1st hop of $\mathcal{F}_3$ in $\mathcal{S}_1$ with $\sigma_t = 4$ and c_ch $= 1$. The window corresponding to 1st hop of $\mathcal{F}_3$ is $[1, 7]$. For every slot $\sigma'_t \in [1, 7]$ and every channel $ch \in [1, 2]$, we call trConf() and check for conflicting transmission. $2 \rightarrow 3$ has conflicting transmission with $[1 \rightarrow 2, 2 \rightarrow 4, 3 \rightarrow AP]$ in $\mathcal{S}_1$. Therefore, (slot,channel) pairs such as, $(1, 1)$, $(5, 2)$ and $(6, 2)$ are rejected due to transmission conflict with $(4, 1)$. Similarly, (slot,channel) pairs such as $(5, 1)$ and $(7, 2)$ are also rejected by function deadPr() due to violation of deadlines of the flow instances. (slot,channel) pairs $(5, 1)$ and*

Algorithm 2: *Sched_Gen*

```
1 for tick = 1,2, ..., hp do
2     for j = 1,2, ..., |$\mathcal{F}$| do
3         if tick == $\mathcal{F}_j$.deadline then
4             inst = tick/$\mathcal{F}_j$.deadline;
5             for p = 1,2, ...,$\mathcal{F}_j$.n_hops do
6                 $\sigma_t$ = slot of $p^{th}$ hop of inst;
7                 elig_list = {};// empty list
8                 if m == 1 // single-channel
9                 then
10                    lb = inst * $\mathcal{F}_j$.release_time;
11                    ub = inst * $\mathcal{F}_j$.deadline;
12                    for $\sigma'_t$ = lb, lb+1,..., ub do
13                        if $\mathcal{S}'[\sigma'_t] \neq \mathcal{F}_j$ then
14                            Add $\sigma'_t$ to the elig_list;
15                    $\sigma_{random}$ = random(elig_slots);
16                    swap ($\sigma_t, \sigma_{random}$);
17                else
18                    c_ch = channel of $p^{th}$ hop of inst;
19                    if p == 1// first hop
20                    then
21                        lb = inst * $\mathcal{F}_j$.period;
22                    else
23                        lb = slot of $(p-1)^{th}$ hop of inst + 1;
24                    if p == $\mathcal{F}_j$.n_hops // last hop
25                    then
26                        ub = inst * $\mathcal{F}_j$.deadline;
27                    else
28                        ub = slot of $(p+1)^{th}$ hop of inst - 1;
29                    for $\sigma'_t$ = lb, lb+1,..., ub do
30                        for ch = 1,2,...,m do
31                            $b_1 = trConf(\sigma_t, c\_ch, \sigma'_t, ch, C)$;
32                            $b_2 = deadPr(\sigma_t, c\_ch, \sigma'_t, ch)$;
33                            $b_3 = flowPr(\sigma_t, c\_ch, \sigma'_t, ch)$;
34                            if $b_1$ && $b_2$ && $b3$ == 1 then
35                                Add $(\sigma'_t, ch)$ to elig_list;
36                    $(\sigma, c)$ = random(elig_list);
37                    swap($\sigma_t$, c_ch, $\sigma$, c);
38                update hop_list, edge_list and $\mathcal{S}'$;
39 return $\mathcal{S}'$;
```

$(7, 2)$ correspond to the second instance of $\mathcal{F}_2$ with release time at 5th slot and deadline at 8th slot. Hence, the second instance of $\mathcal{F}_2$ cannot be swapped with any other slot before slot 5 or after slot 8. Similarly, flowPr() does not allow (slot,channel) pairs $(1, 2)$, $(6, 2)$ and $(7, 2)$ in the eligible list in order to preserve the hop sequences of flows. If the transmission corresponding to 1st hop of 1st instance of $\mathcal{F}_2$ (via edge $4 \rightarrow 5$) of (slot,channel) pair $(1, 2)$ is allowed to swap with $(4, 1)$, then the second hop of that instance of $\mathcal{F}_2$ would have been scheduled before the first hop, violating the hop sequences of the flow instances. Finally, the list of eligible (slot,channel) pairs are — $[(2, 1), (2, 2), (3, 1), (3, 2), (4, 2), (6, 1), (7, 1)]$. Let $(3, 2)$ be the randomly selected element. Swapping the transmissions and the flow instances between $(3, 2)$ and $(4, 1)$ and iterating the same procedure over all the flow instances generates a completely new feasible schedule.

Online Selection of Schedules: On executing *Sched_Gen*() K times in offline mode, we get a set of feasible schedules $\mathbb{S}$. At the time of network initialization, each node is informed about the time-slots in which it can send/receive messages in each of these K hyper-period schedules. The online schedule selector runs at each node once in every hyper-period, selects a schedule $\mathcal{S}$ from $\mathbb{S}$ uniformly at random and executes $\mathcal{S}$ over that hyper-period. To ensure that the same schedule is selected at each node, we propose to use a pseudo-random number generator (PRNG) [23] (assumed to be secure) initialized with the same seed at each node. This allows each node to select the same schedule every hyper-period without any additional communication.

7. MEASURE OF UNCERTAINTY

Given a set of schedules $\mathbb{S}$ generated by *Sched_Gen*(), we need to quantify the amount of uncertainty in the schedules in $\mathbb{S}$. In [3], *schedule entropy* is used to measure the uncertainty of a given schedule for a uniprocessor system. We have redefined *schedule entropy* as a function of the slot and channel entropy to measure the randomness in the schedules in $\mathbb{S}$. In a multi-channel WirelessHART network, each of the slots σ_i in a schedule $\mathbb{S}$ consists of m channels which can be represented as $\sigma_i = \{c_{i1}, c_{i2}, \ldots, c_{im}\}$. Given a hyper-period schedule $\mathcal{S}$ over l slots and m channels for a set of flows $\mathcal{F}$, the occurrence of the j^{th} flow $\mathcal{F}_j$ in the k^{th} channel of i^{th} slot is a discrete random variable with possible outcomes from 0 to n, where 0 represents idle flow, n is the total number of flows in $\mathcal{F}$. Let $c_{ik} = j$ denotes the j^{th} flow occurring in the k^{th} channel of i^{th} slot of $\mathcal{S}$. However, the occurrence of the j^{th} flow in the k^{th} channel of the i^{th} slot restricts the occurrence of some other flow $\mathcal{F}'_j$ in the same channel of the same slot. Also, if a flow $\mathcal{F}_j$ completes its hops in the i^{th} slot in the schedule, it cannot occur in the subsequent slots until the arrival of its next instance. We therefore, define *Schedule entropy* as

Definition 5: ***Schedule entropy*** *over a set of flows $\mathcal{F}$ for a WirelessHART network with m channels is the conditional entropy of $\mathcal{F}_j$ occurring in the k^{th} channel of the i^{th} slot, given the entropy of all the slots from 1 to $i-1$. It is represented as*

$$H(\mathcal{S}) = \sum_{i=1}^{l} H(\sigma_i|\sigma_1, \sigma_2, \ldots, \sigma_{i-1}) \tag{2}$$

$$H(\sigma_i) = -\sum_{c_{i1}=0}^{n}\sum_{c_{i2}=0}^{n}\ldots\sum_{c_{im}=0}^{n} \Pr(c_{i1}, c_{i2}, \ldots, c_{im}) \log_2 \Pr(c_{i1}, c_{i2}, \ldots, c_{im}) \tag{3}$$

For a multi-channel WirelessHART network with n flows ($n > 16$), the number of possible permutations in the calculation of the joint probability for each slot is exponential. Hence, we consider the empirical probability distribution of the flows across all the channels in each slot which is an upper-approximated value of *slot entropy* as the joint probability is always less than or equal to the sum of individual probabilities [24]. Further, calculation of conditional entropy in Equation (2) involves joint probability distribution of slots in $\mathcal{S}$, which is exponential in nature. So, we consider the empirical probability distribution of the slots in $\mathcal{S}$.

Definition 6: ***Upper-approximated slot entropy*** $\widetilde{H(\sigma_i)}$ *and* ***Upper-approximated schedule entropy*** $\widetilde{H(\mathcal{S})}$ *are defined respectively as follows*

$$\widetilde{H(\sigma_i)} = -\sum_{k=1}^{m}\sum_{j=0}^{n} \Pr(c_{ik} = j) \log_2 \Pr(c_{ik} = j) \tag{4}$$

$$\widetilde{H(\mathcal{S})} = \sum_{i=1}^{l} \widetilde{H(\sigma_i)}. \tag{5}$$

where $\Pr(c_{ik} = j)$ is the probability mass function of the j^{th} flow occurring in the k^{th} channel of the i^{th} slot.

8. EVALUATION

Simulation setup: We use Cooja simulator [4] of Contiki 3.0 to test the feasibility of our schedules. We generated three random topologies with 100 simulated Tmote Sky motes by varying the degree of nodes (θ) or the number of incoming and outgoing edges incident on a node — (1) Graph A (θ between 2 to 4) (2) Graph B (θ between 3 to 6) (3) Graph C (θ between 3 to 8). More the degree of a node, more are the chances of conflicting transmissions and less is the number of available flows for a particular time-slot. Nodes with highest number of neighbors are considered to be the APs.

Flow Generation: A fraction (α) percent of the nodes are randomly selected as the source and destination nodes. The source and destination nodes are disjoint. In our experiments we varied α between 20-80%. We selected the number of hops of each flow to be between 2 to 8 [25] and considered the shortest path as the primary path. The flows have implicit-deadline with periods varying randomly in the range of 2^7 to 2^{10}.

Experiments: We fixed the hyper-period at 2^{10} time slots and ran experiments upto 10000 hyper-periods with the number of flows and the number of channels varying between 10 to 40 and 1 to 4 respectively. For each condition, we generated 100 random instances and measured the upper approximated schedule entropy ($\widetilde{H(\mathcal{S})}$) for each of these instances. Figure 2 shows $\widetilde{H(\mathcal{S})}$ for all the tested scenarios. It has been observed that $\widetilde{H(\mathcal{S})}$ is maximum for single-channel WirelessHART network for all three graphs. This is because in single-channel WirelessHART networks, there is no conflicting transmissions among the flows in the network. As a result, a flow can be scheduled at any slot within its release time and deadline. For a fixed number of channels, $\widetilde{H(\mathcal{S})}$ increases significantly with increase in the number of flows upto 30. After that, there is no significant increase in the value of $\widetilde{H(\mathcal{S})}$ with increase in the number of flows. This is because, with increase in the number of flows more flows can appear in a slot. However, as the number of flows increase, the number of conflicting transmissions among the flows increase which in turn restricts the number of available flows to be scheduled in a particular slot. $\widetilde{H(\mathcal{S})}$ also increases with increase in the number of channels between 2 to 4, as the number of available positions for a flow to be scheduled get increased. However, it has been observed that with increase in the number of channels, the increase in $\widetilde{H(\mathcal{S})}$ is significantly less for Graph C. Among all the three graphs, the number of edges is maximum in Graph C resulting in more conflicting transmissions among the flows thereby restricting the number of available positions to schedule a flow.

Although we ran our algorithm upto 10000 hyper-periods to measure the randomness in the generated schedules, the amount of memory available to each Tmote sky mote is not sufficiently large to store large number of schedules. We measured that each mote can only support a maximum of 2000 time slot information. We observed that, if a node is in the path of all the 40 flows, then it requires to store at-least 80 time slot information per schedule (40 for transmissions and 40 for re-transmissions). With this specification, we were able to store 25 schedules in each node. We can man-

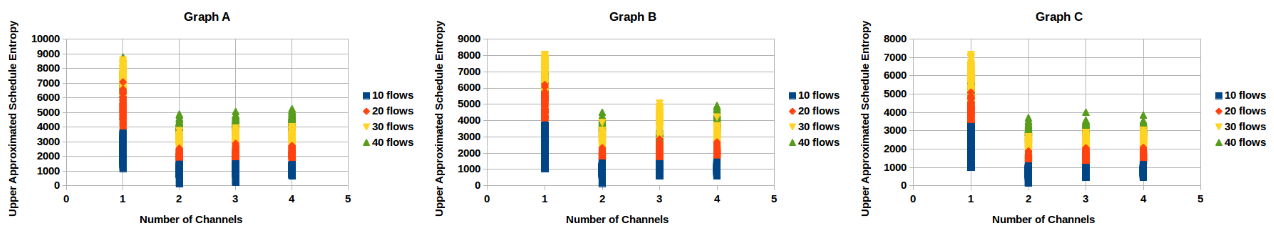

Figure 2: Upper Approximated Schedule Entropy over Graph A,Graph B and Graph C, with number of flows varying between 10 to 40 and number of channels between 1 to 4 with a hyper-period of 1024 time slots

ually tune the nodes with different sets of schedules after several hyper-periods to further reduce the chance of predicting the schedules. Our MTD technique only involves an additional random number generation in each node once in every hyper-period, the power consumption of which is negligibly small.

9. CONCLUSION

In this work, we presented an MTD mechanism, the *SlotSwapper*, to reduce the predictability of TDMA slots in a real-time WirelessHART network. We used schedule entropy to measure the uncertainty of the schedules generated by our algorithm. We illustrated the feasibility of the schedules on simulated networks in Cooja with 100 Tmote sky motes.

10. ACKNOWLEDGEMENT

This work was conducted within the Delta-NTU Corporate Lab for Cyber-Physical Systems with funding support from Delta Electronics Inc. and the National Research Foundation (NRF) Singapore under the Corp Lab@University Scheme.

11. REFERENCES

[1] Ralph Langner. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security & Privacy*, 2011.

[2] Dragonfly: Western energy sector targeted by sophisticated attack group, 2017. https://symc.ly/2Df3VTi.

[3] Man-Ki Yoon, Sibin Mohan, Chien-Ying Chen, and Lui Sha. Taskshuffler: A schedule randomization protocol for obfuscation against timing inference attacks in real-time systems. In *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2016 IEEE*. IEEE.

[4] Fredrik Osterlind, Adam Dunkels, Joakim Eriksson, Niclas Finne, and Thiemo Voigt. Cross-level sensor network simulation with cooja. In *Local computer networks, proceedings 2006 31st IEEE conference on*. IEEE.

[5] Ke Jiang, Petru Eles, Zebo Peng, Sudipta Chattopadhyay, and Lejla Batina. Sparta: A scheduling policy for thwarting differential power analysis attacks. In *Design Automation Conference (ASP-DAC), 2016 21st Asia and South Pacific*. IEEE.

[6] Alejandro Proano and Loukas Lazos. Selective jamming attacks in wireless networks. In *Communications (ICC), 2010 IEEE International Conference on*. IEEE.

[7] Aristides Mpitziopoulos, Damianos Gavalas, Charalampos Konstantopoulos, and Grammati Pantziou. A survey on jamming attacks and countermeasures in wsns. *IEEE Communications Surveys & Tutorials*, 2009.

[8] Deepali Virmani, Ankita Soni, Shringarica Chandel, and Manas Hemrajani. Routing attacks in wireless sensor networks: A survey. *arXiv preprint arXiv:1407.3987*, 2014.

[9] Kanthakumar Pongaliur, Zubin Abraham, Alex X Liu, Li Xiao, and Leo Kempel. Securing sensor nodes against side channel attacks. In *High Assurance Systems Engineering Symposium, 2008. HASE 2008. 11th IEEE*. IEEE.

[10] Raymond Pickholtz, Donald Schilling, and Laurence Milstein. Theory of spread-spectrum communications–a tutorial. *IEEE transactions on Communications*, 1982.

[11] Alejandro Proano and Loukas Lazos. Packet-hiding methods for preventing selective jamming attacks. *IEEE Transactions on dependable and secure computing*, 2012.

[12] Anthony D Wood, John A Stankovic, and Gang Zhou. Deejam: Defeating energy-efficient jamming in ieee 802.15. 4-based wireless networks. In *Sensor, Mesh and Ad Hoc Communications and Networks, 2007. SECON'07. 4th Annual IEEE Communications Society Conference on*. IEEE.

[13] Spase Stojanovski and Andrea Kulakov. Efficient attacks in industrial wireless sensor networks. In *ICT Innovations 2014*. Springer.

[14] Marco Tiloca, Domenico De Guglielmo, Gianluca Dini, Giuseppe Anastasi, and Sajal K Das. Jammy: a distributed and dynamic solution to selective jamming attack in tdma wsns. *IEEE Transactions on Dependable and Secure Computing*, 2017.

[15] Marco Tiloca, Domenico De Guglielmo, Gianluca Dini, Giuseppe Anastasi, and Sajal K Das. Dish: Distributed shuffling against selective jamming attack in ieee 802.15. 4e tsch networks. *ACM Transactions on Sensor Networks (TOSN)*, 2018.

[16] Chenyang Lu, Abusayeed Saifullah, Bo Li, Mo Sha, Humberto Gonzalez, Dolvara Gunatilaka, Chengjie Wu, Lanshun Nie, and Yixin Chen. Real-time wireless sensor-actuator networks for industrial cyber-physical systems. *Proceedings of the IEEE*, 2016.

[17] Deji Chen, Mark Nixon, and Aloysius Mok. *WirelessHART: Real-Time Mesh Network for Industrial Automation*. Springer Publishing Company, Incorporated, 2010.

[18] Jianping Song, Song Han, Al Mok, Deji Chen, Mike Lucas, Mark Nixon, and Wally Pratt. Wirelesshart: Applying wireless technology in real-time industrial process control. In *IEEE real-time and embedded technology and applications symposium*. IEEE, 2008.

[19] Jianping Song, Song Han, Xiuming Zhu, Aloysius K Mok, Deji Chen, and Mark Nixon. A complete wirelesshart network. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*. ACM, 2008.

[20] Xia Cheng, Junyang Shi, and Mo Sha. Cracking the channel hopping sequences in ieee 802.15.4e-based industrial tsch networks. 2019.

[21] Kanika Grover, Alvin Lim, and Qing Yang. Jamming and anti-jamming techniques in wireless networks: a survey. *International Journal of Ad Hoc and Ubiquitous Computing*, 2014.

[22] Wenyuan Xu, Wade Trappe, Yanyong Zhang, and Timothy 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*. ACM, 2005.

[23] Wikipedia contributors. Pseudorandom number generator — Wikipedia, the free encyclopedia, 2019.

[24] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE mobile computing and communications review*, 2001.

[25] Rajeev Alur, Alessandro D'Innocenzo, Karl H Johansson, George J Pappas, and Gera Weiss. Modeling and analysis of multi-hop control networks. In *Real-Time and Embedded Technology and Applications Symposium, 2009. RTAS 2009. 15th IEEE*. IEEE, 2009.