Diagnosis-guided Attack Recovery for Securing Robotic Vehicles from Sensor Deception Attacks

Pritam Dash
pdash@ece.ubc.ca
University of British Columbia, Canada

Guanpeng Li
guanpeng-li@uiowa.edu
University of Iowa, USA

Mehdi Karimibiuki
mkarimib@ece.ubc.ca
University of British Columbia, Canada

Karthik Pattabiraman
karthikp@ece.ubc.ca
University of British Columbia, Canada

ABSTRACT

Sensors are crucial for perception and autonomous operation in robotic vehicles (RV). Unfortunately, RV sensors can be compromised by physical attacks such as sensor tampering or spoofing. In this paper, we present DeLorean, a unified framework for attack detection, attack diagnosis, and recovering RVs from sensor deception attacks (SDA). DeLorean can recover RVs even from strong SDAs in which the adversary targets multiple heterogeneous sensors simultaneously. We propose a novel attack diagnosis technique that inspects the attack-induced errors under SDAs, and identifies the targeted sensors using causal analysis. DeLorean then uses historic state information to selectively reconstruct physical states for compromised sensors, enabling targeted attack recovery under single or multi-sensor SDAs. We evaluate DeLorean on four real and two simulated RVs under SDAs targeting various sensors, and we find that it successfully recovers RVs from SDAs in 93% of the cases.

CCS CONCEPTS

• Computer systems organization → Robotic autonomy; • Security and privacy → Systems security.

KEYWORDS

Cyber-Physical Systems, Resilience, Physical Attacks

1 INTRODUCTION

Autonomous Robotic Vehicles (RV) such as drones and rovers RVs rely on their onboard sensors to perceive their physical states (e.g., position, angular orientation). Based on the physical states, specialized algorithms plan the RV’s autonomous navigation in a feedback control loop [24]. Unfortunately, attackers can feed erroneous sensor measurements to the RV through its physical channels (i.e., physical attacks), causing the RV to deviate from its course and/or crash [29, 41]. Physical attacks such as GPS spoofing have occurred in the wild on military drones [27] and marine systems [26].

We refer to physical attacks on RVs that target one or more sensors simultaneously as Sensor Deception Attacks (SDAs). SDAs have been demonstrated in autonomous driving systems [16, 36], and can also be extended to RVs. SDAs cannot be mitigated by traditional security techniques as they target the sensor hardware rather than the software. Further, attack detection [19, 38] is not enough, as the RV may crash after the detection [22]. Many drones have hardware fail-safe mechanisms (e.g., landing upon failure). However, activating these upon attack detection may not be safe, as the drone may land in adverse environments or fall into attackers’ hands [6]. Therefore, it is important to recover RVs from SDAs.

Prior work on attack recovery from physical attacks [31, 45, 46], uses the RV’s historic state information to derive safe control actions under the attack. However, these techniques assume that all the sensors in the RV are under attack (i.e., a worst-case recovery scenario) regardless of the number of sensors actually attacked. As a result, the recovery technique performs either overly conservative or overly aggressive recovery actions during the RV mission that leads to unnecessary mission disruptions and delays (shown later in the paper). This is because, under the worst-case assumptions, prior recovery techniques isolate all the sensors and use historic states to derive recovery control actions. Thus, they lack real-time sensor feedback and accurate information about the RV’s current state. Attack recovery must ensure minimal deviations from the set path and ensure timely recovery without delaying critical tasks.

Furthemore, attackers can launch stealthy attacks on RVs, causing small disruptions that build up over time [20]. While techniques have been proposed to detect such stealthy attacks [22, 38], they incur a delay in detection due to the controlled nature of these attacks. As attack recovery techniques [31, 45, 46] rely on the RV’s historic states information, the detection delay degrades the effectiveness of attack recovery. This is because the RV may have significantly deviated from the desired path before the stealthy attack is flagged.

The above issues arise from the lack of attack diagnosis, and a unified method to attack detection and recovery in RVs. To secure RVs from SDAs, we need to: (i) reliably identify sensors targeted by the SDA and selectively remove them from the RV’s feedback control loop, and (ii) bridge the disconnect between attack detection and recovery to prevent the impact of detection delay on recovery.
In this paper, we propose DeLorean, a unified framework for attack detection, diagnosis, and recovery from SDAs in RVs. We propose a graph-based probabilistic attack diagnosis technique to identify the sensors targeted by the SDA (i.e., attack diagnosis), and integrate the diagnosis with existing attack recovery techniques \cite{45, 46}. This eliminates the worst-case assumptions in recovery techniques and enables targeted attack recovery. Finally, we design methods to integrate existing attack detection \cite{19, 22, 38} with recovery techniques to address the attack detection delay, which ensures safe recovery of the RV even from stealthy attacks.

DeLorean assumes there is an existing attack detection technique deployed on the RV \cite{19, 22, 38}. After the attack detector raises an alert, DeLorean activates attack diagnosis to reason about the abnormalities in the RV’s physical states under an SDA, identifies the sensors targeted by the attack, and selectively removes those sensors from the RV’s feedback control loop. As RVs use sensor fusion to derive state estimates, it is challenging to attribute abnormalities in physical states to a specific sensor being under attack. We address this challenge by monitoring error inflation in all physical state estimates derived from the RV’s sensors, and perform causal analysis using factor graphs \cite{15} to identify targeted sensors. The attack diagnosis is independent of the analysis performed by the attack detection techniques \cite{19, 22, 38}. This helps mask false positives in the attack detection, as it is unlikely for both the detection and diagnosis techniques to incur false positives simultaneously.

DeLorean also introduces an automated method to record attack-free historic state information of the RV while accounting for the attack detection delay. Based on attack diagnosis outcomes, DeLorean estimates the physical states corresponding to the compromised sensors using the historic state information, and incorporates the uncompromised sensor estimates to derive reliable RV state estimates despite the attack. Finally, the attack recovery technique uses these reliable state estimates to perform attack recovery.

By unifying attack detection, diagnosis, and recovery, DeLorean provides a targeted approach to recover RVs from both single and multi-sensor SDAs including stealthy attacks. To the best of our knowledge, DeLorean is the first attack diagnosis-guided attack recovery technique for RVs. Our contributions are as follows:

- Propose a graph-based probabilistic diagnosis approach to identify the RV’s sensors targeted by SDAs through causal analysis, independent of the attack detection.
- Propose automated methods to record attack-free historic state information, and state reconstruction strategies that enable existing attack recovery techniques \cite{31, 45, 46} to perform targeted attack recovery under SDAs.
- Design DeLorean, a unified framework that integrates existing attack detection and attack recovery techniques with the proposed attack diagnosis technique, to provide diagnosis-guided attack recovery from SDAs.
- Evaluate DeLorean on six RVs - four real RVs and two simulated systems in a wide range of scenarios, and under SDAs targeting different numbers of sensors in the RV.

Our experimental results are as follows: (1) DeLorean’s attack diagnosis under multi-sensor SDAs achieves a 96% true positive (TP) rate and a 5% false positive (FP) rate. (2) DeLorean effectively masks FPs from the attack detector, leading to a 4X reduction in gratuitous recovery activations in contrast to three other baseline diagnosis approaches. (3) DeLorean successfully recovers RVs from multi-sensor SDAs in 93% of cases, resulting in over 2X reduction in the RV’s deviations from the set path, and a 2.5X reduction in mission delays compared to a worst-case recovery technique. (4) DeLorean also mitigates stealthy attacks and achieves 100% mission success under such attacks, and (5) DeLorean incurs modest overheads in both runtime and memory when executed on real RVs, and is able to recover them from SDAs without incurring any crashes.

2 BACKGROUND AND THREAT MODEL

In this section, we present an RV’s state estimation process. Then, we present sensor deception attacks (SDA), and the threat model.

2.1 State Estimation and Control in RVs

An RV’s state estimation and control process operates in a feedback loop. It uses the sensor measurements to determine the current physical state of the vehicle, and derive actuator signals for positioning the vehicle in the next state. For example, GPS measures the RV’s position, gyroscope measures angular velocity, accelerometer measures velocity and acceleration, magnetometer measures the heading direction, and barometer measures the altitude. In addition, RVs use sensor fusion techniques like Extended Kalman Filter (EKF) \cite{34} to enhance the physical state estimations by fusing measurements from sensors with a model of the systems’ dynamics. Typically, a PID (Proportional Integral Derivative) controller is used for the RV’s position, velocity, and orientation control.

![State Estimation and Feedback Control in RVs. SDA influencing RV’s state estimation and control.](image)

2.2 Sensor Deception Attacks

Physical attacks that manipulate sensor measurements from an external source via physical channels have been demonstrated in RVs. e.g., gyroscope measurements can be manipulated through acoustic noise injection \cite{41, 43}, and GPS measurements can be manipulated by transmitting false GPS signals \cite{29, 37}. Such attacks are launched by injecting false data to sensor \cite{19, 20, 23, 38}.

In this paper, we consider a strong form of physical attacks against RVs - we call these Sensor Deception Attacks (SDA) – in which, one or more sensors in the RV are compromised simultaneously. Any possible combination of sensors can be targeted by an SDA, even up to all the sensors. An SDA is launched by injecting malicious signals to the RV’s sensors. For example, an adversary can set up a fake GPS emitter, acoustic signal injectors, and magnetic signal injectors at a geographical location, and target the GPS, gyroscope, accelerometer, and magnetometer sensors simultaneously during an RV mission. Such an SDA can be debilitating for
We focus on physical attacks that maliciously perturb one or more of the RVs, as it simultaneously corrupts the physical properties measured by different types of sensors on the RV (e.g., position, velocity, acceleration, angular velocity, heading direction etc.).

SDAs targeting multiple sensor types have been demonstrated against bare metal systems [36] and self-driving cars [16]. For drones, Nashimoto et al. [36] showed that an adversary can target gyroscope, accelerometer, and magnetometer sensors at once through acoustic and magnetic signal injection to influence state estimation even in the presence of sensor fusion (EKF). We later show that such an attack can crash a drone almost immediately.

Attackers can also exploit knowledge of an RV’s attack detection strategy to launch stealthy attacks [20]. Stealthy attacks inject controlled sensor manipulations while evading detection, and can cause significant disruptions when performed over long durations [21].

### 3 RELATED WORK AND MOTIVATION

In this section, we first present the related work and highlight its limitations. Then, we present a motivating example for introducing attack diagnosis guided targeted attack recovery from SDAs.

#### 3.1 Related Work

**Attack Detection.** Many attack detection techniques have been proposed for RVs [19, 22, 30, 38, 44], and other cyber-physical systems (CPS) [13, 17, 25] that detect attacks based on either invariants or model estimations. However, all of these techniques focus only on attack detection which, by itself, is not enough for securing RVs.

All three components attack detection, diagnosis, and recovery are necessary for effectively mitigating SDAs against RVs.

**Attack Recovery.** We classify prior work in attack recovery on RVs into two categories as follows: (1) Model-based recovery, and (2) Checkpoint-based recovery techniques.

Model-based recovery techniques rely on a model of the RV to derive corrected sensor or actuator signals to recover RVs from attacks. For example, Software Sensor-based Recovery (SSR) [18] emulates the physical sensor signals using virtual sensors, and derives recovery control actions. Another example is PID-Piper [22], which uses a machine learning (ML) based feed-forward controller (FFC) to derive robust actuator signals under attacks.

On the other hand, Checkpoint-based recovery techniques isolate sensors upon attack detection, and instead use the historic trustworthy states of the system to determine robust control actions and recovery RVs. Kong et al. [31] use the historic states to roll forward the RV to expected states under attack. Zhang et al. use a linear programming-based recovery controller [45], and generate recovery control actions using historic states for bringing a system under attack back to a set target. In subsequent work, they proposed a linear quadratic regulator (LQR) based recovery controller [46] which improves the recovery control actions and ensures that the system remains in a safe state after recovery.

### Limitations in Existing Work and Research Gap

A fundamental issue with Model-based approaches [18, 22] is that they are based on a predictive model of the RV, and hence they tolerate an error margin (approximation error) between model estimations and the RV’s real behavior. The error margin is small enough under single sensor attacks to derive robust control actions for recovering the RV. However, when multiple sensors are attacked simultaneously, the model’s approximation error increases. Thus, both SSR and PID-Piper are ineffective under multi-sensor attacks.

In contrast, Checkpoint-based techniques derive recovery control actions aimed at restoring the RV to a predefined state, instead of merely tolerating attack-induced manipulations. Thus, they prove superior in safeguarding RVs from attacks. However, Checkpoint-based recovery methods assume all the sensors are targeted simultaneously [31, 45, 46] and perform a worst-case recovery irrespective of the actual number of sensors targeted. We show later that the worst-case assumptions lead to overly conservative or aggressive maneuvers of the RV which result in disruptions and a failed recovery, when only a subset of sensors are attacked.

**Scope of this Paper.** Our goal is to bridge the gap between Model-based and Checkpoint-based techniques by enabling attack recovery methods to handle single or multi-sensor SDAs, even those that target a subset of sensors. To achieve this, we propose an attack diagnosis technique that integrates seamlessly with existing attack detection [22, 38] and recovery techniques [46]. Upon attack detection, our proposed attack diagnosis identifies the targeted sensors and isolates them from the RV’s feedback control loop. Subsequently, recovery control actions are generated corresponding to the compromised sensors, eliminating the worst-case recovery assumptions when only a subset of sensors is targeted.

Further, to address attack detection delays caused by stealthy attacks which can remain undetected for a prolonged duration and impact recovery process, we design a method to determine the maximum duration a stealthy attack can remain undetected. We then synchronize attack detection, diagnosis, and recovery techniques in a unified system to prevent disruptions due to stealthy attacks.

#### 3.2 Motivation

We performed an experiment and launched SDAs targeting a subset of the RVs sensors to highlight the gaps in the current Checkpoint-based attack recovery techniques. The details of the experimental setup are explained later (Section 5). We used the state-of-the-art Checkpoint-based recovery technique, which employs a Linear...
Quadratic Regulator (LQR) [46] for deriving recovery control actions i.e., corrective maneuvers for safeguarding the RV. Henceforth, we refer to this method as LQR-O. Recall that when recovery is activated LQR-O isolates all the sensors from the RV’s feedback control loop, and assumes a worst case recovery scenario.

In this experiment, we used a Pixhawk drone (details in Section 5), and we launched SDAs targeting the RV’s GPS and accelerometer sensors. We deliberately target two out of the five sensors of the drone to study LQR-O’s recovery when only a subset of sensors is targeted by SDAs. We assume the drone has an attack detector [22] that raises an alert upon an SDA, activating attack recovery. Figure 2 shows the drone with LQR-O’s recovery under the SDAs.

![Figure 2: LQR-O’s recovery from SDA targeting 2 sensors simultaneously. The bottom graph shows the SDA duration. Two instances of SDAs launched from t=5-30s and t=65-95s.](image)

The first instance of SDA was launched between 5 to 30s, during the takeoff phase of the RV mission. As LQR-O assumes a worst-case recovery it disengages the inputs from all the sensors. Then, using the historic states it derives recovery control actions to set the RV’s altitude at $10\text{m}$ as per the mission trajectory. However, due to the absence of real-time sensor feedback, LQR-O erroneously believes the RV remains below the 10-meter target altitude. Hence, LQR-O triggers an aggressive maneuver, propelling the RV to an elevation of 18 meters, a substantial deviation from its expected trajectory. While LQR-O eventually steers the drone back on course, it incurs significant deviations and prolongs the mission by 15s.

The second instance of SDA was launched from t=65 to 95s, during the landing phase of the RV mission. LQR-O again performs a worst-case recovery, this time exhibiting an overly conservative approach in deriving recovery control actions. This is because LQR-O lacks information regarding the rate at which the RV is descending due to the absence of real-time sensor feedback. Consequently, LQR-O extended the RV’s landing duration by approximately 20 seconds. Furthermore, it resulted in an imprecise landing with the RV touching down 15m away from the set destination, which is a deviation much larger than the standard GPS offset of $\approx 5\text{m}$.

**Why attack diagnosis guided targeted attack recovery?** The above experiment highlights the limitations of the state-of-the-art Checkpoint-based recovery technique that undertakes a worst-case recovery strategy, irrespective of the actual number of sensors targeted by an SDA. This results in overly conservative or aggressive maneuvers, which is particularly undesirable during critical phases like drone takeoff or landing, and can lead to mission disruptions or failure. Recall that Checkpoint-based techniques operate under the assumption that all the sensors are under attack regardless of the actual number of sensors targeted. Due to the worst case assumptions, Checkpoint-based recovery techniques fail to leverage feedback from the uncompromised sensors, thereby limiting their ability to improve recovery control actions. While worst-case recovery outcomes may be unavoidable if an SDA targets all the sensors, targeted attack recovery can prevent unnecessary recovery failures when the attack targets only a single sensor or a subset of sensors.

Our goal is to enhance the effectiveness of Checkpoint-based techniques by introducing targeted attack recovery strategies guided by attack diagnosis. Our approach eliminates the need for worst-case assumptions, and enables deriving recovery actions corresponding to the sensors targeted by an SDA.

**Why a unified framework?** Checkpoint-based techniques heavily rely on the RV’s historical behavior to estimate safe recovery control actions. However, the persistence of attack detection delay poses a significant challenge as it may lead to erroneous recovery actions. This can be addressed by integrating attack detection and recovery techniques, and ensuring that attack-free historical information is used for deriving recovery actions. Furthermore, multi-sensor attacks introduce significant challenges for existing Checkpoint-based recovery techniques. Without attack diagnosis, existing recovery techniques are confined to worst-case assumptions. The integration of attack diagnosis with attack recovery enables targeted recovery actions, improving recovery success rates.

### 4 DESIGN

Figure 3 shows the DeLorean framework running onboard the RV. DeLorean uses an existing attack detector to detect attacks, and performs diagnosis to identify sensors targeted by the attack. There are many existing attack detection techniques for RVs [19, 22, 38].

![Figure 3: Sequence of operations in attack detection, diagnosis, and recovery. These operations run onboard the RV.](image)

DeLorean activates diagnosis when the attack detector indicates an attack. After identifying targeted sensors through diagnosis,
Diagnosis-guided Attack Recovery for Securing Robotic Vehicles from Sensor Deception Attacks

\[ P(S_i|e_i) = \frac{P(e_i|S_i)P(S_i)}{P(e_i)} \] (1)

We observe that in attack-free segments of the mission, the error between the RV’s past and present states is relatively low i.e., \( e_i = |s_{state_i} - s_{state_i-1}| > \delta \) and \( e_{i+1} = |s_{state_i+1} - s_{state_i}| < \delta \), where \( i \in \text{PS} \) (all the physical states of the RV, shown in Equation 1) However, under attack, the error \( e_i \) increases i.e., \( e_i \gg \delta \) [19, 22, 38]. Therefore, our approach for attack diagnosis is to monitor the inflation in error \( e_i \) in four consecutive time steps and find the probability that the corresponding sensor is under attack given the observed errors (including both steady and non-steady state errors)

\[ PS = x, y, z, \dot{x}, \dot{y}, \dot{z}, \phi, \theta, \psi, \omega_x, \omega_y, \omega_z, x_m, y_m, z_m \] (1)

However, error inflation in the RV’s physical states can also occur due to the RV’s physical dynamics in the mission (e.g., mode changes), and not only due to attacks. Thus, it is important to find the root cause of the error inflation in RV’s physical states. We use factor graphs (FG) to monitor the error inflation in physical state. FGs are probabilistic graphical models that allow expressing conditional relationships between variables [15]. In particular, we use FG to represent the causal relationship between the observed error \( e_i \) and find the probability that the corresponding sensor is under attack i.e., the outcome \( s_i \). We consider binary outcomes for the sensors i.e., malicious (targeted by SDA) or benign (attack-free).

The causal relationship between \( e_i \) and \( s_i \) can be expressed as a conditional probability problem i.e., \( P(s_i|e_i) \). There are two ways to calculate conditional probability: (1) Using a joint probability distribution over the observed \( e_i \) and the probable outcomes \( s_i \) (i.e., benign or malicious sensors). However, expressing the joint probability distribution at runtime for binary outcomes will have high computation and storage overheads. Further, as the mission progresses, the time taken to calculate probabilities will increase exponentially. Therefore, this approach is not practical.

(2) Using a Bayesian approach, we can calculate \( P(s_i|e_i) \) based on the values of known probabilities of \( P(e_i|s_i = \text{malicious}) \) and \( P(e_i|s_i = \text{benign}) \). To calculate \( P(s_i|e_i) \), we will need to run a large number of experiments and profile the observed error \( e_i \) in the RV’s physical states under attacks as well as in the absence of attacks to calculate \( P(e_i|s_i = \text{malicious}) \) and \( P(e_i|s_i = \text{benign}) \). This will also not work in our case due to two reasons: First, it is difficult to know how many observations are enough. Second, if we observe an error during inference that was not observed during profiling, the Bayesian approach will assign it a zero probability (i.e., Zero Frequency problem). Thus, even if a sensor is under attack, the Bayesian approach will wrongly conclude the sensor is benign.

In contrast to the above approaches, FGs allow fine-grained representation of the complex joint probability distribution between variables through a product of smaller probability distributions. Thus, they incur low computation and storage overheads. Further, instead of relying on profiling, FGs use functions (i.e., factor functions) to express conditional relationships. Thus, FGs can accommodate unseen observations \( (e_i) \), and do not suffer from the Zero Frequency problem. This is why we use FGs for attack diagnosis.

Figure 5 shows an example of FG-based diagnosis. The FG represents the relationships between error \( E = (e_{i_1},..,e_{i_n}) \) and the probable outcome \( S = (s_1,..,s_n) \) using a bipartite graph, where \( e_i \) and \( s_j \) are observed error and outcomes of the RV’s physical states (Equation 1). Factor functions \( F = (f_{i_1}...,f_{i_n}) \) define relationships between observed errors \( (e_{i_1},..,e_{i_n}) \) and probable outcomes \( s_i \). Using

**Table 1: Physical states and corresponding sensors in RVs**

| Sensors | GPS | Gyroscope | Accel | Barometer | Magnetometer |
|---------|-----|-----------|-------|-----------|--------------|
| States | x, y, z | \( \phi, \theta, \psi \) | \( \omega_x, \omega_y, \omega_z \) | \( \dot{x}, \dot{y}, \dot{z} \) | \( x_m, y_m, z_m \) |
FGs, we calculate the probability of each physical state $s_i$ being malicious given the observed error $e_i$, i.e., $P(s_i = \text{malicious}|e_i)$. If certain physical states are found to be malicious, we deem the corresponding sensors (Table 1) to be under attack.

\[ P(s_i = \text{malicious}|e_i) \]

Thus, at each time $t$, we feed the physical states derived from sensor targeted sensors. We calculate the probability of physical states $PS$ (Equation 1) given the observed errors $e_i$. Because the observed $e_i$ inflates under attacks, the $P(s_i = \text{malicious}) > P(s_i = \text{benign})$. We deem those states to be malicious where $P(s_i = \text{malicious}) > 0.5$, and deem the corresponding sensors (Table 1) as being under attack.

\[ P(s_i = \text{malicious}|e_i) \]

4.2 Historic States Checkpointing

When the RV is in an attack free phase, we record the sequence of physical states estimated from all the on-board sensors in a sliding window. Figure 6a shows the details. The physical states at time $t$ are represented as $x(t)$, and the historic physical states recorded in a window are represented as $HS$ (states shown in Equation 1).

From the start of the RV’s mission, we record $HS$ in a sliding window of length $w$. If no alert is raised by the attack detector, we save the $HS$ recorded in window $w_t$ and proceed recording states in the next window $w_{t+1}$ as shown in Figure 6a. At the end of window $w_{t+1}$, we discard the $HS$ recorded in the previous window $w_t$. However, if we detect an attack as shown in Figure 6b, we stop the recording, and discard the states recorded in the current window $w_t$ as these may be corrupted by the attack. We use the $HS$ recorded in the previous window $w_{t-1}$ for state reconstruction and recovery. These $HS$ are attack-free as no alert was raised by the attack detector in window $w_{t-1}$.

\[ P(s_i = \text{malicious}|e_i) \]

Another challenge is that sensors in RVs are sampled at different rates e.g., the sampling rate for gyroscope is 400 Hz, whereas, the sampling rate for Barometer is 100 Hz. It is important to align the data streams when recording the historic states. To address this problem, we select a single target frequency for recording the $HS$, which is the highest sampling rate of all the sensors. We then align the low frequency streams with the high frequency streams by inserting additional data points in the low frequency stream to match the sample points in the high frequency streams. In particular, we duplicate the last data point in the low frequency streams based on the ranges of the sample points of the high frequency streams.

\[ P(s_i = \text{malicious}|e_i) \]

\[ P(s_i = \text{malicious}|e_i) \]

\[ P(s_i = \text{malicious}|e_i) \]

\[ P(s_i = \text{malicious}|e_i) \]
diagnosis i.e., the time interval between the onset of the stealthy attack and the time when the attack detector raises an alert. Due to the detection delay, DeLorean might record corrupted HS for recovery, which it incorrectly deemed safe as no alarms were raised by the attack detector. Therefore, to handle stealthy attacks, we empirically choose the window length to be large enough so that stealthy attacks can be detected within a single sliding window.

4.3 State Reconstruction

After identifying targeted sensors through attack diagnosis, we reconstruct the RV’s state vector represented as $X'(t)$. The goal of state reconstruction is to ensure accurate current state representation for the sensors affected by SDAs (referred to as $x_r(t)$), while preserving the state derived from sensors that remained unaffected by the attack at the time when recovery is activated. Furthermore, the state reconstruction process plays a key role in enabling targeted attack recovery as it facilitates real-time sensor feedback (if available i.e., when SDAs target only a subset of the sensors).

We use the Extended Kalman Filter (EKF) [34] to estimate $x_r(t)$ using the latest trustworthy historic states under SDAs. The EKF estimator is constructed offline using the RV’s non-linear system dynamics (Appendix A.2). The parameters of the non-linear equations are learned using a dataset of control actions and sensor measurements. We collect the data using our subject RVs (Section 5). We learn an approximate model of the RV $f(x_r, u)$ through system identification [10]. The model parameters are optimized using the least squared error approach, i.e., minimize squared error between the model’s estimations and the observed values.

Taking the scenarios in Figure 6b as an example, the latest trustworthy states $x_{t_k}$ are recorded at time $t_k$. The RV’s states evolve over time as per its system’s dynamics [34]. For example, at time $t_{k+1}$ the RV’s states are calculated as $x_r(t_{k+1}) = f(x_{t_k}, u_t) + o_t$, where $u_t$ and $o_t$ are control inputs and process noise respectively. The above step is iteratively performed for each step time between $t_k$ and $t_a$ (i.e., $t_{k+1}, t_{k+2}, ..., t_a$) to estimate RV’s states $x_r(t_a)$. The EKF relies on system’s dynamics and trustworthy historic states instead of compromised sensor measurements.

We reconstruct the RV’s state vector at $t_a$ when recovery is activated as follows. For sensors that remain uncompromised under SDA, we use the states derived from the corresponding sensors, these states are represented as $x_{r}(t_a) = x_{r_1}, ..., x_{r_n}$. We use the trustworthy historic states to estimate states corresponding to the sensors that are targeted by the SDA, these states are represented as $x_r(t_a) = x_{r_1}, ..., x_{r_n}$. We reconstruct RV’s state vector by selectively combining the states derived from uncompromised sensors and estimated states for the compromised sensors as $X'(t_a) = [x_r(t_a), x_r(t_a)]$ to obtain accurate current state representations. $X'(t_a)$ is the initial system state of recovery.

5 EXPERIMENTAL SETUP AND PARAMETERS

In this section, we present our experimental setup, metrics, and attack parameters for evaluating DeLorean. We then present DeLorean’s parameters derived from our experiments.

Table 2: Subject RVs in Evaluation, Number of Sensors, Attack parameters.

| Sensor Type       | P | T | S | AC | R1 | AR | Bias Values | Max Range |
|------------------|---|---|---|----|----|----|-------------|-----------|
| GPS              | 1 | 1 | 1 | 1  | 1  | 1  | 5-50m       | 200       |
| Gyroscope        | 3 | 3 | 3 | 3  | 3  | 3  | 0.5-9.47 rad| 100       |
| Accelerometer    | 3 | 3 | 1 | 3  | 3  | 3  | 0.5-6.2 rad/s2 | 26        |
| Magnetometer     | 3 | 3 | 1 | 3  | 3  | 3  | 180 deg     | -         |
| Barometer        | 1 | 2 | 1 | 1  | 1  | 1  | 0.1 kPa     | -         |

5.1 Comparison with Prior Work

Attack Recovery: We quantitatively compare DeLorean with three prior attack recovery techniques discussed earlier namely SSR [18], PID-Piper [22], and LQR-O [46]. Recall that LQR-O refers to the recovery technique proposed by Zhang et al. which performs worst-case recovery irrespective of the number of sensors targeted under SDA. We collected mission traces of RVs, including both training and testing sets, and assessed the efficacy of existing recovery techniques by reproducing their effectiveness as reported in the respective papers. Subsequently, we evaluate all three recovery techniques under SDAs to observe their effectiveness under multi-sensor SDAs targeting different numbers of sensors in the RV.
Attack Diagnosis: There is no prior work that addresses attack diagnosis in RVs (to the best of our knowledge). We extend two existing model-based attack detection techniques namely Savior [38] and PID-Piper [22], to serve as a potential solution for attack diagnosis. Model-based attack detectors estimate RV’s physical states and compare them with the physical states derived from sensors to compute a residue $r = |Model.states - Sensor.states|$. Traditionally, these attack detectors analyze residues in just one or two of the RV’s physical states, sufficient for flagging an attack. We extend the concept of residual analysis (RA) to monitor all the physical states in RVs for attack diagnosis. Furthermore, RV’s are equipped with EKF that estimates the RV’s physical states, and uses sensor measurements to enhance state estimations (Section 2.1). EKF estimations can be potentially used for attack detection and attack diagnosis by performing residual analysis to monitor EKF estimations and sensor states. When a residue for a specific state exceeds the threshold, the corresponding sensor is deemed to be under attack (Table 1 shows states to sensor mapping). We compare DeLorean with the above three baseline RA-based diagnosis techniques.

5.2 Metrics

Recovery Success Metric: As in our prior work [22], we consider a mission to be successful, if upon completion, the total deviation from the original destination is less than 10m. Most GPS sensors used in commodity RVs have an offset of 5 meters [40]. We consider 2X of the GPS offset as our threshold i.e., 5m offset from the RV’s position, and 5m offset from the destination. This 10m threshold is indistinguishable from the standard GPS error [40]. We consider the mission to be unsuccessful if the RV crashes (physically damaged) or stalls (freezes or stops moving towards the destination).

Recovery Stability Metric: To measure the stability of the RV during recovery we use the Root Mean Square Deviation (RMSD) of the RV’s attitude i.e., roll, pitch, and yaw angles. This metric quantifies the average magnitude of deviations in a recovery-activated mission compared to an attack-free (ground truth) mission on the same trajectory. We calculate RMSD using the same method as Root Mean Square Error, as depicted in Equation 5. As minor attitude deviations between two RV missions even on the same trajectory is expected, we account for these expected deviations by normalizing RMSD with respect to the minimum and maximum expected deviations for an RV on a specific trajectory. (details in Appendix A.5).

$$RMSD = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_{\text{recovery}} - x_{\text{groundTruth}})^2}$$

5.3 Attack Parameters

Attacks: As we did not have access to special equipment (e.g., noise emitter, sound source, amplifier, etc.) for mounting physical attacks, we emulated the attacks through targeted software modifications, similar to what most prior work has done [18, 19, 22, 38]. Our attack code interfaces with the sensor libraries in the RV, and manipulates sensor measurements by adding a bias to them (i.e., false data). When an attack command is initiated, the bias values are automatically added to the raw sensor measurements. We derive the attack parameters i.e., bias values and attack range to mimic realistic physical attacks via software as closely as possible.

Sensor Bias Values: We use variable bias values for each sensor within the allowable limit to cause mission delays, instability, and even mission failure. We derive bias values as per the respective sensor specifications. Table 2 shows the bias values used for each sensor. For example, the update frequency of the GPS module used in many industrial and commodity RVs is 0.1s, and the operational limit in its velocity is 500 m/s [12]. Therefore, the maximum hopping distance of the GPS receiver is 50m (update frequency × maximum velocity). Thus, for GPS, we set the bias value to be between 5-50m, which is the operating limit of the GPS sensor.

Attack Range: We derive the maximum attack range based on prior work that performed the respective attacks through signal injection [36, 37, 42]. We extrapolate their results considering the largest available signal source and amplifier to derive the maximum attack range for each sensor. We found that the optimal attack range for individual sensors is within 20 – 200m. Of all the sensors, the GPS has the maximum attack range [42] of 200m. Thus, we select 200m as the attack range for each instance of SDAs in our experiments i.e., we assume that attackers can manipulate all or any subset of the RV’s sensors in a 200m range. This is a stronger assumption than that made in all the prior work in the area [41, 43].

5.4 DeLorean Modules and Parameters

Attack Detector: As mentioned, DeLorean relies on an attack detection technique, and it provides attack diagnosis and targeted attack recovery after the attack is detected. For attack detection, we use our prior work, PID-Piper’s attack detection module [8], as it represents the state-of-the-art and has a high detection rate. We do not measure the effectiveness of the attack detector (true positives, and false positives) - this is reported in our prior work [22].

Runtime Overheads: We measure the overheads in CPU times, battery consumption, and memory incurred by the RV due to DeLorean. As the overheads in simulated RVs depend on the simulation platform, we report the overheads only for the real RVs. For CPU overheads, we measure the CPU times incurred by the autopilot modules with and without DeLorean. During the recovery process, the RV executes the prescribed recovery control actions in place of the actions generated by the PID controller. As a result, the RV’s motor rotation rates increase. We call this operational overhead. We estimate the battery overhead of DeLorean based on both the CPU overhead and the operational overhead. Finally, we estimate the memory overhead for storing the historic states on the RV corresponding to all the RV’s sensors for targeted attack recovery.
Attack Diagnosis: The attack diagnosis module monitors the error \( e \) between the past and present states of the RV. Recall that in the attack-free phase, \( e \) remains within \( 0 - \delta \). This holds true regardless of whether the RV is in a steady or non-steady state. We use the standard deviation method to derive the value of \( \delta \), i.e., \( \delta = \text{median}(e) + k * \text{std}(e) \) i.e., if the error \( e \) is more than \( k \) standard deviations away from the mean, it is likely to be an outlier (due to the attack) [39]. We collect attack-free RV mission traces (on both simulated and real RVs with diverse mission trajectories), and empirically determine that \( k = 3 \) ensures \( 0 < e < \delta \) in the attack-free phase of the mission. We run between 15-25 attack-free missions for each RV to derive the \( \delta \) values, and we validated \( \delta \) values (i.e., \( e \in 0 - \delta \)) by running another 15 missions.

Figure 8: (a) CDF of error distribution in the Pixhawk drone. \( k = 3 \) ensures \( 0 < e < \delta \) in attack-free phase. (b) CDF of the likelihood of stealthy attack detection. Window size of 15s (Tarrot) and 18.5s (Aion A1) ensures 100% attack detection.

Figure 8a shows an example of \( \delta \) for position error in the Pixhawk drone. The figure shows the Cumulative Distribution Function (CDF) of error \( e \) as a function of the probability that the physical states errors will take a value less than \( \delta \). With \( k = 3 \), we obtain the \( \delta \) value for z-axis position error as 5.2. As shown in Figure 8a, in the attack-free phase, the error \( e \) always remains under the \( \delta \). Table 3 shows the \( \delta \) values for all the physical states of our subject RVs.

Attack Recovery under Stealthy Attacks: Recall that we prevent disruptions due to stealthy attacks by setting the window size for recording historic states to be large enough that stealthy attacks can be detected in a single sliding window (Section 4.2). The key question is what is the maximum duration that the attacker can cause disruptions in the RV mission by launching stealthy attacks while evading detection. Subsequently, we can set an appropriate window size for the effective detection of stealthy attacks. Therefore, to find the appropriate window size, we launched two kinds of stealthy attacks by injecting persistent bias: (1) targeting all the sensors simultaneously, (2) targeting each sensor individually.

Note that the above stealthy attacks were launched on all of our subject RVs, in the presence of PID-Piper’s attack detector [22] that uses CUSUM to detect stealthy attacks [20]. We find that when all the sensors are manipulated persistently, the stealthy attack is detected in under 3.3s. However, when the attack targets individual sensors, the stealthy attacks against GPS remain undetected for the maximum duration i.e., around 15 – 18s.

We determine the window size for each RV to be larger than the duration of a stealthy attack that targets GPS. Figure 8b shows a CDF that depicts for how long a stealthy attack targeting GPS can evade detection as the mission progresses. 100% attack detection rate is achieved for Tarrot drone at 15s, and for Aion R1 rover at 18.5s. Thus, we select window sizes of 15s and 18.5s for the respective RVs, as the likelihood of detecting stealthy attacks in that window is near 100%. Table 2 shows the window sizes for all RVs derived in the same manner.

## 6 RESULTS

We first evaluate DeLorean’s attack diagnosis under SDAs in a simulated RV. Then, we evaluate DeLorean’s targeted attack recovery and compare it with other techniques under SDAs in the simulated RVs. Then we study the significance of attack diagnosis under multsensor SDAs, followed by DeLorean’s attack diagnosis and recovery in real RVs. We also study DeLorean’s effectiveness under stealthy attacks. Finally, we present the overheads incurred by DeLorean.

### 6.1 Effectiveness of DeLorean’s Attack Diagnosis

In this section, we evaluate the effectiveness of DeLorean’s attack diagnosis by measuring the true positives (TP) and false positives (FP). TP is the fraction of times attack diagnosis correctly identified the sensors targeted by SDA. FP is the fraction of times attack diagnosis falsely identified sensors as attacked even in the absence of an attack. because no prior work has addressed attack diagnosis, we extended the residual analysis in existing attack detection techniques namely Savior [38], PID-Piper [22], and EKF [34], for attack diagnosis. We compare DeLorean’s diagnosis that uses factor graphs with techniques that use residual analysis (RA-based diagnosis).

We run 100 missions using simulated RVs and launched SDAs targeting up to \( n - 1 \) sensors of the RV. We launched the same attacks for all the diagnosis techniques to compare them. Table 4 shows the results. We find that, Savior, PID-Piper, and EKF based diagnosis techniques achieve on average 60%, 56%, and 62.5% TPs respectively. In contrast, DeLorean’s diagnosis correctly identified
the targeted sensors in 95% of the cases (on average). This is because DeLorean attributes manipulations in the RV’s physical states to a sensor being under attack, by performing causal analysis using factor graphs. In contrast, the RA-based techniques perform a naive attack diagnosis by simply comparing the residuals against a threshold, thus failing to achieve a high diagnosis TP rate. Thus, DeLorean’s diagnosis resulted in 40 to 50% higher TPs rate on average than RA-based diagnosis methods (Table 4). When SDAs targeting all the sensors were launched, all the techniques including DeLorean achieve a high diagnosis TP rate.

**False Positives and Gratuitous Recovery Activation.** Note that a false alarm in detection leads to an FP in attack diagnosis, and eventually results in gratuitous recovery activation. During gratuitous recovery, the attack recovery technique generates unnecessary recovery control actions, causing needless maneuvers that result in mission disruptions and delays. We evaluate DeLorean’s FPs in the absence of attacks to assess its gratuitous recovery activations.

Attack detection techniques raise false alarms when exposed to environmental disturbances such as wind. To measure FPs in attack diagnosis, we induce false alarms in attack detectors by simulating wind conditions. It is important to strike a balance between the attack detection accuracy and induced false alarms, and not overwhelm the attack detector with an excessive number of false alarms. Thus, we simulated a modest wind speed of 15 km/h [5].

Table 4: DeLorean’s attack diagnosis compared with residual analysis-based diagnosis of Savior, PID-Piper, and EKF. Values are in Percentage.

| Diagnosis FPs number of sensors targeted by SDA | Savior RA | PID-Piper RA | EKF RA | DeLorean |
|------------------------------------------------|----------|--------------|--------|----------|
| 1                                              | 65       | 65           | 75     | 100      |
| 2                                              | 55       | 50           | 60     | 100      |
| 3                                              | 50       | 50           | 55     | 90       |
| 4                                              | 70       | 60           | 60     | 90       |
| Average                                        | 60       | 56           | 62.5   | 95       |
| FPs (no attack)                                | 60       | 15           | 20     | 9        |

We run 20 missions in Arducopter for each technique, activating diagnosis when attack detectors raised false alarms. Table 4 shows the FPs in each technique. The corresponding attack detectors of each RA-based diagnosis technique - Savior, PID-Piper, and EKF reported false alarms in 20%, 15%, and 20% of the cases respectively. As RA-based diagnosis methods rely on attack detectors’ state estimations they also incurred 15-20% FPs. In contrast, DeLorean masked 15% false alarms from the attack detector and incurred only 5% FPs i.e., 4X lower FPs than other diagnosis techniques.

The RA-based diagnosis techniques resulted in significantly higher FPs because they monitor the physical states’ error only in the last step, and hence fail to account for the benign deviations in RV’s physical states. In contrast, DeLorean’s diagnosis monitors the sequence of errors in four previous steps (this is inherent to the FG approach, see Figure 5). Thus, FG captures the causality of manipulations in RV’s physical states to infer if the corresponding sensor is under attack, and achieves both high TPs and low FPs. Furthermore, RA-based diagnosis relies on the attack detector’s response, false alarms in detection propagate as FPs in diagnosis. DeLorean performs attack diagnosis independently, allowing it to mask FPs triggered during attack detection.

The FPs in attack diagnosis resulted in gratuitous recovery activation. We find that the gratuitous recovery operations due to FPs in RA-based diagnosis resulted in RMSD of 0.095 and mission delays of more than 10% (on average). In contrast DeLorean incurred almost negligible RMSD of 0.0045 which resulted in ≈ 1% mission delays. Thus, DeLorean achieves a significant reduction of 4X in gratuitous recovery activation and prevents unnecessary mission disruptions.

### 6.2 Effectiveness of DeLorean’s Attack Recovery

We compare DeLorean with SSR, PID-Piper, and LQR-O in recovering from the SDAs on the simulated RVs. We launched 100 missions for each simulated RV protected with SSR, PID-Piper, LQR-O, and DeLorean respectively. We mounted the same SDAs for all four techniques varying the number of sensor types targeted from 1 up to 5 (all sensors). Note that when we say a sensor is attacked, we mean that all the sensors of that type are attacked in the SDA.

Table 5: DeLorean’s recovery outcomes compared with SSR, PID-Piper, and LQR-O as a function of the number of sensors attacked. MS: Mission Success, values are in percentages.

| # of sensors targeted | SSR Crash | MS | PID-Piper Crash | MS | LQR-O Crash | MS | DeLorean Crash | MS |
|-----------------------|-----------|----|-----------------|----|-------------|----|---------------|----|
| 1                     | 20        | 64 | 0               | 100| 0           | 90 | 0             | 100|
| 2                     | 56        | 20 | 64              | 20 | 0           | 84 | 0             | 100|
| 3                     | 100       | 0  | 100             | 0  | 0           | 84 | 0             | 100|
| 4                     | 100       | 0  | 100             | 0  | 4           | 82 | 4             | 88 |
| 5                     | 100       | 0  | 100             | 0  | 4           | 82 | 4             | 82 |

Table 5 shows the mission outcomes of DeLorean, SSR, PID-Piper, and LQR-O under single and multi-sensor SDAs. From the table, SSR has a mission success rate of 64% for single sensor attacks. However, for attacks targeting two sensors, its mission success rate drops to 20% with a crash rate of 56%. PID-Piper, on the other hand, is 100% effective when only a single sensor is targeted. However, its mission success rate also drops to 20%, and its crash rate increases to 64% when two sensors are attacked. Furthermore, when 3 or more sensors are attacked, both SSR and PID-Piper incurred a 100% crash rate (0% success). Thus, Model-based recovery techniques SSR and PID-Piper incur crashes and mission failures for multi-sensor SDAs, and had average mission success rates of 17% and 24% respectively.

On the other hand, LQR-O has an average mission success rate of 86% with no crashes when 1 to 3 sensors are under attack. However, when 4 or more sensors are under attack, LQR-O’s average recovery success drops to 82%. Because we inject variable sensor bias within the allowable range, When 4 or more sensors are targeted simultaneously, the SDA triggers aggressive deviations from the intended path influenced by the injected bias values. This makes it challenging for LQR-O to derive safe recovery control actions.

In contrast, DeLorean achieves 100% mission success rate when fewer than 3 sensors are attacked, and no crashes. When 4 or more sensors are targeted, DeLorean also achieves 85% mission success and 4% crash rate (same as LQR-O). DeLorean achieves 94% mission success rate on average across all sensor numbers, which is ≈ 4X higher than the mission success rates of SSR, and PID-Piper, and 10% point increase compared to LQR-O. Thus, DeLorean is more resilient to multi-sensor SDAs than prior techniques.
6.3 Need for Attack Diagnosis

To understand if attack diagnosis is needed for recovery from SDAs, we compare DeLorean’s diagnosis guided targeted attack recovery with LQR-O under SDAs targeting various numbers of sensors. Note that LQR-O assumes a worst-case recovery scenario and applies recovery operations corresponding to all the sensors irrespective of the number of sensors targeted by an SDA.

In this experiment, we perform 100 missions each for DeLorean and LQR-O in simulated RVs, and launch SDAs targeting 1 to 5 sensors in each mission (all combinations). In addition to mission success rate and crash rate, we measure Root Mean Squared Deviation (RMSD), and Mission Delay (MD). RMSD provides a measure of the stability of the recovery process, and MD shows the impact of recovery on the expected mission duration. Lower RMSD and MD signifies the recovery has insignificant impact on the RV mission.

Table 6: Comparison between DeLorean and LQR-O (without diagnosis) under SDAs. RMSD: Root Mean Squared Deviation, MD: Mission Delay (shown as the percentage increase in mission duration), MS: Mission Success Rate.

| sensors targeted | LQR-O | DeLorean |
|------------------|-------|----------|
|                  | RMSD  | MD  | Crash | MS   | RMSD  | MD  | Crash | MS   |
| 1                | 0.6259 | 14.3 | 0     | 90   | 0.1256 | 2.41 | 0     | 100  |
| 2                | 0.6319 | 14.66| 0     | 84   | 0.1552 | 3.66 | 0     | 100  |
| 3                | 0.6414 | 14.46| 0     | 84   | 0.2733 | 7.75 | 0     | 100  |
| 4                | 0.6383 | 12.41| 4     | 82   | 0.5784 | 10.31| 4     | 88   |
| 5                | 0.6603 | 12.58| 4     | 82   | 0.6003 | 12.55| 4     | 82   |

Table 6 shows the comparison between DeLorean and LQR-O. As we have already discussed the mission success and crash rates for both techniques in Section 6.2, we do not repeat them here. We find that the RMSD values of LQR-O do not vary much across SDAs that target varying numbers of sensors. In comparison, DeLorean has 2.2X lower RMSD (on average) than LQR-O when SDAs target a subset of the RV’s sensors. Furthermore, when up to 3 sensors are targeted, DeLorean’s recovery incurs 2X to 5X less RMSD compared to LQR-O. This is because LQR-O applies the worst-case recovery strategy in all cases. Such recovery strategy results in LQR-O applying aggressive maneuvers when a subset of sensors is targeted. Unsurprisingly, both DeLorean and LQR-O have the same RMSD values when all the sensors are targeted as both techniques perform the same recovery operations in such a situation. Thus, attack diagnosis-guided attack recovery as done in DeLorean ensures minimal deviations under SDA targeting any combination of sensors.

Furthermore, we find that LQR-O recovery on average, incurs 2.5X higher mission delay than DeLorean when the SDA targets only a subset of the sensors. Recall that LQR-O recovery results in higher deviations from the trajectory compared to DeLorean, consequently LQR-O requires more time to restore the RV to its intended path. Further, as shown in Table 6, LQR-O’s mission delays are consistent for attacks targeting varying numbers of sensors. On the other hand, DeLorean’s mission delays for attacks targeting less than 3 sensors are significantly lower than that of LQR-O. When a single sensor is targeted LQR-O’s recovery increases the expected mission duration by 14.5%. In contrast, DeLorean increases the mission duration only by 2.4%. However, for attacks targeting 4 or 5 sensors simultaneously, DeLorean incurs similar mission delays as LQR-O.

Thus, DeLorean’s attack diagnosis guided attack recovery prevents unnecessary mission delays while recovering from multi-sensor SDAs.

6.4 Attack diagnosis and Recovery in Real RVs

We also evaluate DeLorean on the four real RVs under SDAs targeting different numbers of sensors. Table 7 shows DeLorean’s diagnosis and recovery outcomes. We find that DeLorean’s diagnosis achieves 100% TP rate when the SDA targets up to 2 sensors simultaneously. However, its TP rate drops to between 80% and 90% when the SDA targets 3 or more sensors. On average, DeLorean’s attack diagnosis TP rate is between 92% to 98% across all RVs. We also find that DeLorean’s diagnosis FP rate was between 2% and 6% across all RVs (not presented due to space constraints). These results are comparable to those on the simulated RVs (Section 6.1).

Table 7: DeLorean’s attack diagnosis and attack recovery in real RVs. All values shown in percentage.

| # of sensors targeted | Pixhawk | Tarrot | Sky-viper | Atom R1 |
|-----------------------|---------|--------|-----------|---------|
| 1                     | TP | MS   | TP | MS | TP | MS | TP | MS |
| 2                     | 100 | 100  | 100 | 100 | 100 | 100 | 100 | 100 |
| 3                     | 90  | 80   | 80  | 70   | 80  | 80  | 80  | 80  |
| 4                     | 90  | 80   | 80  | 80   | 80  | 80  | 80  | 80  |
| 5                     | 80  | 100  | 80  | 100  | 80  | 100 | 80  | 100 |
| Average               | 96  | 92   | 98  | 94   | 92  | 92  | 94  | 92  |

We find that DeLorean’s average mission success rate for SDAs targeting upto \( n − 1 \) of the RV’s sensors is 95.62%. Furthermore, DeLorean successfully recovered RVs in 100% of the cases when the SDA targets up to 3 sensors simultaneously. However, its mission success rate drops to 80% when SDAs target all the 5 sensors simultaneously. DeLorean achieves 92.5% average mission success rate across RVs. Furthermore, we observed no crashes in any of the real RV mission. Finally, even for the failed missions, the deviation from the target was very small (15.6m on average).

Figure 9: DeLorean’s recovery in the Pixhawk drone under SDAs. Bottom graph shows when the SDA is launched.

Figure 9 shows DeLorean’s recovery on the real Pixhawk drone when SDAs targeted GPS and accelerometer sensors simultaneously. This experiment closely resembles the one discussed in Section 3.2. We deliberately target 2 out of 5 sensors to compare with LQR-O’s recovery discussed in Section 3.2 and DeLorean’s targeted recovery. In this mission, the drone should attain an elevation of 10m and navigate in a straight line for 70s, and land when it reaches the destination. Two instances of SDAs were launched: (i) during the takeoff from 5s to 30s, (ii) during landing from 65s to 85s. Note that,
in a similar mission, shown earlier in Figure 2, LQR-O incurred an RMSE of 20.66, a mission delay of 18.75%, and resulted in a mission failure with the drone landing 15m off the target destination.

In contrast, when DeLorean is deployed in the Pixhawk drone, it achieved RMSD of 4.21, caused only 2.5% mission delay, and it completed the mission successfully landing the drone at the destination. As shown in Figure 9, DeLorean limited deviations from the expected trajectory to 1.5m under the first SDA during takeoff. DeLorean maneuvered the drone without incurring any additional delays and landed it safely under the second SDA. Therefore, DeLorean’s diagnosis and targeted recovery successfully navigated the drone toward its set target and completed the mission successfully.

6.5 Recovery under Stealthy Attacks
Recall that DeLorean sets a historic states recording window large enough that a stealthy attack that injects persistent bias can be detected in one sliding window using CUSUM [22]. An attacker who is aware of this design can launch adaptive stealthy attacks by introducing variations in the bias. The attacker’s goal is to prolong the stealthy attack duration while evading detection, corrupt the DeLorean’s recorded historic states, and consequently, affect DeLorean’s recovery process. We perform experiments on ArduCopter to evaluate DeLorean under three adaptive stealthy attacks. Instead of persistent bias, these adaptive attacks inject: (A1) random bias, (A2) gradually increasing bias, and (A3) intermittent bias.

Finally, we evaluated DeLorean under attack A3. We launched the attack intermittently following the RV’s takeoff. This attack also managed to evade detection but resulted in a 2.1m total deviation from the RV’s original path. Consequently, DeLorean recorded the corrupted historic states similar to the A2 scenario. We find that DeLorean successfully recovered the RV as the historic states corruptions exhibited negligible impact on the recovery process.

Thus, DeLorean ensures timely detection of a stealthy attack within one sliding window, thereby ensuring no corruption of the recovery process. Furthermore, DeLorean’s robustness to the historic states’ corruptions under adaptive stealthy attacks ensures successful recovery.

6.6 Overheads
Table 3 shows the overheads incurred by DeLorean on the real RVs. The CPU overhead varies from 5.5% to 9.2%, with an average of 7.5% across the four RVs. This is slightly higher than the CPU overhead incurred by SSR and PID-Piper, which are 6.9% and 6.35% respectively (on similar RV platforms). However, because LQR-O does not report CPU overhead, we estimate its overhead based on DeLorean’s overhead. As LQR-O does not perform any diagnosis unlike DeLorean, we subtract the cost of diagnosis from DeLorean’s overhead, which comes to an average 7% across the RVs.

Furthermore, DeLorean’s memory overhead is between 0.45 MB and 0.56 MB across the different RVs. This accounts for less than 3% overhead, as the memory available on the real RVs is over 20 MB.

Finally, under attacks, DeLorean incurs battery overheads between 14.4% and 22% across RVs (18.87% on average). This overhead is a result of recovery control actions and mission delays, causing the motors to remain operational for an extended period. In comparison, LQR-O incurs significantly higher battery overhead as it incurs 2.5X higher mission delays than DeLorean. Note that we did not measure the battery overheads for PID-Piper and SSR as they do not recover from most SDAs.

7 CONCLUSION
We presented DeLorean, a unified framework for attack detection, diagnosis, and recovery from multi-sensor SDAs in RVs. We proposed a factor graphs based attack diagnosis method that inspects the attack induced errors in the RV’s physical states and performs causal analysis to identify the sensors targeted by SDAs. DeLorean then isolates those sensors from RV’s feedback control loop to prevent erroneous control actions. DeLorean integrates existing attack detection and recovery techniques with our proposed attack diagnosis technique to provide targeted attack recovery in RVs. We evaluate DeLorean on four real and two simulated RVs. We find that DeLorean (1) recovers the RVs from SDAs, and achieves mission success in 93% of the cases, (2) minimizes mission disruptions during recovery by performing diagnosis guided targeted recovery, and (3) achieves mission success even against stealthy attacks, and (4) incurs modest performance, memory, and battery overheads.

ACKNOWLEDGEMENTS
This work was partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), and a Four Year Fellowship from UBC. We thank the reviewers of AsiaCCS’24 and our shepherd Dr. Jairo Giraldo for their helpful comments.
A RESEARCH METHODS

A.1 Joint Probability and Factor Graphs

The joint probability of variables $x_1, ..., x_n$ is described as $P(x_1, ..., x_n)$. Causal inference of an event given the observed variable is calculated as: $y = \text{argmax} P(y|x_1, ..., x_{n-1})$, which can be further expanded as the following.

$$P(y|x_1, ..., x_{n-1}) = \frac{P(x_1, ..., x_{n-1})}{\sum_y P(x_1, ..., x_{n-1}, y = o)} \tag{7}$$

Inference on a joint probability distribution $P(x_1, ..., x_n)$ requires $2^n$ storage for events with a binary outcome (e.g., malicious, benign), which is computationally expensive as well.

Factor graphs can be used to overcome this problem. A factor graph (FG) is a probabilistic graphical model that allows expressing joint probability as a product of smaller local functions [15]. Figure 11 shows an example. There are two types of nodes in a FG namely variables ($x_1, x_2, x_3$) and factor functions ($f_1, f_2, f_3$). Variables are used to quantitatively describe an event. A factor function is used to express relationships among variables.

![Factor Graph Example](image)

**Figure 11: An example of factor graph**

Let’s take Figure 11 as an example to understand how to calculate conditional probability using a factor graph. The variables $x_1, x_2, x_3 \in X$, and $y$ is the outcome. $P(x_1, x_2, x_3)$ can be factored as a product of $f_1, f_2, f_3$.

$$P(x_1, x_2, x_3) = f_1(x_1), f_2(x_1, x_2), f_3(x_2, x_3) \tag{8}$$

Thus, the conditional probability of an event given the observed variables is calculated as:

$$P(y|x_1, x_2, x_3) = \prod_{f \in F} f(x) \tag{9}$$

Using sum-product algorithm, the marginal is defined as $P(x) = \sum_{x \in X} P(X)$. Thus, the conditional probability of an event given the observed variables is calculated as the following.

$$P(y|x_1, x_2, x_3) = \sum_{x} f(x) \tag{10}$$

A.2 Extended Kalman Filter Implementation

Extended Kalman Filter (EKF) is a version of Kalman Filter that is used to estimate the state dynamics of a non-linear system. EKF consists of two steps: estimation and correction. State estimation is done using a non-linear function of the system $f(\cdot)$. The following equation shows EKF’s state estimation, where $x_t$ is the state estimated in the previous time step, $u_t$ is the actuator command.

$$x_{t+1} = f(x_t, u_t) \tag{11}$$

Equation 12 shows how EKF state correction is done using the available sensor measurements. $y_t$ is the sensor measurement, $h(x_{t|t-1})$ are EKF computed states in the previous time step, and $K$ is the Kalman Gain. Note that the Kalman Gain $K$ is calculated using Jacobian of transition and observation matrices and the sensor noise covariance matrix.

$$\hat{x}_{t+1} = f(x_t, u_t) + K_t(y_t - h(x_{t|t-1})) \tag{12}$$

We use non-linear system dynamics equations of RVs as models in our EKF estimator and for each RV we learn the model parameters (unknown coefficients) using system identification [10]. We collect a dataset of control actions and sensor measurements for each of our subject RVs. For example, we run missions capturing sensor readings and control signals to the rotors in various modes of operation of a drone - takeoff, loiter, auto, circle, and land.

A quadcopter’s state vector represents its position $(x, y, z)$, velocity $(v_x, v_y, v_z)$, angular orientation $(\phi, \theta, \psi)$, and angular speed $(\omega_x, \omega_y, \omega_z)$.

The following equations show the quadcopter dynamics [38]. $U_t$ is the thrust force, $U_\phi, U_\theta, U_\psi$ are control commands for rotors (roll, pitch, yaw), $I_x, I_y, I_z$ are moments of inertia in x, y, z axis respectively, $g$ is gravity, and $m$ is the mass.

\[
\begin{align*}
\dot{x} &= v_x \\
\dot{y} &= v_y \\
\dot{z} &= v_z \\
\dot{\phi} &= \omega_x \\
\dot{\theta} &= \omega_y \\
\dot{\psi} &= \omega_z \\
\dot{\omega}_x &= \frac{U_\phi}{I_x} + \frac{\beta}{I_x} \left(\frac{I_y - I_z}{I_x} \right) \\
\dot{\omega}_y &= \frac{U_\theta}{I_y} + \frac{\beta}{I_y} \left(\frac{I_z - I_x}{I_y} \right) \\
\dot{\omega}_z &= \frac{U_\psi}{I_z} + \frac{\beta}{I_z} \left(\frac{I_x - I_y}{I_z} \right) \\
\end{align*}
\]

The following equations show ground rover dynamics [32], where $\beta$ is the slip angle - measures sideways movement of wheels, $\delta$ is the steering angle, $l_T$ is the distance between center of mass and front axle, $l_s$ is the distance between center of mass and rear axle.

\[
\begin{align*}
\beta &= \tan^{-1}\left(\frac{l_r}{l_T + l_s} \tan(\delta)\right) \\
\dot{x} &= v \cos(\psi + \beta) \\
\dot{y} &= v \sin(\psi + \beta) \\
\dot{\psi} &= \frac{v}{l_s} \sin(\beta) \\
\dot{\theta} &= a \\
\end{align*}
\]
A.3 Algorithm

Algorithm 1 shows DeLorean’s algorithm for attack diagnosis and recovery. In the absence of attacks (based on the attack detector’s response), DeLorean records the physical states associated with all the on-board sensors (Lines 7-15) in a sliding window. Once the attack detector raises an alert, DeLorean activates recovery mode (Line 17). First, it stops recording, and prepares the most recent safe HS for recovery. Then, it activates the diagnosis procedure (Line 21). The attack diagnosis module determines the sensors under attack, if any, using FGs (Line 28-34). Once the targeted sensors are determined, DeLorean uses the historic states for the compromised sensors and current state estimates from uncompromised sensors to reconstruct RV’s states (Line 23), effectively representing the historic physical states used for recovery (Line 24). The attack diagnosis module determines the sensors under attack, if any, using FGs (Line 28-34). Once the targeted sensors are determined, DeLorean uses the historic states for the compromised sensors and current state estimates from uncompromised sensors to reconstruct RV’s states (Line 23), effectively representing the historic physical states used for recovery.

Algorithm 1: Algorithm for Attack Diagnosis and Recovery

1: w ← historic state recording window
2: S ← historic physical states used for recovery
3: xr ← physical states corresponding to targeted sensors
4: xs ← physical states corresponding to uncompromised sensors
5: procedure AttackRecovery
6: alert ← AttackDetector()
7: while mission_end do
8: if alert then
9: recovery_mode ← false
10: if tw < tω₁ then
11: record ← true
12: w[\_] ← x(t)
13: else
14: delete w,₁
15: t_w = tω₁
16: end if
17: else
18: recovery_mode ← true
19: record ← false
20: HS = w,₁
21: targetedSensors ← Diagnosis()
22: x_r ← estimateStates(HS[\_])
23: x_s[\_] ← stateReconstruction(x_r, x_s)  // state reconstruction
24: end if
25: end while
26: end procedure
27: procedure Diagnosis
28: while !mission_end ∨ sensors do
29: e = ||(e_t) - (e,_,₁)||
30: s_t ← argmaxP(s_t | e)  // most probable outcome
31: if s_t = 1 then
32: return malicious  // targeted sensors
33: end if
34: end while
35: end procedure

A.4 Mission Types

We run a diverse set of missions to derive the experimental parameters and for our experiments. Table 8 shows the details.

A.5 Metrics

Normalized RMSSD As minor attitude deviations are expected in RV missions even in the same trajectory due to variations in control signals and computations in RV’s feedback control loop.

Table 8: Mission paths used in evaluating DeLorean. S: Straight line, MW: Multiple waypoints, C: Circular, and Three different polygonal paths P1, P2, and P3

| Mission Paths | S | MW | C | P1 | P2 | P3 | Total |
|---------------|---|----|---|----|----|----|-------|
| Number of missions | 70 | 70 | 50 | 50 | 50 | 340 |

To ensure that these minor deviations are accounted for in our recovery stability metric, we introduce a normalization technique for the RMSSD values. This normalization approach transforms the RMSSD values (calculation shown in Equation 5) into a standardized range between 0 and 1 as follows:

\[
\text{Normalized RMSSD} = \frac{\text{RMSD} - \text{min RMSSD}}{\text{max RMSSD} - \text{min RMSSD}} \tag{13}
\]

Where min RMSSD and max RMSSD are minimum and maximum RMSSD values among recovery activated missions.

Normalized Mission Delay As minor variations in mission completion times can occur even on the same trajectory, we normalize the mission delay using a baseline mission completion time \(T_{baseline}\) as the reference point for mission completion time for an RV on a given trajectory. \(T_{baseline}\) is derived using min-max approach as shown below. Where \(T_{baseline}\) is the average of the minimum and maximum mission completion times in attack-free missions.

\[
T_{baseline} = \frac{T_{\min} + T_{\max}}{2} \tag{14}
\]