On Train Automatic Stop Control Using Balises: Attacks and a Software-Only Countermeasure

William G. Temple*, Bao Anh N. Tran*, Binbin Chen*, Zbigniew Kalbarczyk†, William H. Sanders‡

*Advanced Digital Sciences Center, Illinois at Singapore, 1 Fusionopolis Way, Singapore 138632
†Electrical and Computer Engineering Dept., University of Illinois at Urbana-Champaign, Urbana, IL 61801
‡Electrical and Computer Engineering Dept., University of Illinois at Urbana-Champaign, Urbana, IL 61801

Abstract—The components and systems involved in railway operation are subject to stringent reliability and safety requirements, but up until now the cyber security of those same systems has been largely under-explored. In this work, we examine a widely-used railway technology, track beacons or balises, which provide a train with its position on the track and often assist with accurate stopping at stations. Balises have been identified as one potential weak link in train signalling systems. We evaluate an automatic train stop controller that is used in real deployment and show that attackers who can compromise the availability or integrity of the balises' data can cause the trains to stop dozens of meters away from the right position, disrupting train service. To address this risk, we have developed a novel countermeasure that ensures the correct stopping of the trains in the presence of attacks, with only a small extra stopping delay.

Index Terms—train automatic stop control, cyber physical system security, railway system, balise, simulation

I. INTRODUCTION

Modern railway systems increasingly rely on real-time communication, embedded computing, and control to support or replace human drivers/operators. As an example, many of today's state-of-the-art metro (mass rapid transit) systems are driverless. The rail industry has always placed a strong emphasis on reliability, availability, maintainability and safety, and this has continued to be an area of focus when it comes to cyber systems and assets. However, until recently, the cybersecurity of those same systems and assets had not been seriously considered. Over the last 5–10 years, the emergence of cyber attacks targeting industrial control systems and physical infrastructure has contributed to a greater focus on evaluating and improving the cybersecurity of railway communication and control systems as well as the physical impact of a successful attack.

In many railway systems, transponders called absolute position reference beacons or balises are mounted on the track to assist with positioning [1]. When a train passes over a balise, the device will transmit an absolute position reference which is used to re-calibrate the train’s local position. This input source is combined with other sources (e.g., local measurements from a tachometer), however the balise position data is assumed to represent ground truth. Unfortunately, balises were not designed with security in mind, and to the best of our knowledge they contain no authentication or integrity checking mechanisms: the specifications in Eurobalise [1] only consider failures such as air-gap noise, cross-talk, and bit errors. In fact, the potential vulnerability of balises has been highlighted by the safety and security engineering communities [2], [3], [4]. In this paper, we examine the trustworthiness of balise position data and evaluate the impact of a loss of their availability and/or integrity on train operations. Our aim is to characterize the physical impact of bad position data and use this understanding to enhance the reliability and security of train operations. To achieve this, we consider the specific case of train automatic stop control (TASC), as described in [5].

In this scenario, a train approaching a station uses data from a series of balises placed in a specific manner to control its braking action, in order to stop at a specific target point (e.g., to match train doors with platform screen doors). Our study is based on the TASC setting and the braking controller design from [5], which has been successfully deployed in a real-world metro system and operated for years.

Our contributions in this paper are threefold:

- Based on the real-world setup and controller in [5], we quantify the significant impact of untrustworthy balise position data on TASC. Our evaluation is conducted using a high-fidelity train braking simulation model we implemented in Simulink.
- We propose a software-only countermeasure that requires no physical changes to the railway system. Our countermeasure provides assurances on the train stopping accuracy against all possible attacks under our threat model, and incurs small overhead in term of the overall time needed for train stopping control.
- We evaluate the operational assurance of our counter-
measure and demonstrate its superior performance. To understand the end-to-end impact on the passengers, we developed a train simulator based on the open-source Open Rails software [6] and incorporated real passenger data from the Shenzhen Metro [7] into our simulator.

The outline of the paper is as follows: Section II presents the background on train systems and train automatic stop control, focusing on the role that balises play in helping a train stop accurately. Section III presents the threat model for attacks that impact the system via balises, while Section IV evaluates the impact of those attacks. Section V then proposes countermeasures to balise attacks, and Section VI evaluates their effectiveness. Section VII discusses related work, the limitations of current study, and future work. Section VIII concludes the paper.

II. TRAIN AUTOMATIC STOP CONTROL

Like many other critical infrastructures, trains increasingly rely on intelligent devices and wireless communication rather than mechanical systems and human operators. The critical subsystems in a train control system are: automatic train supervision (ATS), automatic train operation (ATO), and automatic train protection (ATP). ATS works at the upper-most level, handling train scheduling and time tables. ATO works at the lowest level, providing redundancies and emergency braking capabilities to prevent train collisions. ATO systems control train speed, and can include anything from driver assistance mechanisms to fully autonomous train operation. Railway balises, which we discuss in the next section, are a critical component of both ATO and ATP systems.

A. The Role of Balises

Railway balises, also known as absolute position reference (APR) beacons or transponders, are found in railway systems around the world. Most commonly found in Europe and Asia, thanks to their inclusion in the European Train Control System (ETCS) [1] and Chinese Train Control System (CTCS) specifications [8], balises are also found in Japanese railway systems. There are two types of balises: active (controlled) and passive (fixed). We focus on passive balises in this work. A passive balise is a standalone device on the track that receives a radio frequency signal from a passing train and responds by transmitting its location. The train then uses this information, along with input from other sources (e.g., tachometer), to recalibrate its location and/or current speed (see Figure 1(a)). In ETCS terminology, this corresponds to Level 2 and Level 3 functionality. In contrast, active balises are connected to wayside equipment and can transmit local speed limits or even control commands.

Traditionally, the position data from balises is regarded as ground truth in train control operations. For example, in the ETCS the train’s onboard sensors produce an position estimate that depends on distance travelled [9], [10], and that estimate is re-calibrated every time the train encounters a balise. The purpose of our work is to critically examine this trust assumption by applying realistic train control models (Section II-B) and a clearly defined threat model (Section III).

B. Train Automatic Stop Control Model

To analyze the impact of untrustworthy balise position data on train operations, we apply a model from [5] and introduce additional logic to implement our threat model. We model a scenario where a train approaching a station controls its braking action using balise data to stop at a desired position. The model is based on real systems in use today [5]. It is common in many modern train systems to have a set of platform screen doors (PSDs) between the station platform and the train cars to improve commuter safety. In such systems, the coordination between the train doors and PSD system necessitates a precise stop.

We consider a system with $m$ balises, $B = \{B_1, B_2, \ldots, B_m\}$ where balise $B_m$ is the desired stop point. This is depicted in Figure 1(b) for a system with six balises. A stop is considered successful if it is within $\pm \gamma$ from the last balise. Each balise $B_i$ is represented as a tuple $B_i = < b_i, s_i >$ where $b_i$ is the physical location of the balise and $s_i$ is its reported location. Obviously in the absence of failures or malicious tampering $s_i = b_i$ for all balises. Positions are defined relative to the stopping point, i.e. $b_m = 0$, and $b_1 < b_2 < \ldots < b_m$. As the train passes over balise $B_i$ it will adjust its deceleration rate $\alpha$ to compensate for disturbances as it tries to achieve a full stop at $B_m$.

While there can be many types of disturbances affecting a controller’s performance (e.g., actuator delay, friction, wheel slip), we focus on variations in the characteristic parameters of the train braking system—time delay ($T_d$) and time constant ($T_p$). For the control model, we make use of the heuristic online learning algorithm (HOA) in [5] which has been successfully applied in an operational subway system over several years. Compared to a traditional
propotional/integral/derivative (PID) controller, which is not continuously tuned, the learning-based HOA control can better handle disturbances.

In this model there is a distinction between the expected deceleration without disturbances ($\alpha_i^e$), the controller deceleration ($\alpha_i^c$), and the estimation of the actual deceleration ($\alpha_i^r$) realized by the train. The expected deceleration as a train passes a balise is obtained by applying classical mechanics. Based on an initial velocity $v_i$ and a final velocity of 0, the constant deceleration (without disturbances) is given by:

$$\alpha_i^e = \frac{v_i^2}{2s_i}.$$  

Note that this takes a negative value since $s_i$ is negative with respect to the zero point. The controller deceleration is obtained by adjusting $\alpha_i^c$ in response to the observed train deceleration. Specifically,

$$\alpha_i^{c+1} = \alpha_i^{c+1} - \eta_i (\alpha_i^{r+1} - \alpha_i^{c+1}), \quad (1)$$

where

$$\eta_i+1 = \begin{cases} 0.95 \times \eta_i & \text{if } |\alpha_i^r - \alpha_i^c| > 0.05 \\ 1.05 \times \eta_i & \text{if } |\alpha_i^r - \alpha_i^c| \leq 0.05 \end{cases} \quad (2)$$

$\alpha_i^r$ is calculated using $v_i$, $v_{i+1}$, and the distance between successive balises $D_i = |s_i| - |s_{i+1}|$. The acceleration is given by:

$$\alpha_i^r = \frac{(v_{i+1}^2 - v_i^2)}{2D_i}.$$  

The actual train deceleration, $\alpha_i^{train}$, is influenced by various disturbances and system implementation features. We follow [5] and use a train braking model that was empirically determined based on data from an operating subway system. In the Laplace domain, the actual deceleration of the train is given by the transfer function:

$$\alpha_i^{train} = \frac{\alpha_0}{T_p s + 1} e^{-T_d s}, \quad (3)$$

where $T_d$ is the system’s time delay and $T_p$ is the time constant. The overall control model, which is implemented in Matlab/Simulink, is illustrated in Figure 2. The intuition behind the control process is that the HOA controller adjusts the braking force each time the train passes over a balise, in a manner similar to that of a driver. The adjustments allow the train to smoothly decelerate and stop at the target point in spite of characteristic lag and random disturbances. In our threat model, discussed in the next section, the balises’ data can be altered. Therefore, the model has been adapted to decouple the train’s true position and its perceived position on the tracks.

C. System-level Simulation

The train automatic stop control model introduced above captures the behavior of a single train at a single station. While much of our impact analysis (Section IV) focuses on that micro scale and quantifies a train’s performance in terms of stopping error (distance) and stopping time under different attack scenarios, a broader system perspective is necessary to fully understand how potential cyber attacks and countermeasures affect rail infrastructure. To this end, we have developed a railway system simulator based on an open-source tool that simulates train movement over user-defined maps and time tables.

There are a number of commercial and open-source railway simulation tools available online (see [11]). Ultimately, we selected Open Rails [6] because it (i) simulates train movement along user-constructed tracks and stations; (ii) is built on an engine with time-based simulation and object-oriented programming; and (iii) offers a 3D view of trains as well as a supervisory-level view of all train movement. We have added additional features to support the integration of passenger flow data: e.g., passengers’ station and time for tap-in and tap-out, the number of doors on each train, the train capacity, and adding a user-specified dwell time to be added to a train’s waiting time at each station.

We refer readers to [11] for further information about our customized train simulator. In short, we have built an enhanced version of Open Rails and modeled a line in the Shenzhen metro system to allow system-level modeling of the impact of attacks to balises. We will discuss selected system-level results in Section VI.

III. ATTACKS TARGETING BALISES

In a railway system, the key properties to uphold are passenger safety and operational reliability. In this work we focus on attacks with an operational impact, targeting the track infrastructure (i.e., balises). Previous research in railway system security has indicated that the presence of redundant systems and fail-safe mechanisms makes safety challenging for an attacker to compromise, yet those same systems can make service disruption easier to achieve [3, 12]. Within the context of operational reliability, we focus on attacks targeting track infrastructure rather than the rolling stock (trains) themselves for two reasons.

First, as a large, physically distributed infrastructure, railway track and wayside systems presents a large physical attack surface, whereas the rolling stock is typically housed at a central depot with strict access control. Second, while there are train onboard systems involved in localization (see Figure 1(a)), targeting the balises maximizes an attacker’s impact.
since multiple trains passing through a single compromised track section will be affected. Furthermore, this impact may be different for different trains and/or at different times, making detection more difficult.

We assume the attacker’s goal is to create a service disruption. This is achieved by manipulating balises (e.g., using a manufacturer’s balise reader/programmer device) to cause trains to have inaccurate stops at the station. By maximizing the stopping error, the attacker introduces additional dwell time at stations as the trains correct their alignment with the platform screen doors (either automatically or with human intervention). This process, called jogging, can have an adverse system level effect on train timetables. Within this context, we make the following assumptions:

- **Fixed Balises**: The attacker cannot physically move the balises, nor introduce additional balises. Only balise data availability and integrity may be altered.
- **Trusted Zero Balise**: The last balise, $B_m$, is assumed to be trusted. This balise, at position 0, is often active rather than passive, and it is not used for control in the HOA algorithm. This last balise interacts with other components (e.g., the platform screen door controllers) which would make tampering easier to detect. We assume the attacker is unable to make any other passive balise impersonate the zero balise.
- **Attacker’s Knowledge and Capability**: The attacker is familiar with the technology and operations of the rail system, including countermeasures deployed. Specifically, the attacks can be launched by insiders (e.g., disgruntled employees, malicious contractors) intentionally, or unintentionally (either by mistake, or through a malware-infected balise programming device). In addition, the vast physical attack surface of railway tracks makes it feasible for an external attacker who understand the working principal of balises to launch the attack.

Since the balises are passive, once the attacker alters the availability or integrity of one or more balises, those balises stay in the compromised state. This helps the attacker to create a prolonged impact on the system rather than a one-off incident. Additional assumptions related to our proposed countermeasures are discussed in Section V. In short, we assume that the trains are provided with prior knowledge about where balises are located (e.g., expect 6 balises prior to a station, reporting a fixed set of distance values), and we assume a certain accuracy level for the train’s onboard odometry.

In this work, we focus on the attack on a single station. The attacker can have access to one or more balises at one train station. For future work the study can be extended to consider attacks over multiple stations.

### IV. IMPACT OF ATTACKS

Before introducing and assessing countermeasures for balise availability and integrity attacks, we use the model in Section II-B to establish the impact of those attacks on the system as it is. For the results presented in the rest of this paper we consider a train approaching a station with balises located at positions $[-100, -64, -36, -16, -4, 0]$. The first five balises are passive, while the final one is active. There are several fixed and adjustable parameters that characterize the train stop scenario. We assume an initial position of $x = -100m$ (relaxed in Section VI), initial velocity $v = 10m/s$, maximum deceleration of $a_{max} = -1m/s^2$, and an allowable stopping error of $\gamma = 0.3m$. In our analysis we vary the brake parameters—the time delay ($T_d$) and time constant ($T_p$)—to capture the impact of control disturbances and variations between different trains. We consider a range of $\pm 20\%$ around default values $T_d = 0.6$ and $T_p = 0.4$ as used in [5].

To provide intuition into how the train uses balise data during a stop, Figure 3(a) shows the train’s acceleration curve with no attacks under the default simulation setting described above. Observe that at each balise location the train’s brake controller acceleration changes and the actual acceleration follows the change based on parameters $T_d$ and $T_p$. Figure 3(b) shows the resulting train speed curve, which achieves a smooth and accurate stop in the absence of disturbances or attacks.

#### A. Availability Attacks

Within the context of our threat model (Section III), a potential attacker has access to a single train station and the balises located therein. A natural attack to consider is an availability attack affecting one or more balises. This could be perpetrated by an insider or intruder, or it could simply represent a non-malicious failure from a reliability perspective. To assess the impact of availability attacks on train automatic stop control, we exhaustively evaluate all combinations of available and unavailable balises (31 in total, excluding the scenario where all balises are available). We simulate those 31 scenarios, each with combinations for 10 values of $T_d$ and 10 values of $T_p$ that are uniformly chosen to cover the $\pm 20\%$ range of their default values. This leads to a total of 3100 different settings.

The key metrics for evaluating the impact of balise data attacks (and later countermeasures) are the stopping error,
measured in meters with respect to the 0 point, and the stopping time. A typical value for the allowable stopping error is $\pm 0.3m$. For reference, a typical metro train’s door width is around 1.4m. During normal operations, a train will wait at a station for a dwell time of 30s to 1-2min (based on Singapore’s train system). If the train stops outside of its allowable stopping error it needs to correct either manually or automatically. This process can easily add a few minutes to the dwell time, potentially creating a cascading effect as subsequent trains are also delayed. Based on measurements taken in an automated train system, the additional time associated with automatic jog is substantial. Due to safety considerations, each automatic jog covers a small distance ($0.2 - 0.3m$ on average) and takes about 8.6s. Therefore, correcting a large stopping error (e.g., $5m$) can easily take over a dozen jogs, adding more than 100 seconds to a train’s dwell time at the station.

Out of the 3100 availability attack scenarios evaluated, only 138 (4.5\%) resulted in a successful stop within the allowable $\pm 0.3m$ range. The results are presented in Section VI together with the countermeasure analysis. Specifically, Figure 7 (a) shows scatter plots of the train stopping error (m) and total time to stop (s) for all 3100 scenarios, grouped by the number of unavailable balises. Intuitively, the operational consequences of an attack become more severe as the number of affected balises increases. For example, when two or more balises are unavailable the stopping error can be above 6m: a threshold where automatic jog will not be executed and a train that overshoots must skip the current station and proceed to the next one. In Figure 7 (b), which shows the stopping times for the 3100 scenarios, the cases where a station is skipped are marked as N.A.

To provide a concrete example of these availability attacks, Figure 4 shows acceleration and speed curves for two cases. The first attack makes balise $B_1$ unavailable. Contrasting Figure 4(a) with the no-attack baseline profile in Figure 3(a), we observe late braking action followed by sharper than normal deceleration as the train attempts to compensate. The resulting overshoot and stopping error of $1.3m$, while out of the allowable region, is not terribly disruptive: it takes around 62 seconds to correct the error. However, making a second balise unavailable (Figure 4(c)) causes the braking controller to saturate at $-1m/s^2$ and the train is unable to stop until $22.1m$ past the target point. As explained previously, this train would likely continue to the next station rather than jogging.

### B. Advanced Attacks

While availability attacks on balises can seriously disrupt train automatic stop control, under our threat model the adversary is not limited to these actions. We will discuss in Section V how the design of a countermeasure should deal with all combinations of such attacks. Before that, here we will briefly describe more sophisticated attacks involving data integrity attacks, and combinations of availability and integrity attacks. As we show, an attacker capable of changing the position reported by one or more balises can create disruptive train stopping errors by affecting fewer balises than an availability-only attacker.

**Integrity attack on a single balise:** An attack affecting balise data integrity is able to cause a significant stopping error. For example, an attacker could tamper with balise $B_1$ so that it reports $-4$m rather than $-100m$. As shown in Figures 5(a) and 5(b), the result is a hard stop and an undershoot of $-43.7m$, which would certainly be alarming for passengers. Based on the train jog parameters described previously it would take around 1646 seconds (over 27 minutes) to correct this position error without human intervention.

**Integrity attack on consecutive balises:** Similar to the above attack on the first balise, an attack overwriting the values of $B_1$ and $B_2$ to $-64$ and $-36$, respectively, will cause the train to think it is closer to the stop point than it actually is. The result, shown in the acceleration profile in Figure 5(c), is sharper-than-expected braking and a stop at $-29m$. While the impact in terms of stopping error is less severe due to the lower speed, this attack causes the train to pass consecutive balises reporting $-36$, which could cause an error with the HOA algorithm if exceptions are not handled correctly. This was the case in our original Matlab/Simulink implementation based on [5], and it is unclear whether the operational instances of the HOA controller contain the same error.

**Combination of availability and integrity attacks:** An interesting observation from our evaluation with HOA is that launching a pure integrity or availability attack is more impactful than more sophisticated attacks that combine both integrity and availability compromises. As shown in Figure 5(d), an attack making $B_1$ report $-64$, making $B_2$ unavailable, and making $B_3$ report $-4$ will result in a stop at $-27.6m$. This is certainly impactful, but not as severe as the cases we described earlier. Intuitively, due to the physics of train operation, the first few balises are the most critical: the train is far from the target stop point and moving fast, so for maximum impact an attacker would seek to keep it moving fast, or to brake quickly. Either of those actions can be achieved without mixing availability and integrity attacks.

However, if one considers more advanced controllers rather than HOA, this observation may no longer hold. Let us revisit Figure 5(a) and 5(c). For both cases, before the train fully stops, it actually passes through one balise that is not manipulated by the attacker (the $-64$m balise for Figure 5(a) and the $-36$m balise for 5(c)). While HOA’s heuristic control logic is not able to react fast enough to leverage such genuine balise inputs, a more resilient controller could potentially do so. However, if the attacker can combine availability/integrity attacks and manipulate balises as in Figure 5(d), the train does not have a chance to receive any genuine balise inputs before it enters a full stop, which can make the design of a resilient controller even more challenging.

### V. Countermeasures

In the previous section, our model-based analysis showed that an attacker could cause train stopping errors up to dozens of meters by compromising the availability or data integrity
of the balises at a station. We now propose software-only countermeasures that make a train’s automatic control robust to potential attacks to balises.

A. Design Intuition

The design of our countermeasure is based on four key observations, which we discuss in the paragraphs below.

Observation 1: Trains can reduce stopping errors by sacrificing some performance. Consider the scenario where all the balises used for the control algorithm are compromised (only the final stopping position can be learned). A train can still stop close to the right position by travelling at a low speed and applying the brake when it passes position 0. If the speed is low enough, the train should be able to stop within a short distance from the correct position. The drawback of this conservative strategy is that the train’s stopping performance (in terms of time) may be sacrificed. In particular, if the train slows down to 0.3m/s at 30m away from the position 0 (e.g., due to local position estimation error), the train then needs 100s to cover this short distance.

Observation 2: Trains can leverage the trustworthy physical-world setup information. In a railway environment, the distance between stations and the position of balises are fixed once the system is built and put into operation. It is a common practice to load such physical-world information into a train’s on-board controller and the controller can use that trustworthy information to make decisions. Note that our threat model assumes the attacker cannot change the physical setup, e.g., moving a balise to a different position. Such physical attacks would be time-consuming and easier to detect than cyber attacks.

Observation 3: Trains can leverage less accurate but more trustworthy sensing data. A train can infer its location using multiple sensors, including its onboard tachometer and Doppler radar. The estimation of location from onboard sensors is not as accurate as the value obtained directly from balises: the estimation error accumulates with the distance that a train travels. For example, estimates based on wheel rotation can be inaccurate when the wheels slip on the rails. Fortunately, with modern information fusion techniques and model-based controls, the onboard system is already able to provide
well-bounded accuracy based on these imperfect sensors. For
ETCS, the minimum accuracy requirement for on-board train
distance measurement is \(\pm (5m + 5\%d)\) where \(d\) is the distance
travelled since the last reference point [9]. In practice, the
accuracy for a system can be better: for example an error less
than \(\pm 20m\) for every 1000m travelled (\(\pm 2\%d\) [10]). Since we
believe it is harder for an attacker to compromise on-board
equipment, we consider these sensors more trustworthy than
balises, though they may be less accurate.

**Observation 4:** Trains can use trusted information to verify
information from untrusted sources. As mentioned above, both
the physical setup information and the position estimations
based on on-board sensors are deemed trustworthy information.
They can be used to cross check the report from the less
trustworthy sources (i.e., the balises). Once the information
from balises is deemed trustworthy, it can be used to improve
performance. In particular, consider a train that adopts the
conservative strategy after it misses the first balise. If it can
trust the information it gathers from a later balise, it will be
able to use that reading, e.g., to avoid slowing down too early.

**B. Detailed Design**

Based on our observations, we have developed a software-
only countermeasure for dealing with attacks on balise data.
To describe our countermeasure in detail we introduce some
additional notation: Let \(p\) denote the train’s position as it
approaches a station; let \(p_{\text{est}}\) denote the train’s onboard position
estimate, which has an associated position error \(\delta\); let \(v_{\text{con}}\)
denote a reduced speed setting. At a high level, our robust
controller uses two parallel control modules—the original
HOA controller, and a new conservative controller—which are
activated based on an anomaly detection and correction (ADC)
module. This control architecture is shown in Figure 6.

By default a train will operate using the HOA controller [5],
as it normally would. If the ADC module detects that there
is a balise missing, by using its knowledge about the physical
world setup, its onboard estimated position \(p_{\text{est}}\) in conjunction
with whether or not data is received, it will fall back to
a conservative controller (Observation 1). This detection is
possible thanks to the static physical-world setup information
(Observation 2) and the less accurate but more trustworthy
local sensing data (Observation 3). The conservative controller
uses 2 PID controllers to slow the train to \(v_{\text{con}}\), where it
remains until it reaches the stop marker. At that point the
train will brake at \(\alpha_{\text{max}}\) until it stops. There is a need to
select between multiple PIDs at different velocity stages due
to conflicting requirements in terms timing and stability. The
key parameter is \(v_{\text{con}}\), which is selected based on the train
system’s allowable stopping error \(\gamma\) (in our analysis, \(\pm 0.3m\)).
Note that this design avoids the need for further adjustment
through jogs after the train first stops. One can further trade
off between the (initial) stopping error and the overall stopping
time by allowing a small number of jogs to be carried out.

When the ADC switches to the conservative controller, it
will still check and correct the position information it receives.
The corrected information, denoted as corrected balise value

**Algorithm 1 Derive-Trustworthy-Info(), with \(p_{\text{est}}, \delta\) being
the estimate so far.**

1: if there is a unique known balise position \(b_i\) such that
\[|p_{\text{est}} - b_i| < \delta\] then
2: return \((b_i, 0)\) as new \((p_{\text{est}}, \delta)\);
3: else if there are multiple known balise positions \(b_i \in B\) such that
\[|p_{\text{est}} - b_i| < \delta\] then
4: set \(r \leftarrow \{\text{local} = p_{\text{est}}, \text{candidates} = B\}\);
5: append \(r\) into \(R\), an unverified record list;
6: if there are multiple records \(r_1, \ldots, r_k\) in \(R\) then
7: for \(i\) in 1, \ldots, \(k - 1\) do
8: \(d \leftarrow r_{k, \text{local}} - r_{i, \text{local}}\);
9: compute distance between all combinations of
\(r_i, \text{candidates}\) and \(r_k, \text{candidates}\);
10: if there is only one pair of candidates \((b_i, b_m)\) that
has distance of \(d\) then
11: return \((b_m, 0)\) as new \((p_{\text{est}}, \delta)\);
12: end if
13: end for
14: end if
15: end if

in Figure 6, is then supplied to the conservative controller. This
corrected information is determined based on Algorithm 1,
which returns an updated position estimate and error, if there
is new balise input that is deemed trustworthy. This implementa-
tion is inspired by our Observation 4. In particular, if even
a partial set of balise information can be obtained and their
trustworthiness can be attested, the position estimation can be
made more accurate. As a result the conservative controller’s
performance can be improved. For example, if the conservative
controller knows that it is still some distance away from the
final stopping point, it can wait a while before slowing down
to the pessimistic speed of \(v_{\text{con}}\). Opportunistically using
trustworthy information reduces the delay.

To derive trustworthy information (in spite of potential
availability and integrity attacks), Algorithm 1 runs two dif-
f erent checks based on trustworthy information:

**Check 1:** (line 1 in Algorithm 1): if the train encounters
a balise, and it can uniquely associate the balise with a
valid balise position, according to both its current trustworthy
estimate of \(p_{\text{est}}\) and \(\delta\), as well as the static information
about the physical setup, this encounter will be used to
update \(p_{\text{est}}\) and \(\delta\). Note that, \(p_{\text{est}}\) is updated to the balise
position \(b\) according to the static physical model, instead of
the information reported by the encountered balise. This is because
an attacker may manipulate a balise at an actual position of \(b\)
to report a different position of \(b'\). For example, if the train’s
local estimate is \(p_{\text{est}} = -90m\) from the stopping point, with a
\(\delta = 20m\), when it encounters a balise, the train can potentially
be at any point between \(-70m\) to \(-110m\) from the target.
Based on the static physical information, there is only one
valid balise that can fall in this region (at \(-100m\)). Hence, even
if the balise reports its value as some other value (e.g., \(-16m\)),
the train’s estimated position will be updated as \(-100m\).
**VI. Evaluation**

As in Section IV for the existing HOA train controller, we implement our proposed hybrid controller (Figure 6) in Matlab/Simulink to assess its performance. An ideal controller will be able to minimize stopping error and stopping time with and without the presence of cyberattacks. With the conservative controller there is an inherent tradeoff between the two objectives, since $v_{\text{min}}$ is low. However, as we show below, once the jog time is considered even this slower-acting controller performs strongly against HOA in a relative sense.

**A. Evaluating Availability Attacks**

In the absence of countermeasures, simply making one or more balises unavailable as an automated train stops can have serious operational consequences. From the analysis in Section IV, in over 95% of the evaluated stopping scenarios the train failed to reach its target by over or undershooting. We now examine those same attacks after implementing the countermeasures outlined in the previous section.

In Figure 7 we contrast the performance with and without the conservative controller, assuming 2% position estimation error. Subfigures (a) and (b) show the impact of attacks with no countermeasures, as discussed in Section IV. Below, in subfigures (c) and (d), we show results from the same 3100 availability attack scenarios after the countermeasures (anomaly detection & correction with a conservative controller) have been implemented. In every case the conservative controller ensures that the stopping error is within the acceptable $\pm 0.3m$ range, as indicated in Figure 7 (c). The other key metric, stopping time, is shown in Figure 7 (d) for the worst case situation where the train’s local position estimate is off by 20 meters. In other words, using its onboard sensors the train estimates that its position is $-80m$ when its actual position is $-100m$, therefore it slows down to $v_{\text{con}}$ more than 20m away from the actual stopping point and it could take around 100s to stop. Note that, the average case stopping time can be much smaller than this worst case, since the stopping time reduces linearly with reduced local estimation error.

Figure 7 also shows that when some balises are available, our algorithm can derive trusted information from them, and use that to reduce the stopping time. In comparison, while HOA (with potential additional automatic jogs), i.e., the original scheme without any countermeasure, can sometimes stop faster, there are 1083 (out of 3100) settings where the train overshoots by more than 6m, so it cannot be jogged back to the current station (marked as N.A. in the figure). There are an additional 290 cases where the train takes more than 150 seconds (including jog time) to stop. Even for the case when the attacker only makes a single balise at $-16m$ unavailable, the stopping time of HOA can be as long as 157 seconds, 50% worse than the worst case stopping time with our countermeasure. The gap further increases when more balises are made unavailable.

**B. Evaluating Advanced Attacks**

We also evaluate our countermeasure under advanced attacks, where an attacker can combine both availability and integrity compromises. Our evaluation confirms that our countermeasures are able to detect and discard all manipulated balise values, making advanced attacks equivalent to the case where an attacker simply launches availability attacks. Revisiting the attacks from Section IV-B, the integrity attack on $B_1$ (Figure 5(a)) has its stopping error reduced from $-43.7m$ to an acceptable $0.15m$; the corresponding elimination of jog time reduces the stopping time from 1646.4s to 19.8s. Similarly, the integrity attack on two balises (Figure 5(c)) reduces stopping error and time from $-29m$ and 1098.8s to 0.15m and 19.8s, respectively.

The final advanced attack, combining availability and integrity attacks (Figure 5(d)) illustrates the performance trade-off of the conservative controller in terms of stopping time. The countermeasure successfully reduces stopping error
Fig. 7. Evaluation of train automatic stop control performance in the presence of availability attacks before (top) and after (bottom) our countermeasures.

(a) Error (before jog), no countermeasure

(b) Time (after jog), no countermeasure

(c) Error with countermeasure

(d) Time with countermeasure. Worst-case for 2% local est. error

Stopping Error [m]

Threshold for opening doors

Threshold for automatic jog

Number of Balises Unavailable

1 2 3 4 5

Stopping Time [s]

10

100

1000

N.A.

Number of Balises Unavailable

1 2 3 4 5

Number of Balises Unavailable

1 2 3 4 5

Number of Balises Unavailable

1 2 3 4 5

Number of Balises Unavailable

1 2 3 4 5

Stopping Error [m]

Threshold for opening doors

Threshold for automatic jog

(a) Error (before jog), no countermeasure

(b) Time (after jog), no countermeasure

(c) Error with countermeasure

(d) Time with countermeasure. Worst-case for 2% local est. error

Fig. 7. Evaluation of train automatic stop control performance in the presence of availability attacks before (top) and after (bottom) our countermeasures.

 stopping time (1046.4s to 932.3s). However, this stopping time is still slower than under normal operating conditions; therefore we want to understand how the elongated stopping time impacts a rail system.

To explore the system-level impact we use our enhanced version of the Open Rails train simulator [6]. We built a model of Shenzhen Metro Line 1 [13], created timetables, and integrated publicly available real-world passenger data [7]. Then, we simulated the advanced attack at one of the busiest stations, Gou Wu Gong Yuan, during the morning rush-hour. Figure 8(a) shows a plot of the system punctuality index: a railway metric indicating the percentage of trains on time. This is defined as $P = \frac{t_s - t_l}{t_s} \times 100$ where $t_s$ is the number of scheduled trips within a certain period of time (2 minutes in this work), and $t_l$ is the number of lost or delayed trips. For the final attack scenario above, the worst-case punctuality with the countermeasure is 82%, while in the absence of countermeasures, the service level degrades rapidly to a low of 13%. The difference in passenger delay is also substantial, as shown in Figure 8(b). In the normal case, 4.1% of passengers have travel delays of 15min or longer. This number spikes to 64.7% without countermeasures, while many experience delays over 30min. After introducing countermeasures the performance is close to the normal case, with 4.5% of passengers delayed 15min or longer, and none over 30min.

VII. RELATED WORK AND DISCUSSION

Cybersecurity for rail transit systems has received less attention than other critical infrastructures such as electric power, water/gas distribution, and aviation. However, with proliferation of wireless networking, commercial-off-the-shelf products, and automation in rail systems this is starting to change [14], [15]. In terms of funded research projects, over the last few years some European efforts began to systematically study the cybersecurity [16], [17] and security/safety engineering [18] of railway systems. As far as standards and best practices, security has not been a focus in the past [19], however there is a growing effort to adapt standards and introduce new best practices to improve the security level of railway systems [20].

Much of the existing academic work on railway security has focused on security issues for the ERTMS and GSM-R in particular [21], [22]. Another area of work is on system-level modelling and analysis [23]. Our work adopts lower-level control models, and is not limited to ERTMS, although railway balises are a component of that system. Recent work examining security issues for rail transportation [3], [2], [12] indicates that balises could be a disruptive point of attack, and [4] examines possible attacks and countermeasures. However, to the best of our knowledge, our work is the first systematic investigation of attacks on railway balises using high-fidelity physics-based models.

There are certain limitations in this study that we would like to discuss before we conclude the paper. First, the threat model used in this work assumes that attackers are not able to introduce additional balises into the railway system. One could imagine situations where additional balises, or device imitating
balises, could be physically put into the system by determined attackers. Our proposed countermeasure would not directly work under such a threat model. How to defend against such attacks is an interesting open topic for future work. Second, within the context of the availability and integrity attacks considered, our study in this paper focuses on one train brake control algorithm (HOA from [5]) and one specific balise layout, as both are used in real deployment. However, the simulation tools we developed and the countermeasure we designed are not specific to the HOA algorithm or balise layout. One area of future work is to evaluate these attacks and countermeasure with other control logic and balise patterns.

Finally, our proposed software-only countermeasures is just one approach to reduce the damage of balise data attacks. There are other potential solutions that we have not explored. For example, one alternative is replacing existing railway balises with enhanced devices that contain stronger authentication and integrity checking mechanisms. Another important countermeasure is to strengthen the cyber and physical access control to prevent balises from being compromised. In comparison, our software-only solution minimizes changes to the legacy systems, hence could be easier for adoption. It also provides an additional layer of protection even when the balises are compromised.

VIII. CONCLUSION

We evaluate a train automatic stop controller that is used in real systems and show that it can be disrupted by compromising the availability or integrity of railway balise position data. To mitigate the threat, we propose and evaluate a software-only countermeasure using high-fidelity train braking models in Simulink and a system-level simulator based on the open-source Open Rails software. Our results show that the countermeasure ensures accurate train stopping with only a small extra stopping delay.

ACKNOWLEDGMENTS

This work was 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, and supported in part by Singapore’s Agency for Science, Technology, and Research (A*STAR) under a research grant for the Human-centered Cyber-physical Systems Programme at the Advanced Digital Sciences Center.

REFERENCES