Determining Tolerable Attack Surfaces that Preserves Safety of Cyber-Physical Systems

Carmen Cheh*, Ahmed Fawaz†, Mohammad A. Noureddine*, Binbin Chen‡, William G. Temple‡ and William H. Sanders§
*Department of Computer Science, †Information Trust Institute, §Department of Electrical and Computer Engineering
University of Illinois, Urbana, Illinois, USA
‡Advanced Digital Sciences Center, Singapore
Email: cheh2@illinois.edu, afawaz2@illinois.edu, nouredd2@illinois.edu, whs@illinois.edu,
binbin.chen@adsc-create.edu.sg, william.t@adsc-create.edu.sg

***Abstract*—As safety-critical systems become increasingly interconnected, a system's operations depend on the reliability and security of the computing components and the interconnections among them. Therefore, a growing body of research seeks to tie safety analysis to security analysis. Specifically, it is important to analyze system safety under different attacker models. In this paper, we develop generic parameterizable state automaton templates to model the effects of an attack. Then, given an attacker model, we generate a state automaton that represents the system operation under the threat of the attacker model. We use a railway signaling system as our case study and consider threats to the communication protocol and the commands issued to physical devices. Our results show that while less skilled attackers are not able to violate system safety, more dedicated and skilled attackers can affect system safety. We also consider several countermeasures and show how well they can deter attacks.**

***Index Terms*—cyber-physical system, safety analysis, formal verification, attacker model, timed automata**

I. Introduction

Cyber-physical systems involve the monitoring and control of physical processes and devices through the interactions among computing elements over a network. Many cyber-physical systems have an impact on human life, especially critical infrastructure systems such as the smart grid and transportation systems. For that reason, much effort has gone into ensuring that such cyber-physical systems are *safe*. *Safety* of a system means that the system will never enter a bad state that adversely affects people and the environment. For example, *safety* in the context of a railway system refers to the absence of collisions and derailments.

In practice, a safety analysis is performed to ensure that a system is safe despite component faults and non-malicious human or system errors. The safety analysis is typically conducted using approaches such as hazard analysis, failure mode and effect analysis, formal verification, and fault tree analysis [1]. However, as the control of physical processes is increasingly conducted remotely over communication links, the safety of a system is dependent not just on the reliability of the system components and their interactions but also on the security of the messages sent over the network. It is also necessary to consider the system security at each architectural layer because of advances in cyber attack strategies. Many researchers, domain experts, and government bodies have thus identified the need for cyber security in safety-critical systems [2], [3] and have proposed different approaches for merging safety analysis and security analysis [4]–[6].

Safety-critical systems can be targeted by a variety of malicious actors with differing levels of skill and system access, like nation-state attackers and hacktivists. As seen in the Stuxnet incident in 2010 [7], it is hard to guarantee a system's safety when it is under the threat of a nation-state attacker willing to dedicate time, energy, and resources to infiltrate the system. While it may not be possible to assure complete safety of a system, it is important to identify the necessary qualifications a malicious actor needs in order to carry out an attack that violates system safety. A safety case can be built using that knowledge as supporting evidence [8], and a security practitioner can focus efforts on strengthening the system against such attacks.

In this paper, we analyze the safety of a railway system relative to various types of malicious actors. We consider several classes of attackers with respect to their capabilities and system access, motivated by the existing literature on the feasibility of various attacks. Based on our collaboration with railway industry partners, we constructed a hybrid automata model of a railway system by using UPPAAL [9], a tool for modeling and verifying real-time systems. For each attack capability, we constructed a generic model pattern that represents the effects of an attack. This model pattern can be appended to the appropriate components in the system state automaton. We then generated a state automaton that describes both the normal workings of a system and the actions of an attacker on that system. Finally, using statistical model checking techniques, we assessed the safety of the system

This material is based upon work supported by the Maryland Procurement Office under Contract No. H98230-18-D-0007. This work is also supported in part by the National Research Foundation (NRF), Prime Minister's Office, Singapore, under its National Cybersecurity R&D Programme (Award No. NRF2014NCR-NCR001-31) and administered by the National Cybersecurity R&D Directorate. This research is also supported by the National Research Foundation, Prime Ministers Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme. Part of this work was done when Carmen Cheh was a research intern at ADSC. We also want to thank our rail industry collaborators for providing us with data and domain knowledge.

under various input configurations of the trains in the system.

Our contributions in this paper are thus as follows:

- We describe how we conducted a safety analysis of a railway system subjected to a variety of malicious attacks. We modeled the system and the attacks by using state automata.
- We constructed a framework for generating automata components in UPPAAL that represent the effects of an attack on a system. This allows us to generate a state automaton that can be used for analysis of different attacker models.
- For each set of attacks, we verify the system's safety by using statistical model checking techniques. We inspect those attacks that induce a violation of safety to draw insights into the system components that are more vulnerable to attack and the possible countermeasures.

The structure of this paper is as follows. We present the related literature in Section II. We describe the railway system and the threat model in Section III and model the system and attacker capabilities in Section IV. Then, in Section V, we verify the system safety and present the results. Finally, we discuss future work in Section VI and conclude in Section VII.

II. Related Work

For many years, the well-grounded theory of formal methods has been used to prove the safety of industrial systems [10]. One of the most well-studied and successful applications of formal methods is on verifying the safety of railway signaling systems [11]. With the rapid evolution of the traditionally human-operated railway systems into a more automated communications-based system, the target of the formal verification techniques has also changed, so that it now focuses on proving the correctness of software and application data [12]–[14]. Since the communication network also plays an integral role in the system, work has also been done to verify that service availability is not affected by network degradation or failure [15].

With the rise in cyber attacks on safety-critical systems, more researchers have investigated the safety of railway systems under communication and command errors. Chen et al. [16] conducted a security analysis of railway systems using methods from both the safety and security engineering domains, and show the need for a cyber-physical perspective to study the complicated physical consequences of cyber breaches. In [17], Ghosh et al. modeled the physical train movement and communication with the servers in the network. The safety property was specified as maintaining a safe distance between two trains. The authors found counterexamples of the system's being in an unsafe state when there were errors in the application data or in the movement authority that determines the maximum distance the train is allowed to move. In [18], the authors focused on threats to network communication. They simulated anomalies that included unknown information from track sensors, unexpected train data, and unauthorized command messages. In [19], the failure logs of a real accident were used in failure analysis to discover the source of the accident, which was traced to communication failures. Finally, Cappart et al. [20] used statistical model checking to verify that application data were accurate and that trains would not collide when there were potential errors in application data and in locking points.

However, those approaches consider only a few specific attacks (e.g., an inaccurate movement authority) and do not provide a comprehensive understanding of safety when the system is subjected to a variety of attacks. Bloomfield et al. [21] presented a risk assessment of a national railway system and considered the impact of various attacker models on the system design. The authors used a fault tree analysis to conduct their risk assessment based on expert knowledge and analysis. However, this approach may miss many potential combinations of attack vectors that affect system safety. Thus, a more automated and formal approach is needed to investigate the space of potential attacks.

Rocchetto and Tippenhauer [22] used the CL-Atse tool to find counterexamples (i.e., potential attacks) on a real-world water treatment testbed, given a Dolev-Yao attacker model [23] that they extended to include physical-layer interactions with the system [24]. However, their approach does not take into account network topology and assumes a single network channel. Puys et al. [25] considered both the Dolev-Yao attacker model and three other attacker models that each had a subset of the capabilities of the Dolev-Yao attacker. However, they did not consider physical-layer interactions in the attacker model. The authors also considered the network architecture and the position of the attacker within the system.

Since both of those approaches consider a generic attacker, the state space of the model is extremely large and, given a more complicated system, would result in state space explosion. In addition, the counterexamples that are found from such a model may not give an intuitive insight into the circumstances under which attack capabilities translate into an unsafe situation. In our work, we present a more parameterizable attacker model that allows the modeler to define the situations in which an attack is executed. We also consider physical-layer interactions and the attacker's position.

III. Railway Transportation Systems

Over the years, railway systems have started to integrate cyber technologies and infrastructure to improve system efficiency with regard to route capacity, that is, the number of trains that can be driven on the rail lines while maintaining safety conditions. The safety condition is measured by the acceptable distance between trains, or *headway*.

Previously, the train signaling system was a fixed-block system wherein train locations were determined by old track circuits located at fixed intervals on the rail tracks. Now, the train signaling system has evolved into a moving block system whereby trains constantly relay their locations to a controller via a networked system, i.e., a *Communication-Based Train Control* (CBTC) system. This increased precision in identifying the location of trains allows the system to enforce a shorter headway between trains.

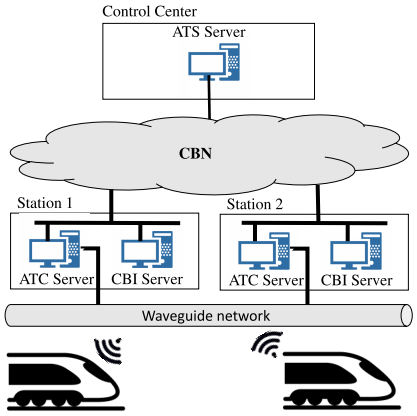


Fig. 1. The architecture of the signaling components of the railway system.

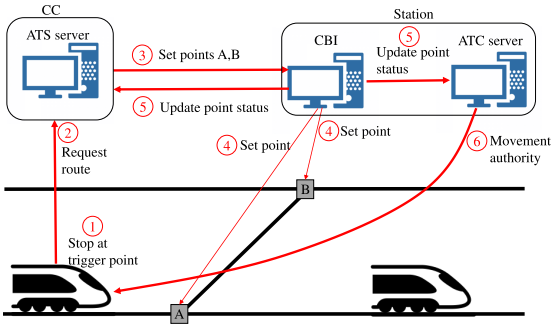


Fig. 2. The protocol underlying the movement of trains as they approach a branch in the tracks. The numbers in circles indicate the sequence of messages sent, and the black undirected lines represent the tracks.

Although the quality of system service increases as a result of the CBTC system, the dependence on cyber infrastructure also increases the system's susceptibility to attacks. The impact of such attacks can be very severe, ranging from service delays to derailment. For example, a Polish teenager once rewired a remote control to communicate with a wireless switch junction, causing derailment of a train and injury of twelve people [26]. It is thus important to understand the potential extent of malicious actors' impact on railway system safety.

In this paper, we focus on the signaling components of the railway system, because they directly affect the speed of the trains and the routes that they take. While the general workings of the signaling systems are the same across all railway systems, railway companies differ in the exact communication protocols used and divisions of computation among the architectural components. We modeled our signaling system based on information we obtained via collaboration with our railway system partner. Although our results are based on this model, they are easily generalizable to other signaling systems.

A. Signaling System

The signaling subsystem consists of distributed components in three different locations: the *Control Center* (CC), the station, and the trackside, as shown in Figure 1. The *Automatic Train Supervision* (ATS) server in the CC manages the train schedules and routes, while the *Automatic Train Control* (ATC) server controls the safe distance that trains are allowed to travel. The *Computer-based Interlocking* (CBI) server manages the point machines that operate mechanical switches (i.e., points). Points are located at track branches and guide the train from one track to another.

Permanently fixed beacons placed at regular intervals across the length of the track transmit absolute positions to the trainborne system on the train. The train regularly transmits its location in a train status update message to the ATC and ATS server. Based on the information collected about the current system state (i.e., train locations, points' statuses), the ATC server generates a *movement authority* (MA) for each train that describes the maximum distance the train can travel before encountering an obstacle. The train then adjusts its speed based on the received MA. Communication between the trackside equipment and train and the station servers is through a wireless waveguide system placed along the entire stretch of the track. Communication among the servers is through a wired backbone network that is isolated from the Internet.

When a train reaches a branch in the tracks, it halts and sends a message to the ATS server to request a route, as shown in Figure 2. The ATS server checks the train's schedule and, based on the embedded route tables in its software, will choose the appropriate route and inform the CBI server to lock the appropriate points. Once the CBI server has locked the points, it informs the ATC and ATS server of the change in point status. The route is then confirmed, and the ATS server reserves the set of track segments for the train. The ATC server sends the new MA to the train, allowing the train to move to the end of the reserved track segments.

B. Threat Model

We assume that the attacker does not have direct physical access to the trainborne system or the trackside devices, because those components are located in places that are hard to access. For example, accessing the trainborne system involves opening up the car exoskeleton, and accessing trackside devices involves entering an underground tunnel.

We describe different levels of capability an attacker can possess, while providing examples of how the attacker could have gained access to such a capability, as shown in Table I. Since railway systems around the world vary in their implementations of security mechanisms, we take into consideration both the case in which a certain security mechanism is in place, and the case in which it is not.

The attacker with the lowest capability level is the outsider who has the least access to the railway system. However, the outsider can access the system as a passenger, and thus can interact with the communication between the trainborne system and the waveguide network. In particular, the attacker can jam communications between the trainborne system and the rest of the network, as shown in several research papers [27], [28].

Even if current modern authentication and integrity protection levels are provided for the network messages, a skilled attacker can also spoof messages to and from the train, given enough time to capture and analyze transmitted messages

TABLE I

THE ATTACKER MODELS DESCRIBED IN TERMS OF THE POSITION THEY OCCUPY IN THE SYSTEM AND THE POTENTIALLY AFFECTED MESSAGES OR COMMANDS. THE NUMBERS IN THE "PROTOCOL COMPONENT" COLUMN REFER TO THE NUMBERED MESSAGES IN FIGURE 2, WHERE 0 REPRESENTS THE TRAIN STATUS UPDATE.

Position in System	Protocol Component Affected	Example of Gaining Position
Train	0, 2, 6	Ride the train
ATC Server	0, 5, 6	Insider, malware
CBI Server	3, 4, 5	Insider, malware
Station Network	0, 2, 3, 5, 6	Insider, malware
ATS Server/CC Network	0, 2, 3	Insider, malware

between the train and network [29], [30]. Depending on the networked system and authentication of endpoints, an attacker could also set up his or her own access points and perform *man-in-the-middle* (MITM) attacks.

The next attacker we consider is the insider who is a legitimate employee in the railway system company. The insider can have physical and/or cyber access to servers and networking equipment. Although insiders may not have all the prerequisite skills for maliciously manipulating the system, they may be in league with a skilled outsider party who can tailor more sophisticated attacks to be delivered to the system with the help of the insider's access. They can then control the physical devices in the system by, for example, sending commands to the point machines by accessing the CBI server either directly or via injection of malware through USB ports. (We assume that such malware can tamper with messages without altering the logical behavior of the server's programs.) Depending on the location of the insider within the system architecture (e.g., control center, station, or a server), an attacker can also manipulate the communication messages that pass through his or her stronghold. More precisely, the attacker can remove, insert, modify, or delay those network packets, much like a Dolev-Yao attacker. The insider has a much higher success rate in attacking the communication messages than the outsider does.

IV. MODELING ATTACKS AND SYSTEM OPERATION

Our approach to verifying the safety of the railway system is shown in Figure 3. The first step is to model the normal system operation as a state automaton, and define the model templates for attacks. Then, we insert those attack templates into the system automaton to create a parametrizable state automaton. Next, we generate the full state automaton with the given parameters and use UPPAAL's model checker to verify that the state automaton satisfies the safety property. In this section, we describe the first step of modeling normal system operation and defining model templates.

There are many junctions in a railway line that are utilized for different purposes, like moving trains to and from the depot or isolating and redirecting trains during emergencies. In this paper, we model a diamond junction at an end station as shown in Figure 4. We chose this junction because the four points at this junction are used the most frequently in the railway line. The points are used to redirect incoming trains to the

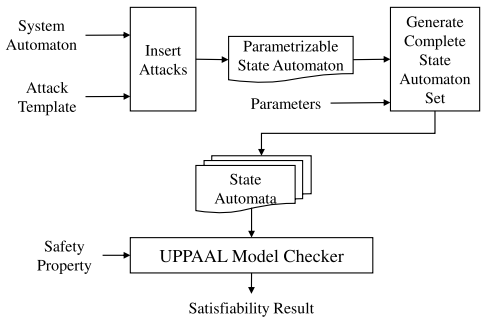


Fig. 3. The general workflow of our approach.

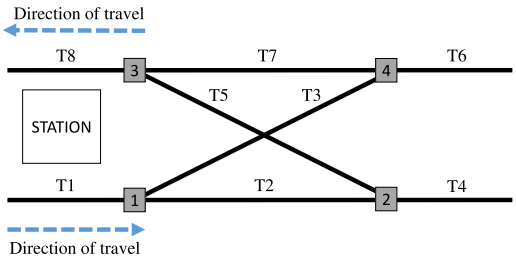


Fig. 4. The diamond junction at an end station. Boxes on the track represent the points and are numbered. The track segments are numbered T1 to T8. Trains travel in the direction indicated by the dotted arrows.

opposite outgoing track. A large volume of passengers also passes through this junction. For that reason, it is a prime target for attackers who wish to cause collisions or derailment. There are four paths that a train can take in a diamond junction. Trains coming into the station move from track T6 to T7 and then T8, or from T6 to T3 and then T1. Trains moving out of the station move from T8 to T5 and then T4, or from T1 to T2 to T4.

We use UPPAAL to model and verify the railway signaling system. The tool allows us to specify networks of timed automata and verify properties, which are specified using temporal logic, on the automata. We model the trains' movement by using differential equations, an approach that is supported by the *Statistical Model Checking* (SMC) extension of UPPAAL [31].

Definition 1. *We use $B(V)$ to denote a Boolean constraint over a set of symbols (or variables) V. A timed automaton is a tuple (V, C, L, l_0, I, A, E) where*

- *V is a set of variables,*
- *C is a set of clocks,*
- *L is a set of locations (or states),*
- *$l_0 \in L$ is the initial location,*
- *$I : L \to B(C \cup V)$ maps locations to invariants over clocks and variables,*
- *$A : V \times 2^C \to V \times 2^C$ is a set of actions, and*
- *$E \subseteq L \times A \times B(C \cup V) \times L$ is a set of transitions.*

Timed automata model the progression of time, and networks of such timed automata model concurrent processes. So timed automata are very suitable for modeling network communications and physical processes. The system moves from one state to another by taking transitions. A transition

$(l, a, g, l') \in E$ is fired if its guard g (i.e., a condition on variables and clocks) is enabled. If the transition is taken, its action a (i.e., assignment of variables and clocks) is performed[1]. The UPPAAL modeling language also extends the timed automata with concepts such as *synchronization* that allow state automata to communicate via *channels* [9]. More specifically, two or more automata will take a transition at the same time if they are synchronized on the same channel `chan`. Messages are sent on a channel through use of the syntax `chan!`, and that process is synchronized by (possibly multiple) receivers via the syntax `chan?`.

A. Trains

In our model, we allow an arbitrary number of trains to be specified, but we considered two trains in our experiments, because the system design allows only for two trains in the junction. We abstract away the length of the train and model the train as a single point on the track, because only the point of contact matters when dealing with derailment and collisions. Each train with an `id` is characterized by its horizontal distance, position on the track topology, speed, acceleration, and direction of travel. So each train has the variables $V_T = \{dist, trackNum, velocity, acceleration, direction\} \in V$.

We simplify the train's braking curve by specifying a constant acceleration and deceleration c. We define a Loc automaton with a single state $move$ and the invariant $I(move) = dist' == direction \times velocity \wedge velocity' == acceleration$.

We define a separate automaton that periodically sends a train status update message (`sendLoc?`) to the server. We define global variables V_M representing the message contents. The transition with the `sendLoc?` synchronization has the action $V_M = V_T$, which updates the message contents.

We also define a Control automaton that describes how the train's movement is controlled, as shown in Figure 5a. The trains are initialized in a halted state. When the transition `locFront?` is fired, an MA is received, and the action is calculation of the distance at which the train should start braking, `DIST_THRESH`. The train transitions to an accelerated state with action $acceleration = c$. When the guard $velocity \geq$ `MAX_VELOCITY` is enabled, the train transitions to a fixed-speed state with action $acceleration = 0$. The train transitions to a braking state with the action $acceleration = -c$ when its guard $direction \times dist \geq direction \times$ `DIST_THRESH` is enabled. When the train stops and has reached a triggering point, which is a fixed distance away from the nearest point, it sends a request route message by firing the transition `reqRoute!`. The message is resent periodically until an MA is received.

B. Track

While the Loc automaton models the physical distance, the Track automaton models the discrete position of the train in the topology, as shown in Figure 5b. We model the topology of the track by dividing it into track segments similar to those shown in Figure 4. The guards for the transitions in the Track automaton are based on the distance of the train and the status of the points. When the train derails because of a particular alignment of points, it transitions to a bad state, DERAIL.

C. Servers

Each of the servers (ATS, ATC, and CBI) has a separate automaton. The servers start from an idle state and transition to a processing state when they receive a message. They then use our custom-defined functions to process the received information. Then they transition back to the idle state after sending the appropriate message. The automaton for the process threads in the ATC server is shown in Figures 5c and 5d. The automata for the two other servers are similar, so we do not show them in this paper.

The ATS server automaton receives train status update (`sendLoc?`), request route (`reqRoute?`), and point status update (`pointStatus?`) messages. The server maintains its local variables V_S, which describe the position of the trains, $V_S = V_M$. When the transition `reqRoute?` is fired, the action is a function that checks which of the four routes (mentioned in the previous section) are available; if one is available, it books the path (e.g., T8, T5, T4). The transition `pointSet!` is then fired, sending a message to the CBI server, with the action assigning the message contents to global variables. The contents include the points to be set, and an indication of whether each point should be set in the normal or reverse position. When the transition `pointStatus?` is fired, the action is setting of the route. When the train clears the last segment in its set route (e.g., T4), the server releases the route.

The ATC server automaton receives train status update messages and point status update messages. When the transition `sendLoc?` is fired together with the corresponding synchronization from the train automaton, the action is updating of the server's local variables and firing of a `frontLoc!` transition to send an updated MA to the trains' automata. When the transition `pointStatus?` is fired, the action is calculation of an updated MA and firing of the `frontLoc2!` transition. That transition is synchronized with the train held at the triggering point, allowing the train to move until it clears its set route.

Finally, when the CBI server automaton's transition `pointSet?` is fired, the action is setting of the POINTS variables' values as given by the global message variables. Then, the `pointStatus!` transition is fired.

D. Attacker Capabilities

In general, the attacker capabilities we consider in this paper are (1) removing, (2) delaying, and (3) inserting network messages and control commands (i.e., CBI's sending of a signal to set points). For each class of attack, we define a generic model pattern that can be applied to a specified target.

The circumstances under which an attack is performed can be either probabilistic or deterministic. For example, the attack on spoofing messages between the trainborne system

[1]UPPAAL allows custom-coded functions in the actions to perform the assignment of variables and clocks.

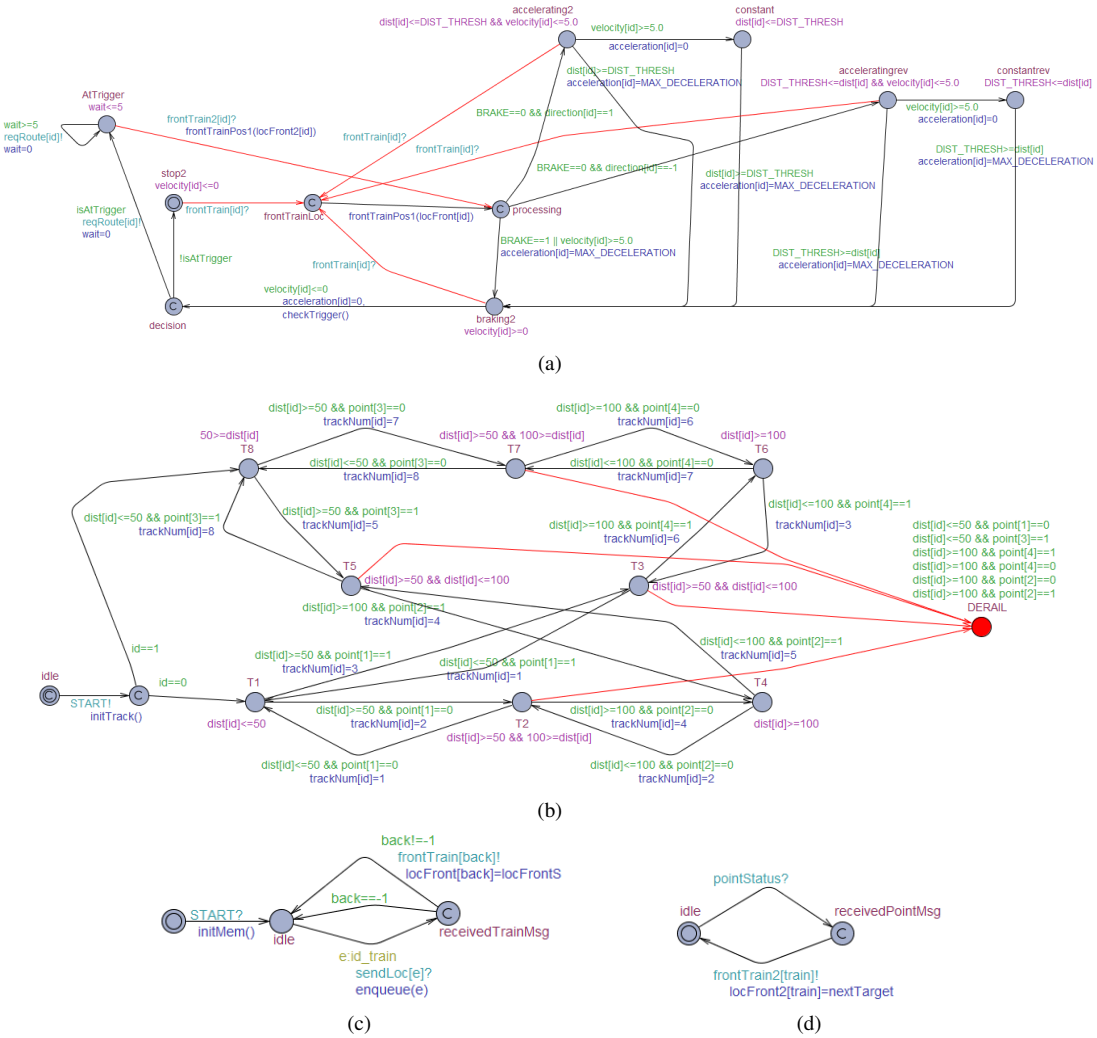


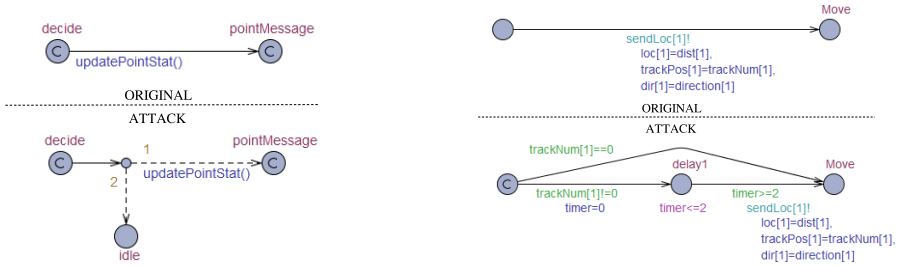
Fig. 5. The state automata for (a) the trainborne system, (b) the track layout, (c) the train message thread in the ATC server, and (d) the point message thread in the ATC server.

and server mentioned in [30] involves a probabilistic chance of being able to successfully craft a message. On the other hand, a jamming attack relies on the location of the jammer in the case of certain wireless communications, and as such can be represented as a guard. We represent that circumstance as a branch point when probabilities are involved or as a committed location when deterministic decisions are involved. These two modeling components can be combined when the circumstances involve both a deterministic choice and a probabilistic bound.

We first model the class of attacks that involve removal of a network message or command. If the attacker is trying to remove a network message, the target of the attack is a transition with a synchronization label `message`. If the attacker is trying to remove a command, the target is a transition with the corresponding action. Targeting a `message` can happen on either the sender's or receivers' side. If the sender's `message!` is targeted, all receivers are equally affected by the change. On the other hand, if a receiver's `message?` is targeted, only that receiver is affected. Removal of a network message involves removal of the synchronization `message?` or `message!` from the transition and removal of any other actions that refer to the assigning of the message contents. Removal of a command involves removal of the actions associated with the command. We show in Figure 6a an attack that probabilistically succeeds in removing a command.

Delay of a network message or command $(u, a, g, v), u, v \in L$ by t time units involves addition of a location x and a new clock $delayTimer \in C$. The invariant of x is $I(x) = delayTimer \leq t$. The transition (u, a, g, v) is replaced by (x, a, g', v) with a guard $g' = delayTimer \geq t$, which implies that the delayed time has elapsed. A transition (u, a', g, x) is added with the update $a' = delayTimer = 0$ to start the timer. We show in Figure 6b an attack that deterministically decides when to delay a network message.

Finally, insertion of a network message or command is more circumstantial than the first two classes of attacks. In general, we can model such an insertion by creating a new automaton and modeling the circumstances under which the insertion should take place. The insertion itself is modeled by a transition with either a synchronization `message!` in the case of network messages, or updates in the case of a command.



(a) (b)

Fig. 6. Model template that (a) removes a command with probability 0.3, and (b) delays a network message by 2 time units if the train is on a given track.

E. Safety Property

In this paper, we focus on verifying the safety of a railway system. The verification of service availability is discussed further in Section VI. The safety property is specified in temporal logic and has three requirements, as follows:

$$A\Box(\neg Track(0).\texttt{DERAIL} \wedge \neg Track(1).\texttt{DERAIL}) \tag{1}$$

$$A\Box(\neg(Track(0).\texttt{T3} \wedge Track(1).\texttt{T5}) \wedge \neg(Track(0).\texttt{T5} \wedge Track(1).\texttt{T3})) \tag{2}$$

$$A\Box\ (\ |dist[0] - dist[1]| \leq 20 \implies \nexists p \in \text{POINTS s.t. } trackNum[0], trackNum[1] \in connection(p)\ ) \tag{3}$$

where POINTS $= \{point[i], i \in [1,4]\}$ is the set of points and $connection$ is a function that maps points to the set of tracks that it connects in the topology. Formally,

$$connection(p) = \begin{cases} \{T_i, T_j\} & \text{if } p = 0, \\ \{T_i, T_k\} & \text{otherwise} \end{cases}$$

Statement (1) means that a train must not transition into a derailed state, i.e., the train must not run through a point that is not aligned in the right direction. Statement (2) implies that two trains must not be located on tracks that cross each other, i.e., T3 and T5. Finally, statement (3) implies that if the two trains are on tracks that are connected, they must be separated by a safe distance of at least 20 meters.

V. Experiments

In this section, we generate multiple state automata representing the system subjected to a set of attacks. We then use statistical model checking to verify system safety. If the safety property is not satisfied, UPPAAL will generate a counterexample showing a possible attack execution trace that results in an unsafe system state. By varying the set of attacks modeled in the state automata, we analyze the extent to which attackers will be able to compromise system safety, and we then propose countermeasures to those attacks.

Since the model we construct is complex and large, the state space explosion makes it hard to perform a full state space exploration. Statistical model checking, however, does not involve exploration of the full state space to determine satisfiability. Instead, it relies on generation of multiple simulations of the system and use of statistical reasoning to determine satisfiability. Although it does not cover all possible system trajectories, it gives a confidence bound on the probability of the satisfiability. Therefore, we use UPPAAL's SMC extension to model check our state automata.

We specify the safety property defined in Section IV as the below query in UPPAAL's syntax. We abstract away some syntax and details involved in listing the crossing tracks (i.e., T3 and T5) and the set of connected tracks. The model checker will return the probability that within 100 time units of simulation, the safety property will always be satisfied.

$$Pr[\leq 100]([\,]\ !Track(0).\text{DERAIL}\ \wedge\ !Track(1).\text{DERAIL}\ \wedge\ (-20 \geq dist[0] - dist[1] \geq 20\ \vee\ (Track(0) \neq Track(1)\ \wedge\ \text{notCrossing}(Track(0), Track(1))\ \wedge\ \text{notConnected}(Track(0), Track(1))\ )))$$

A. Experiment Setup

The initial parameters to the model include the trains' starting positions and the status of the points. There are three possible combinations of valid starting positions: (1) T1 and T8, (2) T1 and T6, and (3) T6 and T8. There are sixteen possible combinations of initial point positions. In total, we have 48 input configurations. We set the train status updates to be sent periodically every 300 ms. The maximum speed of the trains is 33 m/s. Finally, the length of each straight track segment (T1, T2, T4, T6, T8) is 50 meters.

The SMC extension in UPPAAL allows the modeler to configure the statistical parameters for model checking. We set the probabilistic uncertainty to 0.001 with a 95% confidence interval. We conducted the experiments on a Windows 10 Pro machine with a 3.4 GHz CPU core and 32 GB of RAM.

We consider a subset of the attacks defined in Table I that have the potential to affect the system's safety. Since we do not focus on system availability in this work, we do not consider attacks that merely reduce the quality of system services, such as by delaying the sending of the MA to the train. For each of the attacks that we consider, we define in Table II the circumstances under which the attack is executed and the extent of the attack.

B. Results

We tested for system safety under different combinations of attacker capability and input configurations. The results of our experiments are shown in Table III, where we show the number of input configurations in which safety was violated.

TABLE II
LIST OF ATTACKS DESCRIBING THE CIRCUMSTANCES IN WHICH THEY ARE EXECUTED AND THE EXTENT OF THEIR EFFECTS.

Attacker Position	Target	Attack Class	Circumstance	Parameter	Value
Outside	Train Status Update (0)	Remove	Train located on track segment	Number of track segments	2 and/or 3
		Delay	N/A	Number of time units	2–4
		Insert	Every second after requesting route, probabilistically determine success of attack	Probability of success	0.01
Outside	Movement Authority (6)	Insert	Every second, probabilistically determine success of attack	Total number of messages inserted	1–10
		Remove	Every second, probabilistically determine success of attack	N/A	N/A
ATC Server	Movement Authority (6)	Insert	Send MA after train requests route	N/A	N/A
CBI Server	Set Points (4)	Remove	N/A	N/A	N/A
		Delay	N/A	Number of time units	1–10
Station Network, ATC Server, or CBI Server	Point Status Updates (5)	Insert	Probabilistically determine success of attack	Probability of success	1 or 0.01

The numbers of runs that UPPAAL took to decide there was safety or to provide a counterexample range between 1,843 and 951,597.

1) Outsider: As mentioned in Section III-B, the outsider can access the communication between the trainborne system and the base stations. The least skilled outsider can jam signals and potentially delay network communications. Although such attacks can affect service availability, the attacker will not be able to affect system safety, as shown in Table III.

If the communication channel is not properly secured (i.e., there are no authentication or integrity checks), an attacker can inject messages into the communication channel with a probability of 1. Even with current security practices, a skilled outsider can utilize vulnerabilities in the communication protocol and inject messages, but with a probability of 0.01, as measured in [30].

We experimented with inserting one message in which the MA was specified as the end of the track line; that would allow the train to move unhindered as if there were no obstacles. As shown in Table III, insertion of just one MA message does not affect system safety, because the ATC server will continue to send MAs that will readjust the train's movement to a safe state. However, if such insertions are combined with either delay or removal of the train status messages, or removal of the MAs, the system safety is impacted.

Delay of the train status messages affects fewer configurations, since the train will only move past the first track segment and cannot proceed past the second track segment, because the MA sent to it will ensure that it stops before the second point. It causes unsafe states when the two trains move into the crossing tracks (T3 and T5) or when the train takes the wrong path from T1 to T3. However, removal of the train status messages affects almost all input configurations except for the case in which the trains are on T1 and T6 and all the points are in the normal state. Then, the tracks are parallel to each other, and both trains move as per normal.

2) ATC Server: An attacker positioned at the ATC server can manipulate the MA messages sent to the trains. Unlike the outsider, the insider has full control over the MA, and thus the execution of the attack has a success rate of 1. We consider the case when the attacker sends an MA specifying the end of the line to the train after it has stopped at the triggering point. Since the attacker does not have control over the locking of the points and the train status update messages, the attacker cannot cause an unsafe state when the two trains are on tracks T1 and T6. Because of the locking of the routes, the two trains will move in parallel with each other. However, for the other two track configurations (i.e., T1 and T8, and T6 and T8), the routes of the trains overlap with each other, and the trains should proceed one after the other. Thus, the attack will cause either derailment of one train or a collision at the crossing.

3) CBI Server: An attacker positioned at the CBI server can manipulate the commands sent to the point machines. We consider the case in which the attacker drops or delays the commands sent to the point machines while continuing operations as per normal. Since the insider has full control over the commands, the execution of the attack has a success rate of 1. For the attack to succeed when it is delaying the commands, we used the model checker to determine that the minimum delay is 900 ms. The attack affects almost all track configurations unless the points have already been set in the correct position for both trains. The attacks cause the trains to derail or move onto the wrong path.

4) Station Network: Finally, the last attack we consider is the addition of the point status update. The attacker may be positioned at the CBI server, at the ATC server, or within the station network. If the attacker is within the servers, the attacker has full control, and thus the attack always succeeds. Otherwise, if the attacker is within the station network, the attack has only a 0.01 probability of success, because of the security mechanisms in place. The effect is similar to that of an injection of an MA from the ATC server, so this attack affects the same number of track configurations.

5) Safety Countermeasures: Several safety countermeasures may be in place in the system. In particular, if a train does not receive an MA within a time limit, the train will come to an emergency stop. This safety measure affects how far the train will move even after an attacker has successfully injected a single MA message. So injection of a single MA message and removal of the train status update do not affect as many

TABLE III

RESULTS OF TESTING FOR SYSTEM SAFETY WITH AND WITHOUT SAFETY COUNTERMEASURES. FOR EACH SET OF ATTACK CAPABILITIES, WE LIST THE NUMBER OF INPUT CONFIGURATIONS THAT VIOLATE SAFETY, AND THE AVERAGE PROBABILITY OF THE SYSTEM'S BEING SAFE.

Attacker Position	Attack Capability	Affected Input Config	Probability Range
Outsider	Remove and delay train status update	0	[0.998,1]
	Add 1 MA	0	[0.998,1]
	Add 1 MA, delay train status update	24	[0.910,0.912]
	Add 1 MA, remove train status update	44	[0.910,0.912]
ATC Server	Add MA	32	[0,0.002]
CBI Server	Remove point set command	46	[0,0.002]
	Delay point set command	36	[0,0.002]
Station Network, CBI or ATC Server	Add point status update	32	[0.910,0.912] or [0,0.002]
Safety Countermeasure: Stop When No MA Received			
Outsider	Add 1 MA, remove train status update	24	[0.910,0.912]
	Add 9 MA, remove train status update	44	[0.910,912]
	Remove MA, insert train status update	32	[0.910,0.912]
Safety Countermeasure: Secondary Track Detection System			
Outsider	Add 1 MA, delay train status update	0	[0.998,1]
	Add 1 MA, remove train status update	0	[0.998,1]
ATC Server	Add MA	0	[0.998,1]
	Add MA, remove trackside device signal	32	[0,0.002]
Station Network, ATC Server	Add point status update	0	[0.998,1]
	Add point status update, remove trackside device signal	32	[0.910,0.912] or [0,0.002]

configurations. To affect as many configurations as before, the attacker needs to inject more MA messages to prevent the train from coming to an emergency stop. We used the model checker to determine that the minimum number of messages that need to be injected is nine.

Under that new safety measure, an attacker could drop MA messages to force a train to stop while he or she is injecting (or modifying) train status updates to trick the servers into believing that the train has cleared its route. We created a separate automaton that modeled the movement of an imaginary train A that accelerates faster than the actual train. The location of the imaginary train is then used in the train status updates after the train has passed the triggering point. When the servers believe that train A has cleared its route, the other train will be given the go-ahead to move. The two trains will then collide, since train A has stopped before its expected destination, unless the tracks are parallel to each other (i.e., the two trains are on T1 and T6).

Another safety countermeasure is to use secondary track detection systems that are legacy devices such as axle counters and track circuits that use physical mechanisms to detect the presence of a train on a given part of a track. The servers can correlate data sent from such trackside devices with train status update messages to check the position of a train as it enters a junction crossing. This safety countermeasure prevents an attacker from deceiving the servers about the train position, so attacks that involve delaying, inserting, and removing train status update messages will fail. Thus, all attacks from the outsider adversary will fail, but an insider with access to either the ATC server or the station network would be able to drop the signal from such trackside devices and successfully carry out the attacks.

In conclusion, a less skilled attacker who can only jam communication signals from the train will not be able to affect system safety. However, even under current security protections, the vulnerabilities in the network protocol between the trainborne system and the server can be exploited to violate system safety. Secondary track detection systems can stop outsiders from attacking the system. However, if an attacker has gained access to the internals of the system, such as the station network or the servers, he or she can easily force the system into an unsafe state. Therefore, it is vital that the communication channel between the trains and the station be fully secured, and that good security practices be enforced within the station network to ensure that safety-critical components are isolated and well-protected.

VI. DISCUSSION AND FUTURE WORK

In this paper, we analyze the safety of a diamond crossing by taking into consideration the possible attacks on the network protocol and commands issued to point machines. Based on our results, we suggest the need for additional security mechanisms and secondary track detection systems to be put into place. However, such an increase in safety and security countermeasures would involve an overhead in terms of management and could decrease route capacity because of the increase in latency. Our work gives the security practitioner a preliminary insight into the level of security countermeasures that could be deployed in a system.

In future work, we will investigate the safety in other locations on the railway line, such as railway yards, to gain a better understanding of the overall system safety. We can reapply our model to those locations by modifying the track topology model and routing table functions in our server automaton. While we do not consider system availability in this work, we intend to extend our analysis to include the impact of attacks on system service. In addition to attacks on network messages and commands, we aim to consider other factors such as the traction power supplied to the track, relevant signals sent to the train, and the environmental conditions of the railway tunnel that are controlled by computing devices in the station.

From a technical perspective, we want to improve the model checking techniques we've used in this paper. Although statistical model checking provides a good measure of system safety, we want to be able to say with certainty that safety is guaranteed under certain conditions. We are looking at ways to reduce model complexity by removing the differential equation that represents the train movement and discretizing certain elements of the model. We also intend to bound the model to prevent state space explosion. Then, we can use UPPAAL's in-built model checker to explore the full state space.

VII. Conclusion

It is important to analyze the safety of a cyber-physical system under threat from different attacker models. In this paper, we describe a safety analysis we conducted for a railway signaling system subjected to both outsider and insider threats. We modeled the adversary's attacks by using model templates that can be applied to a state automaton representing the normal system operations. We defined the circumstances under which an attack may be carried out and represented the attack in UPPAAL. We applied our attack model templates to the railway system and focused on modeling attacks on the network messages and cyber commands issued to physical devices. Our results show how the different combinations of attack capabilities with respect to the position in the system held by the attacker can affect system safety. This preliminary analysis of system safety also provides insight into which defense countermeasures can deter attacks.

References

[1] J. Rouvroye and E. van den Bliek, "Comparing safety analysis techniques," *Reliability Engineering & System Safety*, vol. 75, no. 3, pp. 289–294, 2002.

[2] J. Smith, S. Russell, and M. Looi, "Security as a safety issue in rail communications," in *Proceedings of the 8th Australian Workshop on Safety Critical Systems and Software*, 2003, pp. 79–88.

[3] N. Howe, "Cybersecurity in railway signalling systems," *Institution of Railway Signal Engineers News*, vol. 236, pp. 1–4, 2017.

[4] W. G. Temple, Y. Wu, B. Chen, and Z. Kalbarczyk, "Reconciling systems-theoretic and component-centric methods for safety and security co-analysis," in *Computer Safety, Reliability, and Security*, S. Tonetta, E. Schoitsch, and F. Bitsch, Eds. Cham: Springer, 2017, pp. 87–93.

[5] W. Young and N. Leveson, "Systems thinking for safety and security," in *Proceedings of the 29th Annual Computer Security Applications Conference*. ACM, 2013, pp. 1–8.

[6] G. Macher, A. Höller, H. Sporer, E. Armengaud, and C. Kreiner, "A combined safety-hazards and security-threat analysis method for automotive systems," in *Computer Safety, Reliability, and Security*, F. Koornneef and C. van Gulijk, Eds. Cham: Springer International Publishing, 2015, pp. 237–250.

[7] D. Kushner, "The real story of Stuxnet," *IEEE Spectrum*, vol. 50, no. 3, pp. 48–53, 2013.

[8] E. Denney and G. Pai, "Evidence arguments for using formal methods in software certification," in *Proc. IEEE International Symposium on Software Reliability Engineering Workshops*, 2013, pp. 375–380.

[9] "UPPAAL," http://www.uppaal.org/, accessed: 2018-05-20.

[10] A. Borälv, "The industrial success of verification tools based on Stålmarck's method," in *Computer Aided Verification*, O. Grumberg, Ed. Springer, 1997, pp. 7–10.

[11] A. Fantechi, "Twenty-five years of formal methods and railways: What next?" in *Software Engineering and Formal Methods*, S. Counsell and M. Núñez, Eds. Cham: Springer, 2014, pp. 167–183.

[12] S. Busard, Q. Cappart, C. Limbrée, C. Pecheur, and P. Schaus, "Verification of railway interlocking systems," in *Proc. 4th International Workshop on Engineering Safety and Security Systems*, 2015, pp. 19–31.

[13] M. Oz and O. Kaymakci, "An automatic formal model generation and verification method for railway interlocking systems," *Gazi University Journal of Science*, vol. 30, pp. 133–147, 2017.

[14] H. Wang, F. Schmid, L. Chen, C. Roberts, and T. Xu, "A topology-based model for railway train control systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 2, pp. 819–827, 2013.

[15] S. Qiu, M. Sallak, W. Schön, and Z. Cherfi-Boulanger, "Availability assessment of railway signalling systems with uncertainty analysis using statecharts," *Simulation Modelling Practice and Theory*, vol. 47, pp. 1–18, 2014.

[16] B. Chen, C. Schmittner, Z. Ma, W. G. Temple, X. Dong, D. L. Jones, and W. H. Sanders, "Security analysis of urban railway systems: the need for a cyber-physical perspective," in *International Conference on Computer Safety, Reliability, and Security*. Springer, 2014, pp. 277–290.

[17] S. Ghosh, P. Dasgupta, C. Mandal, and A. Katiyar, "Formal verification of movement authorities in automatic train control systems," in *Proc. International Conference on Railway Engineering*, 2016, pp. 1–8.

[18] P. di Tommaso, F. Flammini, A. Lazzaro, R. Pellecchia, and A. Sanseviero, "The simulation of anomalies in the functional testing of the ERTMS/ETCS trackside system," in *Proc. 9th IEEE International Symposium on High-Assurance Systems Engineering*, 2005, pp. 131–139.

[19] X. Han, T. Tang, J. Lv, and H. Wang, "Failure analysis of Chinese train control system level 3 based on model checking," in *Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification*, T. Lecomte, R. Pinger, and A. Romanovsky, Eds. Cham: Springer International Publishing, 2016, pp. 95–105.

[20] Q. Cappart, C. Limbrée, P. Schaus, J. Quilbeuf, L. Traonouez, and A. Legay, "Verification of interlocking systems using statistical model checking," in *Proc. IEEE 18th International Symposium on High Assurance Systems Engineering*, 2017, pp. 61–68.

[21] R. Bloomfield, M. Bendele, P. Bishop, R. Stroud, and S. Tonks, "The risk assessment of ERTMS-based railway systems from a cyber security perspective: Methodology and lessons learned," in *Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification*, T. Lecomte, R. Pinger, and A. Romanovsky, Eds. Cham: Springer International Publishing, 2016, pp. 3–19.

[22] M. Rocchetto and N. O. Tippenhauer, "Towards formal security analysis of industrial control systems," in *Proc. of the ACM Asia Conference on Computer and Communications Security*. ACM, 2017, pp. 114–126.

[23] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–208, 1983.

[24] M. Rocchetto and N. O. Tippenhauer, "CPDY: Extending the Dolev-Yao attacker with physical-layer interactions," in *Formal Methods and Software Engineering*, K. Ogata, M. Lawford, and S. Liu, Eds. Cham: Springer International Publishing, 2016, pp. 175–192.

[25] M. Puys, M.-L. Potet, and A. Khaled, "Generation of applicative attacks scenarios against industrial systems," in *Foundations and Practice of Security*, A. Imine, J. M. Fernandez, J.-Y. Marion, L. Logrippo, and J. Garcia-Alfaro, Eds. Cham: Springer International Publishing, 2018, pp. 127–143.

[26] G. Baker, "Schoolboy hacks into city's tram system," http://www.telegraph.co.uk/news/worldnews/1575293/Schoolboy-hacks-into-citys-tram-system.html, January 11, 2008.

[27] S. Y. Chang, B. A. N. Tran, Y. C. Hu, and D. L. Jones, "Jamming with power boost: Leaky waveguide vulnerability in train systems," in *Proc. IEEE 21st International Conference on Parallel and Distributed Systems*, 2015, pp. 37–43.

[28] M. Heddebaut, S. Mili, D. Sodoyer, E. Jacob, M. Aguado, C. P. Zamalloa, I. Lopez, and V. Deniau, "Towards a resilient railway communication network against electromagnetic attacks," in *Proceedings of the Transport Research Arena 5th Conference: Transport Solutions from Research to Deployment*, 2014, pp. 1–10.

[29] I. Lopez and M. Aguado, "Cyber security analysis of the European train control system," *IEEE Communications Magazine*, vol. 53, no. 10, pp. 110–116, 2015.

[30] T. Chothia, M. Ordean, J. de Ruiter, and R. J. Thomas, "An attack against message authentication in the ERTMS train to trackside communication protocols," in *Proc. of the Asia Conference on Computer and Communications Security*. ACM, 2017, pp. 743–756.

[31] A. David, K. G. Larsen, A. Legay, M. Mikučionis, and D. B. Poulsen, "UPPAAL SMC tutorial," *International Journal on Software Tools for Technology Transfer*, vol. 17, no. 4, pp. 397–415, 2015.