# Stealthy Attacks Against Robotic Vehicles Protected by Control-based Intrusion Detection Techniques

PRITAM DASH, University of British Columbia
MEHDI KARIMIBIUKI, University of British Columbia
KARTHIK PATTABIRAMAN, University of British Columbia

Robotic vehicles (RV) are increasing in adoption in many industrial sectors. RVs use auto-pilot software for perception and navigation and rely on sensors and actuators for operating autonomously in the physical world. Control algorithms have been used in RVs to minimize the effects of noisy sensors, prevent faulty actuator output, and recently, to detect attacks against RVs. In this paper, we demonstrate the vulnerabilities in control-based intrusion detection techniques, and propose three kinds of stealthy attacks that evade detection and disrupt RV missions. We also propose automated algorithms for performing the attacks without requiring the attacker to expend significant effort, or to know specific details of the RV, thus making the attacks applicable to a wide range of RVs. We demonstrate the attacks on eight RV systems including three real vehicles in the presence of an Intrusion Detection System (IDS) using control-based techniques to monitor RV's runtime behavior and detect attacks. We find that the control-based techniques are incapable of detecting our stealthy attacks, and that the attacks can have significant adverse impact on the RV's mission (e.g., deviate it significantly from its target, or cause it to crash).

CCS Concepts: • **Security and Privacy → Intrusion detection systems**; • **Computer systems organization → Sensors and actuators**.

Additional Key Words and Phrases: Cyber Physical Systems (CPS), Invariant Analysis, Robotic Vehicle Security

**ACM Reference Format:**
Pritam Dash, Mehdi Karimibiuki, and Karthik Pattabiraman. 2020. Stealthy Attacks Against Robotic Vehicles Protected by Control-based Intrusion Detection Techniques. 1, 1 (September 2020), 25 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

Robotic Vehicles (RVs) are cyber-physical systems (CPS) that operate autonomously leveraging closed-loop feedback control mechanisms (*e.g.*, PID controller [23]). Two prominent examples of such systems are Unmanned Aerial Vehicles (UAVs), also known as drones) and Unmanned Ground Vehicles (UGVs), also known as rovers. Such vehicles are utilized in a variety of industrial sectors (*e.g.*, agriculture, surveillance, package delivery [6, 8, 58], warehouse management [61]) and even critical missions such as space exploration [44]. Unfortunately, such vehicles are not well protected, and are vulnerable to both physical and cyber attacks. Examples of such attacks demonstrated in previous research are GPS spoofing [30, 64], gyroscope sensor tampering [60], attacks on vehicles' braking system [59]. These attacks can cause significant damage to the RV, and cause it to fail in its mission.

Authors' addresses: Pritam Dash, University of British Columbia, Vancouver, Canada, V6T 1Z4, pdash@ece.ubc.ca; Mehdi Karimibiuki, University of British Columbia, Vancouver, Canada, V6T 1Z4, mkarimib@ece.ubc.ca; Karthik Pattabiraman, University of British Columbia, Vancouver, Canada, V6T 1Z4, karthikp@ece.ubc.ca.


XXXX-XXXX/2020/9-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Because RVs inherently use control algorithms for minimizing sensor or actuator faults and for trajectory planning [54], control-based techniques have been proposed to detect attacks. Control Invariants (CI) [15] and Extended Kalman Filter (EKF) [9] are two such techniques that uses RV's mission profile data (*e.g.*, control inputs and outputs) to extract invariants of the system and create a model that correlates between sensor inputs and actuator outputs. Based on the current sensor inputs, CI and EKF models estimate both the next state and the control output signal of the RV. The estimated values are used to monitor the RV's runtime behaviour and flag anomalous behaviour, thus detecting attacks.

In this paper, we highlight the vulnerability of control-based intrusion detection. We propose automated techniques to launch attacks against RVs protected by CI and EKF techniques. Our main insight is that by design, CI and EKF techniques have to tolerate some degree of deviation from the planned trajectory due to environmental factors such as friction, wind or sensor noise, and hence have a certain threshold for flagging deviations as attack. Further, we found that the invariants extracted by CI and EKF fail to accurately model RV's runtime behavior. Therefore, CI and EKF techniques set a large threshold in order to avoid false alarms. We propose an automated process by which an attacker can learn the thresholds and the tolerances of each system for any arbitrary RV that uses Proportional Integral Derivative (PID) control, the most commonly used controller [36], and consequently perform targeted attacks against the RV. By controlling the deviation introduced and the timing of the attacks, we show that the attacker can remain stealthy and not be detected by techniques such as CI and EKF. Furthermore, though the deviations may be small, the consequences of the attacks are severe as they can be performed over a prolonged period of time, and at a time and place of the attacker's choosing. This makes them particularly insidious when RVs are used in safety-critical and time-critical scenarios.

We propose three types of attacks on RVs that are undetectable by current control-based techniques.

(1) *False data injection*: We devise an automated approach through which the attacker can derive the control state estimation model of RVs and reverse engineer it to obtain the detection threshold and monitoring window used in the IDS. Exploiting the aforementioned threshold related imperfections, the attacker can launch sensor and actuator tampering attacks such that the deviations in the control output are always maintained under the detection threshold, i.e., a false data injection attack [37]. By performing such a controlled false data injection over a period of time, the attacker will be able to significantly deviate the RV from its original mission path.
(2) *Artificial delay*: We launch artificial delays into the system's control execution process, which will affect the timing behaviour of crucial system functions. We show that the attacker can inject intermittent delays in the reception of the RVs gyroscopic sensor measurements, which will, in turn influence the estimation of RV's angular orientation while eluding detection. By launching stealthy, intermittent delays, the attacker can adversely influence the RV's performance and efficiency.
(3) *Switch-mode attack*: Finally, we identified that the invariants derived by CI and EKF fail to accurately model the RV's runtime behaviour when the RV switches modes (*e.g.*, when a drone switches from steady flight to landing), hence do not provide tight bounds. We exploit this weakness to launch another form of false data injection attack on sensor and actuator signals, which is triggered upon the RV switching modes.

Prior work has focused on exploiting the vulnerabilities in communication channels, and attacks on the RV's sensors through noise injection [15, 59, 60] in the absence of any protection. In contrast, we consider a scenario where the RV is protected by both CI and EKF technique, which makes the attacker's job much more difficult. Further, unlike prior work, we make minimal assumptions on the RV itself, and instead completely automate the attack generation and learning process, without requiring any apriori knowledge of the system on the part of the attacker (other than that the RV is using a PID control system). This makes our technique applicable to a wide range of RVs. *To the best of our knowledge, we are the first technique to automatically find attacks against the control state estimation model of RVs without being detected by existing techniques, or targeting a specific type of RV.*

We make the following contributions in this paper:

(1) Demonstrate three types of stealthy attacks namely: false data injection, artificial delay, and switch mode attacks against RVs in the presence of both CI and EKF attack detection techniques. The attacks can significantly deviate the RVs from their missions trajectory, disrupt RV missions and even result in crashes without being detected.
(2) Propose automated algorithms for launching the above three attacks against any *arbitrary* RV without apriori knowledge of its internals. We derive the thresholds and states of the RVs, and the protection techniques by repeated observations, and learn the control models used for state estimation.
(3) Implement the attacks on eight RV systems based on Ardupilot, PX4 and Paparazzi auto-pilot platform among which three are real RV systems. We also use simulation platforms to demonstrate the attacks on a wider variety of missions and trajectories.
(4) We find that attackers can learn the thresholds, monitoring windows, and states of the RVs using a modest amount of effort (typically 5 to 7 missions). We further show that the stealthy attacks can have severe repercussions such as deviating a drone by more than 160 meters from its trajectory (for a mission distance of 5 Kilometers), and deteriorating the efficiency and performance of rovers and drones by increasing their mission duration by more than 65% and 30% respectively. If launched strategically at vulnerable states, the stealthy attacks can also cause a drone to crash while landing, or cause other undesirable effects (*e.g.*, ignoring user commands). Finally, we show that the attacks can be generalized across different RVs.

## 2 BACKGROUND

We first discuss the architecture and control processes of RVs, followed by a description of its modes of operation, and how attacks propagate in RV systems. Then, we present the control-based attack detection mechanisms namely Control Invariants [15] and Extended Kalman Filter [9] (EKF). Finally, we present the attack model.

### 2.1 Robotic Vehicle Control

At a high level, the RV system consists of three components: (i) Flight controller software *e.g.*, Ardupilot [7], PX4 [63] or Paparazzi [62] that provides high-level functions to enable complex flight modes as well as other control functionalities, (ii) Controller hardware such as Pixhawk, Bebop or Navio2 that serves a centralized interface to command and control low-level hardware, (iii) Low-level hardware consisting of sensors, motors, propellers, etc.

An RV system uses a number of sensors (*e.g.*, barometer, gyroscope, accelerometer, magnetometer, and GPS) for navigation and perception. The raw sensor data captures the physical state of the vehicle in the environment (*e.g.*, angular and linear position), and aids in calculating the actuator signals (*e.g.*, rotors speed, steering) for positioning the vehicle in the next state. RVs use Proportional-Integral-Differential (PID) control algorithm to determine the actuator signals based on error and a weighted sum of the propositional (P), integral (I), and derivative (D) terms. Typically, in the case of drones or rovers, a PID controller is used for position control (*e.g.*, estimating altitude, latitude, longitude), and attitude control (*e.g.*, estimating yaw, roll, pitch). Figure 1 (based on ArduPilot [7]) shows an example illustrating the PID controller used in path planning along each axis.

The position control is done using a $P$ controller to convert the target position error (difference between the target position and actual position) into target velocity, followed by a PID controller to derive the target angle (roll, pitch, yaw). Similarly, the target angles are given as input to the attitude controller, and using the PID control functions, a high-level motor command is calculated. A PID controller can be described by the following formula [15]:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{\mathrm{d_e(t)}}{\mathrm{d}t}$$

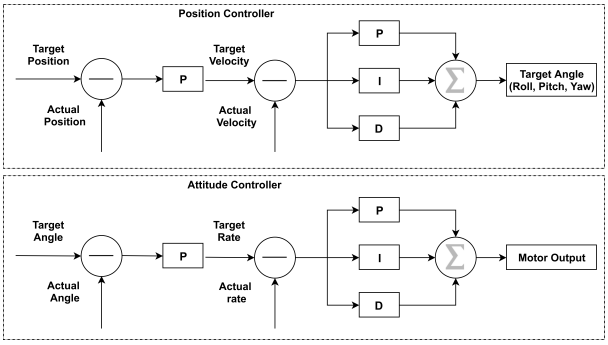



Fig. 1. PID Control operations in RVs: Position Control and Attitude Control.

$P$ is the proportional term, which aims to adjust the control signal (e.g., the rotor currents) proportional to the error; $I$ is the integral term, which is for tracing the history of the error. It compensates for $P$'s inability to reduce the error in the previous iterations. $D$ is the derivative term to avoid stark change in the error.

## 2.2 Modes of Operation in RV Mission

For a given flight path, an RV transitions through a series of high level states typically referred as modes of operation. In the case of a drone for instance, when a mission starts, the drone is armed at its home location. When the *Takeoff* mode is triggered, the drone takes off vertically to attain a certain height. Subsequently, a series of modes can be performed such as *Loiter* mode, *Waypoint* mode, which will prompt the drone to fly autonomously to a pre-defined location, and Return to launch (*RTL*) mode, which will prompt the drone to return to home. The *Land* mode enables the drone to drop elevation when it arrives at the destination. Figure 2 (based on ArduPilot SITL [7]) shows a state diagram of the various mode of operation commonly deployed in a drone. The change in mode of operation causes a change in the angular orientation, control input and actuator signals. The PID control algorithm plays a crucial role in balancing the RVs, and ensuring smooth flight when a dynamic mode change is triggered during an RV mission.

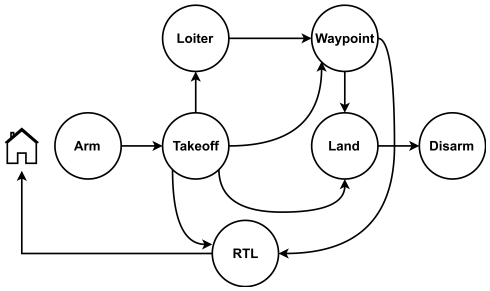



Fig. 2. Modes of operation in RVs.

## 2.3 Attacks Against RVs

As RVs inherently rely on sensor measurements for actuation, one of the most successful ways of triggering attacks against RVs is through sensor tampering or spoofing [30, 60]. Attackers often launch sensor tampering

attacks by injecting false data (an arbitrary bias value) to raw sensor measurements through acoustic noise injection [37, 60]. False data injection attack (FDI) can be launched to manipulate both sensor and actuator signals. When an FDI attack is launched, the sensor signal $u$ is replaced with an manipulated signal $u^a = u + bias$, where an arbitrary bias value is selected so that $u^a$ causes significant fluctuations in RV's control operations. A similar FDI attack can be launched to manipulate actuator signals $y$.

Another way of launching attacks against RVs is by compromising the inertial measurement unit (IMU) to trigger artificial delays in the control processes or a denial of service [67]. When an artificial delay attack is launched at time $t_i$, the receptors will not receive the recent sensor signals $u_i.....u_n$. Such an obstruction will prevent the RV system from performing critical control operations. These sensor tampering and artificial delay attacks cannot be prevented through traditional software security measures such as encrypted communication and memory isolation [15]. Real-time invariant analysis has been proven effective in detecting such attacks [1, 3, 5, 11]. As RV systems use control algorithms for position and attitude control, control-based invariant analysis techniques have been proposed for securing RVs [15, 38].

In this paper, we build on the ideas of FDI and artificial delay injection attacks to design novel stealthy attacks against RVs, which are undetected by invariant analysis techniques using control properties.

## 2.4 Control Invariants

The control invariant (CI) approach [15] models the physical dynamics of an RV and leverages its control properties to derive invariants (*e.g.*, control outputs). The control invariants are determined by two aspects namely, vehicle dynamics, and the underlying control algorithm. For a given RV, the CI model captures the system's sensor inputs, based on its current state to estimate the systems' control outputs. The approach then derives invariants using the following state estimation equations.

$$x'(t) = Ax + Bu \tag{1}$$

$$y(t) = Cx + Du \tag{2}$$

Where $x(t)$ is the current state, and $u(t)$ is the control input. $A, B, C, D$ are state space matrices determined through system identification [40]. The above equations determine the next state $x'(t)$ and output $y(t)$ of the system based on the current state and control input signal. The CI model uses a stateful error analysis, where it accumulates the error (deviation) between the estimated output and the actual output in a pre-defined monitoring window. When the accumulated error exceeds a pre-defined threshold, the CI technique raises an alert *e.g.*, if the error for roll angle ($error = |y(t)_{est} - y(t)_{act}|$) is larger than 91 degrees (threshold) for a window of 2.6 seconds.

## 2.5 Extended Kalman Filter

Extended Kalman Filter [9] is commonly used in RVs to fuse multiple sensor measurements together to provide an optimal estimate of the position and/or orientation. For example, EKF fuses accelerometer, gyroscope, GPS and magnetometer measurements with a velocity estimate to estimate the UAV's yaw, pitch and roll. The estimate of the system's state is given by the following equation:

$$x'(t) = Ax + Bu + K(y(t) - C(Ax + Bu)) \tag{3}$$

Where $K$ is the steady-state Kalman gain, and $A, B, C$ are the state space matrices. An IDS based on EKF uses the residual analysis technique to detect sensors and actuator attacks. The difference between the real-time sensor measurement and the estimate of the sensor measurement is the residual vector, which is defined as:

$$r(t) = y(t) - C(Ax + Bu) \tag{4}$$

Where $r(t)$ is residual at each time-instant $t$. An IDS based on EKF compares if the residual $r(t)$ is larger than a pre-defined threshold for a certain monitoring window, and raises an alarm when such anomalous behaviour is observed [39].

## 3 ATTACK MODEL

The goal of the attacker is to perform stealthy attacks and prompt deviations in the RV's mission by manipulating sensor and actuator signals in the presence of an IDS using the CI and EKF techniques, and modify the timing behaviour of the system events or control events of the RV and adversely influence its performance and efficiency. Stealthy means the attack does not cause any immediate unexpected behaviour. For instance, a stealthy attack manipulates the sensor measurements and actuator signals in a controlled manner such that the deviations are within the expected thresholds during an RV mission. When performed over a prolonged period, the attack deviates the RV from its defined mission path and/or adversely affects its performance and efficiency.

Our Systems Under Test (SUT) are quadcopters and ground rovers. There are two attack vectors through which the attacker can perform the stealthy attacks, namely (i) malicious code injection, and (ii) acoustic noise injection or sensor spoofing. *The former has to be done via the GCS as RVs today only accept commands from it. The latter can be done directly on the RV provided it is in physical proximity.* With the increase in adoption of RVs in industrial use cases, it is expected that future RVs (*e.g.*, delivery drones) will operate in a distributed manner and communicate with each other to complete tasks efficiently [13, 61]. In this case, it is possible for attackers to use a compromised RV to send malicious packets to other RVs.

We assume that the attacker has the following capabilities:

- Manipulate the sensor measurements (*e.g.*, GPS, gyroscope, accelerometer) through acoustic noise injection.
- Snoop on the control inputs and outputs and derive the RV's state estimation model (*i.e.*, the state-space matrices).
- Access the application binary that runs on board the RV systems.
- Replace the dynamically linked system libraries in the RV's software stack through code injection [4, 27].
- Perform a coordinated attack by tampering multiple sensor measurements at once.

However, we assume that the attacker cannot tamper with the firmware, does not have root access to the Operating System (OS), and cannot delete system logs. Furthermore, the attacker does not know the physical properties of the RV, such as the detailed specifications of its shape. In addition, the low-level control parameters (*e.g.*, how the vehicle reacts to control signals) and the commands from the auto-navigation system (*e.g.*, mission semantics of the vehicle) are not known to the attacker. However, the attacker does need to know that the IDS uses CI and/or EKF models to derive the invariants - this is so that he/she can modulate the attack accordingly (if not, the attacker can simply assume both techniques are deployed together).

## 4 LIMITATIONS IN EXISTING METHODS AND STEALTHY ATTACKS

This section describes the limitations in CI and EKF techniques, and how we exploit those limitations to design stealthy attacks. Then, we discuss a few attack scenarios to analyze the repercussions of such attacks when targeted at RVs deployed in industrial use-cases. Finally, we describe the main challenge we address.

### 4.1 Stealthy Attacks

As mentioned, the CI and EKF techniques derive invariants leveraging the control properties, and estimate the vehicles' position and angular orientation. An IDS based on CI and EKF models will analyze the error (*i.e.*, deviation) between the real-time values and the estimated values. If the error is substantial for a pre-defined monitoring window ($t_w$), it is treated as an anomaly and an alarm is raised. However, RVs may incur natural errors caused due to environmental factors. Therefore, to avoid false positives due to natural errors, and to

accommodate overshooting of the RV, the IDS accumulates errors in a monitoring window, and compares the aggregated error with a pre-defined threshold. Therefore, instead of performing direct comparison between the real-time control outputs and the predicted control outputs, the detection techniques perform a threshold-based ($\tau$) comparison as shown below.

$$IDS(t_w) = \begin{cases} 1, & if \sum_{t_i}^{t_j} |V_{predicted} - V_{realtime}|_n > \tau \\ 0, & otherwise \end{cases} \tag{5}$$

Attackers can exploit the aforementioned attack detection principle and successfully perform stealthy FDI attacks on sensor and actuator signals in three ways as follows.

First, the error values under the threshold limit for a certain monitoring window are acceptable, and will not be reported as anomalies. Assuming the attacker figures out the threshold, he/she can trigger stealthy attacks by injecting false data $f_i$ to the sensor signal $u$ in a controlled manner to replace it by $u^a : u^a = u + f_i$. Such manipulations in sensor signals will result in fluctuations in actuator signal $y$ (shown in Equation 6) causing the RV to gradually deviate from its mission trajectory.

$$y_i^a = y_i + (\tau - |V_{predicted} - V_{realtime}|_i) \tag{6}$$

The false data $f_i$ is calculated such that $u^a$ does not cause major deviations in actuator signal $y$. By performing such an attack for a long period of time, the attacker will be able to cause a substantial deviation. Because the deviation $d = y_i^a - y_i$ is within the accepted threshold $\tau$ ($d < \tau$), the control-based techniques (CI and EKF) will not be able to detect it.

Second, because the detection techniques employ a fixed monitoring window for threshold comparison, an attacker can inject artificial delays between time $t_i$ and $t_j$, which will obstruct the system from receiving the current set of sensor measurements $u_i....u_j$. Such delays can stop the system for a few seconds, and prevent the system from performing critical operations such as mode changes. The attacker can inject the delay attacks intermittently to avoid accumulating large errors, which might trigger the IDS.

Finally, we found that the invariants derived using CI and EKF are insufficient in providing a close estimate of target angles when the RV switches modes (*e.g.*, when the drone is commanded to land after flying at a fixed height). In other words, the difference between the runtime values and the estimated values becomes larger when the RV switches to *Land* mode from *Waypoint* mode. Therefore, the detection techniques will have to employ a larger threshold to avoid false alarms. This enables the attacker to inject large false data $f_{sm}$ into $u$ or $y$ signals such that the deviation $d = d + f_{sm} : d < \tau$, without triggering alarms, and abruptly destabilizing the RV.

## 4.2 Attack Scenarios

Each of the stealthy attacks presented in this paper exploits a weakness in the CI/EKF techniques identified in the previous section. In this section, we discuss the impact of the attacks when performed against RVs in industrial scenarios. Table 1 shows the attackers' goal, the type of attack to achieve the goals, and how the attack would affect the RVs operations in an industrial use-case.

**False Data Injection (FDI)** This attack enables the attacker to mutate the sensor measurements to a value desirable for them. For instance, an attacker may inject false readings to the gyroscopic sensor measurements, which would make the drone unstable. Prior work [15, 60] simulated similar attacks using acoustic noise signals to tamper with the sensors of an RV, causing a major deviation in the intended path of the RV. However, we are interested in performing more subtle mutations to sensor readings. The goal is to simulate subtle and minor deviations in a controlled manner for an extended time period, and to maintain the deviation just under the threshold pre-defined by an IDS using CI and EKF. Instead of causing a large deviation (*e.g.*, 60 degrees) at once (which might trigger the IDS and result in the attack being detected), the attacker can intermittently inject small false data values to the sensor readings. This will influence the control operations causing a difference in the

Table 1. Attacker's goal, types of attack and its consequences.

| Attack Goal | Scenarios | Attack Type | Consequences |
|---|---|---|---|
| Deviate the RV to a desired location | Deviating a delivery drone | False data injection | Drone may deliver a package at wrong location |
| Influence RVs performance | Disrupting productivity of warehouse rovers | Artificial delay | Rovers may not follow the right organization pattern and products will be stored randomly. |
| Damage, crash or cause major disruptions | Crash a drone while landing | Switch mode | Payload items could be damaged |

drone's position and angular orientation. By performing such minor deviations for a prolonged period of time, the attacker will be able to divert the drone to his/her desired location.

**Artificial Delay** With this attack, the attacker influences the timing behaviour of the system events or the controller events by injecting artificial delays. Such artificial delays can allow attackers to change the timing of important system actions (*e.g.*, change in mode of operation), delay essential API calls, or cause other controller functionality to be suppressed. For instance, autonomous rovers are increasingly deployed in warehouses to facilitate inventory management and packaging. These rovers receive real-time commands to pick up or drop a package at a given location in the warehouse area. With artificial delay attacks, the attacker can cause an RV to receive a particular command at a delayed time. However, if the RV receives the sensor measurements of a previous state in the mission, the difference between the estimated behavior and observed behaviour for a pre-defined motioning window will increase. This may potentially trigger an alert by the IDS. Therefore, to maintain stealthiness, the attacker will need to inject such delays intermittently and not perpetually.

**Switch Mode (SM)** The SM attack is a form of FDI launched at highly vulnerable states in the RV's mission. Knowing the current mode of operation the attacker can inject malicious code, which is triggered when the RV switches its mode of operation. For instance, when a drone switches to *Land* mode, a malicious code snippet will overwrite the actuator signals. This will prompt the drone to gain elevation instead of landing, or increase the rotor speed causing the drone to land harder than is safe, potentially resulting in a crash. When such an attack is launched against delivery drones, it may damage the packages, or hurt the recipients of the package. Because the attack will not cause the monitoring parameters to exceed the pre-defined threshold, the IDS will not be able to detect it.

### 4.3 Challenges

The main challenge for the attacker is to launch attacks against RVs while remaining undetected by the CI and EKF techniques (if the techniques are deployed). Therefore, to remain undetected, the attacker needs to (1) learn the system parameters such as current state, control input, control output, etc., (2) derive the detection parameters such as monitoring windows and thresholds set by CI and EKF techniques, and (3) derive techniques to manipulate the sensor readings or inject delays by just the right amounts, and at the right times to remain stealthy, even while achieving his/her aims. This is the challenge we address in this paper.

## 5 APPROACH

In this section, we describe the approach for performing each of our stealthy attacks. First, we describe the steps necessary for preparing and performing the stealthy attacks. Then, we present the algorithms for executing the stealthy attacks against RVs.

### 5.1 Attack Preparation

Figure 3 presents an overview of the common steps required for carrying out each attack. This section describes the steps in detail.

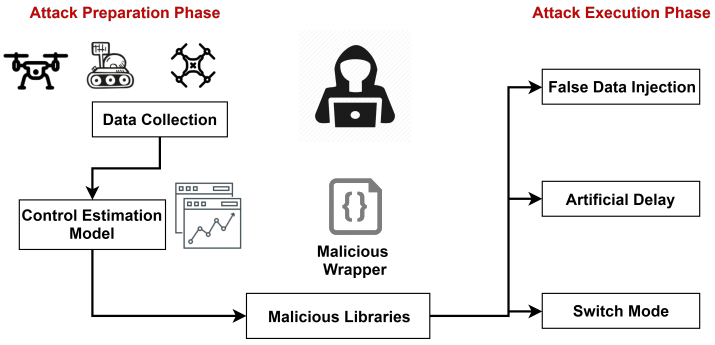



Fig. 3. Attack Overview.

**Data Collection**: The first step in attack preparation is to collect mission profile data of the RV. The attacker can either collect mission profile data from a real RV, or he/she can simulate the missions for the RV to achieve a realistic mission profile. The time series data of the target state $x'(t)$, current state $x(t)$, control input $u(t)$, control output $y(t)$ parameters will be used to derive the state estimation model (*i.e.*, the state space matrices). Ideally, the attacker will collect traces from two control operations namely, position control and attitude control (Figure 4). The *Position Controller* takes the target position as input, and applies the PID control algorithm to calculate the target angles along $X$, $Y$, and $Z$ axis). The actual position is looped back as feedback to the controller. The *Attitude Controller* takes the target angle as input, and calculates the motor outputs (rotation speed). The actual angles are looped back to the controller. The attacker will record the parameters pertaining to the above mentioned control operations (*e.g.*, target velocity and actual velocity along $x, y, z$ axis, target acceleration, target and actual angles, angular velocity and angular rate). Ideally, the attacker will collect mission profile data from different mission trajectories, covering multiple modes of operation to generate an accurate state estimation model. However, the data does not have to be comprehensive to derive the state estimation model for RVs [15].

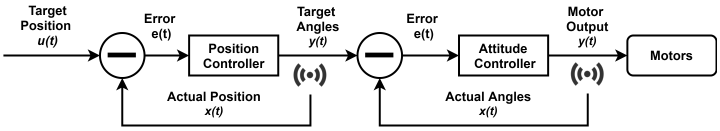



Fig. 4. Position and Attitude Controllers in RV.

**Control State Estimation Model**: Both CI and EKF derive invariants based on the vehicle dynamics and the underlying control algorithm (typically PID control in the case of RVs). The invariant generation process heavily relies on the state estimation model as shown in Equations 1,2, 3 and 4. The attacker's goal is to derive the unknown coefficients for solving the aforementioned equations. The mission profile data collected in the above steps can be used to derive the state estimation model (*i.e.*, state space matrices). To derive the $A, B, C, D$ state space matrices, the attacker can use system identification (SI) [40], which is a process to develop a dynamic

model of the system using the system's control input and control output data. From the state space matrices, the attacker can derive the Kalman gain $K$. The procedure is explained in Appendix A.

**Malicious Libraries** Typically, the RV's software uses two broad set of libraries for i) control operations such as PID control, attitude estimation (AHRS), and motor mixing etc. ii) sensor operations such as performing inertial measurements, GPS interface, optical interface etc. The APIs are specific to each class of RVs, but do not vary within a class (typically distributed as shared libraries). For example, the Autopilot software stack, which is deployed on many RVs, has a common set of shared libraries. One of the ways the attacker can perform the stealthy attacks is by replacing the original shared libraries with malicious ones. The malicious libraries will contain the attack code snippets. Once the unknown coefficients ($A, B, C, D, and\ K$) for solving the control equations are derived, the attacker will package them with the malicious library to perform threshold comparisons in runtime.

**Malicious Wrapper**: The attacker will design a malicious wrapper which will overwrite the original control and sensor libraries with malicious libraries by exploiting the dynamic linking feature [4]. When the RV software makes an API call to the control or sensor libraries, the malicious libraries will be called.

The attacker can also inject acoustic noise at the resonant frequency [60] to achieve the same results. However, because of the difficulties associated with such noise injection (*e.g.*, the noise source has to be in close proximity of the RV, and the impact of the attack is unknown) it will be harder to perform the attacks in realistic settings. Our approach is similar to that of Choi *et al.*[15], who also simulated noise injection through a piece of attack code.

## 5.2 Attack Execution

**False Data Injection (FDI)** To perform an FDI attack, the attacker will need to derive the threshold and the monitoring window for the CI and EKF models as follows. i) CI model - To derive the monitoring window, the attacker can use the time series data collected in the steps above to figure out the maximum temporal deviation between the observed control output sequences and the corresponding estimated control output sequences via a sequence alignment algorithm (*e.g.*, [57]). Once the window is obtained, the attacker can calculate the accumulated error in this window and select the accumulated value as the threshold. This is similar to the dynamic time wrapping technique used in Choi *et al.*[15]. ii) Leveraging EKF's state correction - EKF accumulates the error between the predicted angular orientation and the measurements of accelerometer and gyroscope in a large matrix called *State Covariance Matrix*. When the error is larger than the threshold, it applies a filter, which is referred to as *State Correction*, and the state covariance matrix is updated. The attacker can perform experiments on a simulator by injecting noisy sensor measurements to observe the time interval at which the state covariance matrix is updated. This time interval is the most viable monitoring window that the state estimation model based IDS can employ, and the accumulated error in this monitoring window will be the threshold. To remain stealthy, the attacker will need to manipulate the control input parameters such that the deviations in the control output signal are within the detection threshold of both CI and EKF.

Algorithm 1 shows the algorithm to launch FDI attack on the RV's position controller by manipulating the angular orientation measurements. The function `falseDataInjection` will get triggered when the RV's software components make an API call to the malicious libraries. The pre-computed state space matrices and the threshold values will be packaged with the malicious library (Lines 2 to 5). Based on the error threshold, the attacker will derive a value $f$ for a target sensor. The duration of false data injection $t_{attack}$ is based on the monitoring window. Lines 10 to 13 manipulate the control input $u(t)$ by injecting false data $f$ in the sensor measurements. Lines 18 to 24 manipulate the value of $f$ when the deviation approaches the detection threshold in order to remain stealthy. Since the detection procedure resets the accumulated error (Line 25) for each monitoring window, the attack will not be detected by the CI and EKF techniques.

**Algorithm 1:** FDI in sensor readings

```
1  Function FalseDataInjection():
2      A, B, C, D: pre-calculated state-space matrices.
3      K: pre-calculated Kalman gain.
4      T_CI : pre-defined threshold for CI.
5      T_EKF : pre-defined threshold for EKF.
6      t_attack: duration of attack.
7      f: false data
8      while (t_attack) do
9          T_Angle ← Target angle;
10         A_Angle ← Actual angle; (read data from sensor)
11         A_Angle ← A_Angle + f;
12         attitude target_X = A_angle − T_angle;
13         u = attitude target_X;
14         X_n = A ∗ x + B ∗ u;
15         Y_Roll = C ∗ x + D ∗ u;
16         R = Y_Angle − C ∗ X_n;
17         d = |Y_Angle − T_Angle|;
18         error_CI = error_CI + d;
19         error_EKF = error_EKF + R;
20         d_sum = d_sum + d
21         if d_sum > T_CI and d_sum > T_EKF then
22             f = 0;
23         end
24     end
25     error_CI, error_EKF, d_sum = 0;
26 return T_Angle;
```

**Artificial Delay** The attacker can trigger the artificial delay (AD) attack by including a code snippet in the malicious library called `ArtificialDelay`, which when triggered will perform certain resource intensive operations. Such delays will obstruct other system calls and control operation, thereby disrupting the timing behaviour of the systems. However, if the delay is triggered for a long time period, the error accumulation in the invariant analysis will increase and the IDS might raise an alarm. To remain stealthy under such an IDS, the attacker can use the monitoring window found in the above steps as a threshold ($t_{AD}$) and not allow delays longer than this threshold. By triggering the snippet `ArtificialDelay` intermittently and under the threshold $T_{AD}$, the attacker will be able to bypass the detection mechanism. The function $ArtificialDelay$ in Algorithm 2 shows an example of executing resource intensive operation (*e.g.*, infinite recursion, computationally intensive calculations etc.) to cause delays. The duration of delays to be injected $t_{AD}$ is derived based on the monitoring window ($t_w$) used in the invariant analysis model (CI, EKF) as shown in Line 4.

**Switch Mode** The switch mode (SM) attack is a form of FDI attack launched at a few, highly vulnerable states of an RV mission. To execute this attack, in addition to the detection threshold and the monitoring window as per the CI and EKF techniques, the attacker will have to monitor the mode of operations of the RV. Algorithm 3 shows an example of switch mode (SM) attack launched when the RV changes its operations to *LAND* mode (Line 11). The attacker can also launch such attacks at other mode transitions (*e.g.*, from *Takeoff* to *Waypoint*). Similar to the FDI attack, here the attacker will derive a value $f_{sm}$, which when injected to the motor thrust value will disrupt the RV's behaviour (Line 23). Further, to remain stealthy this attack will be carried out for a specific attack duration $t_{attack}$, which is derived based on the monitoring window. In this case, as the threshold is found

**Algorithm 2:** Artificial Delay Attack

```
Function ArtificialDelay():
    t_Now: current system time.
    t_w: monitoring window.
    t_AD = t_Now + t_w;
    while true do
        if t_AD < t_Now then
            ArtificialDelay();
        else
            break;
        end
    end
    t_AD = Null;
return
```

**Algorithm 3:** Switch mode attack - influencing actuator signals

```
Function SwitchModeAttack():
    A, B, C, D: pre-calculated state-space matrices.
    K: pre-calculated Kalman gain.
    T_CI : pre-defined threshold for CI.
    T_EKF : pre-defined threshold for EKF.
    t_attack: duration of attack;
    f_sm: false data;
    while i<num-motors do
        T_motor = getPWMOutput(i);
        Mode = getCurrentMode();
        if Mode = LAND then
            while (t_attack) do
                X_n = A * x + B * u;
                Y_motor = C * x + D * u;
                R = Y_motor − C * X_n;
                d = |Y_motor − T_motor|;
                error_CI = error_CI + d;
                error_EKF = error_EKF + R;
                d_sum = d_sum + d
                if d_sum > T_CI or d_sum > T_EKF then
                    f_sm = 0;
                end
                motor[i] = thrustToPWM() + f_sm;
            end
        else
            motor[i] = thrustToPWM();
        end
    end
end
```

to be larger than normal, the attacker can inject larger false values which may result in severe consequences in a short time duration, e.g., causing a drone to crash.

## 6 EXPERIMENTS AND EVALUATION

In this section, we discuss the experimental setup, followed by the research questions (RQs) we ask. Then, we present the results of the experiments to answer the RQs.

(a) Pixhawk Drone (b) Aion R1 Rover (c) Sky-viper Drone

Fig. 5. Real RV Systems used for Experiments.

### 6.1 Experimental Setup

To demonstrate the stealthy attacks, we use eight different RV systems among which three are real RVs shown in Figure 5. The other five systems are on simulation platforms. For real RVs, we use (1) Pixhawk based DIY drone [43] (henceforth called *Pixhawk drone*), (2) an Aion R1 ground rover [55] (henceforth called *R1 rover*), and (3) Sky Viper Journey drone [56] (henceforth called *Sky-viper drone*) for real RVs. For the simulations, we use (4) Ardupilot's quadcopter (henceforth called *ArduCopter*), (5) Ardupilot's ground rover (henceforth called *ArduRover*), (6) PX4 Solo software in the loop (SITL) (henceforth called *PX4 Solo*) [63], (7) PX4 Rover SITL and (8) Paparazzi UAV [62]. We run the simulators on an Ubuntu 16.0 64-bit machine with Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz processor and 8 GB RAM.

**Software** We use three different auto-pilot software stacks namely: ArduPilot [7], PX4 [63], and Paparazzi [62]. All the three auto-pilot software stacks use PID controller for position and attitude control. However, they vary in their internal architecture for handling sensor measurements and control functions. For vehicle simulation, we use APM SITL [7], JSMSim [31], and Gazebo [52] platforms.

**Hardware** Both the Pixhawk drone and R1 rover (Figure 5) used in our experiments are based on the Pixhawk platform [43]. Pixhawk is an ARM Cortex based all-in-one hardware platform, which combines flight management unit (FMU) controller, I/O, sensors and memory in a single board. It runs NuttX, which is a Unix-based real-time operating system, on a 32-bit Cortex processor and 256 KB RAM [68]. The Sky-viper drone is based on STM32 processor and uses an IMU including 3-axis accelerometer, gyro and barometer. Note that our attacks are not tied to a specific hardware or software platform.

**Attack Setup** We performed 20 missions on both the simulations and the real vehicles, and collected the time series data of control input $u(t)$, system state $x(t)$, and the control output $y(t)$. The time series data was collected from both the position control and attitude control operations of the RV (Figure 4) because the sensor manipulations are targeted at both the control operations. These data sets were used to derive the state estimation model[1].

To perform the attacks, we designed a set of malicious libraries for the following control libraries of the ArduPilot software suite: `AHRS`, `AttitideControl`, and `PositionControl`. We overwrote the environment variable (*LD_LIBRARY_PATH*) in the *.bashrc* file to point to the malicious libraries instead of the originals - this technique has been used in prior work as well [4][2]. This will load the malicious libraries instead of the original ones.

[1] All the code and data-sets used in this paper can be found at *https://github.com/DependableSystemsLab/stealthy-attacks*
[2] A similar effect can be achieved by executing a Trojan program or shell code, for example.

Henceforth, when the RV software components call functions defined in the above libraries, the corresponding function in the malicious library will be called (as the malicious libraries have a function with the same name as defined in the original library). The malicious libraries can stay dormant until the RV is deployed on a critical mission, at which point, they can get triggered.

### 6.2 Research Questions

RQ1. How much effort does the attacker need to expend to derive the state estimation model?
RQ2. What are the impacts of the stealthy attacks on the subject RVs?
RQ3. How effective are the attacks in achieving the attacker's objectives?

### 6.3 Results

In this section, we present the results of the stealthy attacks experiments performed on the subject RVs to address the above RQs.

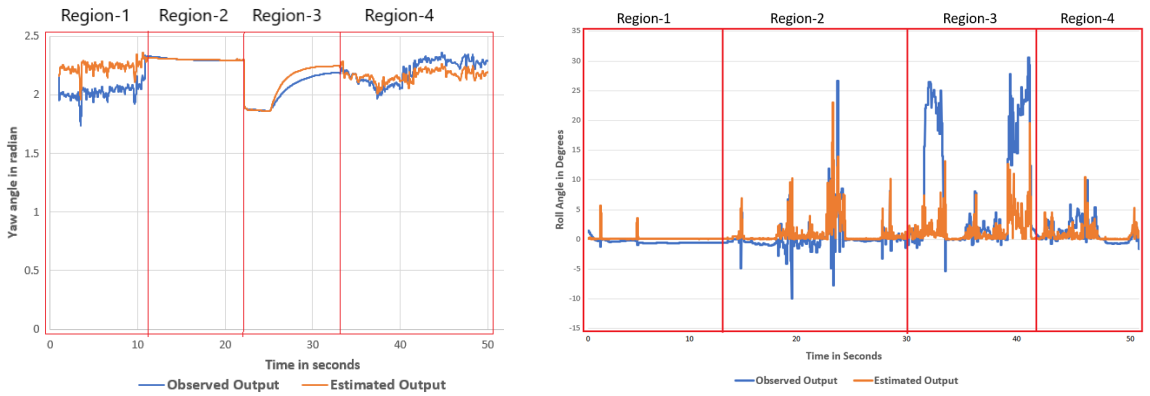
**RQ1: Attacker's effort** The first set of experiments aim to quantify the effort required on the attacker's part in deriving an accurate state estimation model for a subject RV. We divided the mission data into two sets: i) Model extraction set - used to derive the state estimation model (15 missions for each subject RV, both simulation and real vehicles), and ii) Model testing set - used to test the accuracy of the obtained state estimation model (5 missions for each subject RV, both simulation and real vehicles).

We followed an iterative approach in deriving the state estimation model and evaluating its accuracy. In the first iteration, we randomly picked 5 mission profiles from the model extraction set and using system identification [40, 41], we derived the $A, B, C, D$ matrices and the Kalman gain $K$. Following Equations 1, 2, 3, and 4, we estimated the system output (*e.g.*, roll, pitch, yaw) for missions in the model testing set. Then we analyzed the accuracy of the state estimation model by comparing the estimated and the realtime system outputs.

For each subsequent iteration, we added 1 more mission profile data from the model extraction set, derived an updated model, and performed the above analysis to identify if the accuracy of the state estimation model has converged. From this experiment, we found that all model estimated outputs converged to the real-time outputs by the third iteration. For some subject RVs, the convergence occurred after the first iteration. Overall, across all the RVs, the model converged with just 5 to 7 mission data, and hence the attacker can derive an accurate state estimation model with modest effort.

Even in cases where the model converged, for some states of the RV's mission path, the state estimation model failed to provide precise estimates. Figure 6 shows an example of the Pixhawk drone, where the model did not converge. We have divided the graph into 4 regions: 1, 2, 3, and 4, based on the different modes of the RV's mission. As can be seen in Figure 6a, the CI model estimated output converges to the realtime outputs in Regions 2 and 3, but not in Regions 1 and 4. For a different mission, Figure 6b shows that EKF model estimated output are very close to the real outputs in Regions 1 and 4, but not in Regions 2 and 3. The RVs realtime control outputs relies on the realtime sensor measurements (control input $u(t)$) and the $P, I, D$ gains (Section 2). Based on the RV's trajectory and its current mode of operation, the sensor measurements may incur additional noise. The PID control functions may consequently adjust the gain param to mitigate the effects of the noise, which will influence the realtime control outputs. However, the model estimated control outputs are not updated as per the runtime PID parameter adjustments. Therefore, it is difficult to achieve high convergence between the model estimated and the real-time values with system identification based techniques such as CI and EKF. Therefore, both CI and EKF techniques are forced to employ a high detection threshold in order to avoid false alarms.

**RQ2: Impact of the stealthy attacks** In the second set of experiments, we performed the stealthy attacks in the presence of an IDS using the CI and EKF models respectively. Before performing the attacks, the attacker will have to derive the detection threshold and the monitoring window. For the CI model, we followed the dynamic




(a) CI model

(b) EKF model

Fig. 6. Control-based Models - Observed vs Estimated Outputs.

time wrapping method (explained in Section 5), to derive the monitoring window. To identify the monitoring window of the EKF model, we performed experiments in each simulation platforms by injecting small amounts of noise into the sensor measurements, and observing the intervals at which the state covariance matrix is updated (explained in Section 5). Then, we calculated the most viable thresholds CI-based or EKF-based IDS can employ based on the error accumulated in the monitoring windows. The detection thresholds and monitoring windows derived for all the subject RVs are presented in Table 3.

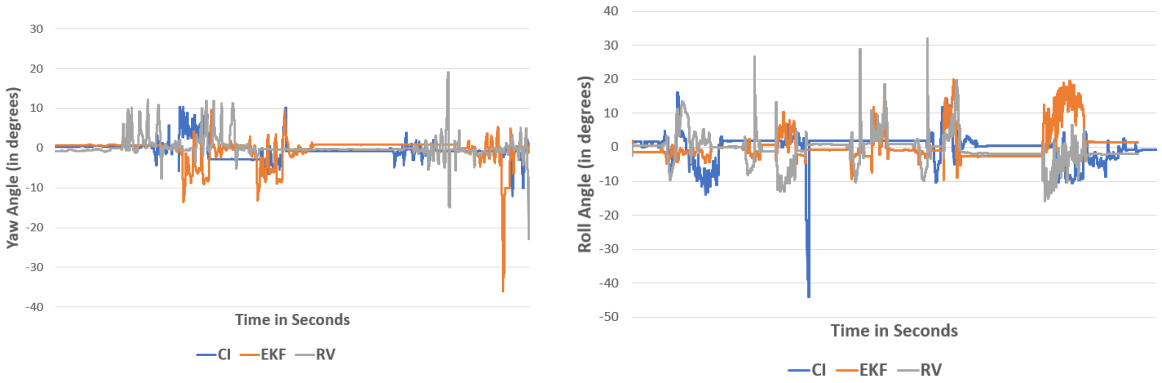
Table 3 also shows the impact of the attacks on the subject RVs. The results shown in the table are based on data from 5 missions, and consist of the average values of the attack's outcomes (deviations, delays) in the presence of both CI-based IDS and EKF-based IDS. Our results show that the thresholds set by CI and EKF model allow a considerable margin for stealthy attacks to be launched. The attacks cause substantial deviation in the RVs trajectory (deviation of 8 to 15 m for a mission distance of 35-50 m), adversely influence their efficiency by increasing the mission duration by 30% to 68%, and even result in crashes when timed during landing. We discuss a few examples of the attacks.

*False Data Injection (FDI)*: For all the subject vehicles, we injected false data as per Algorithm 1 to influence the position and attitude controller of the RVs, which in turn manipulates the actuator outputs, thereby deviating the RV from its trajectory. We inject false data on GPS and gyroscope measurements, which influence the position control outputs (yaw, roll, pitch angles) and attitude control outputs (thrust to PWM) respectively.

Figure 7 shows how the FDI attack manipulates the RV's Euler angles. The injected false data $f$ (discussed in Section 4) modifies the Euler angles in the range $0 - 45$ degrees intermittently. The intermittent and controlled FDI prevents accumulation of large error within the monitoring windows, thereby bypassing the CI and EKF techniques, as it is within the thresholds used by them.

Table 3 shows the deviations caused by FDI attack for all the subject RVs. As can be seen in the table, for the Pixhawk drone running ArduCopter auto-pilot, the average deviation caused by FDI attack is 11 m for a mission distance of 50 m (video can be found at [49]). When running PX4 auto-pilot, the deviations increase to 12.5 m. For the same mission, the average deviation for the Sky-viper drone is 15 m. On the other hand, the FDI attack deviated the R1 rover by 11.2 m from its defined destination (for a mission distance of 35 m). Further, on many occasions, the FDI attack prompted the R1 rover to follow arbitrary paths (i.e., non-deterministic paths) such as turning backwards, or causing the RV to deviate significantly from the defined straight line mission path.

As it is logistically difficult to perform experiments on real vehicles with large mission duration and distance, we instead perform detailed analysis of this attack on the simulator by increasing the mission distance (from 100m to 5000m). We found that the deviation increases almost linearly with mission distance. For example, we found that for a mission distance of 5000 m, the FDI attack can deviate the drone by as much as 160 meters.




(a) Aion R1 Rover

(b) Pixhawk Drone

Fig. 7. FDI attacks on subject RVs.

*Artificial Delay*: The artificial delay attacks were also launched intermittently, and the duration of the delay was lower than the monitoring window duration found above. For a given monitoring window, we injected delays in the vehicles' position and attitude control operations. Such an attack will prevent the actuator from receiving the correct outputs and commands based on the recent sensor measurements. We found that these attacks were more disruptive for the rovers than the drones, both in the real world and in simulations (video can be found at [48]). As shown in Table 3, the attack increases the mission time of Pixhawk drone, Sky-viper drone and R1 rover by 30%, 35% and 65% respectively.

*Switch Mode (SM)*: We only applied the SM attack on drones (both real and simulated), as the rovers used in our experiments only had a few operational modes, and hence did not experience many mode transitions. As we discussed above (Figure 6), the model estimated values and the real-time values do not converge for all of the modes, and hence we posit that the detection threshold should accommodate large offsets to prevent false alarms. We found that for the Pixhawk drone and the Sky-viper drone, the offset was as large as 14 degrees for both CI and EKF models. This enabled us to inject large false values into the sensor measurements during mode switching which resulted in major disruptions.

Figure 8 shows the roll angle predictions of CI and EKF models and how the large faults manipulates the roll angles when an SM attack is launched against the Sky-viper drone. Though the manipulations caused by SM attack are larger compared to the FDI attack, the large thresholds (shown in Table 3) set by CI and EKF provides enough room for manipulation without triggering alarms.

We also observed many instances of the drones crashing during the SM attacks across a wide range of trajectories, mission distances and mission types (shown in Table 2). Further, when the SM attack was launched during landing, it resulted in crashes more often in the case of the Sky-viper drone than the Pixhawk drone. We believe this is because Sky-viper is a very light weight drone with six axis rotation capabilities (its weight is only 150 g while the Pixhawk drone weighs nearly 2000 g). Hence, the SM attack which triggers large manipulations in sensor measurements could drastically destabilize the Sky-viper drone, but not the Pixhawk drone.

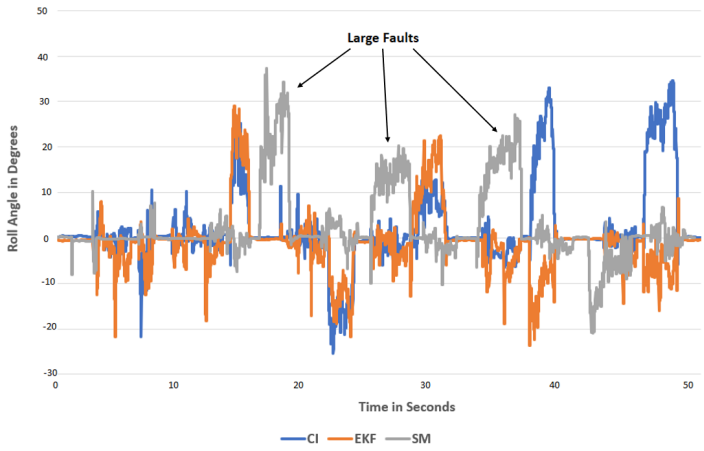



Fig. 8. Switch Mode Attack against Sky-viper Drone.

Table 2. Results of Switch Mode Attack against RVs

| RV systems | No. of missions | Mission Distance (meters) | SM Attack Target | No. of crashes |
|---|---|---|---|---|
| Pixhawk/ArduCopter | 5 | 50 | Land | 2 |
| Pixhawk/PX4 | 5 | 50 | Land | 2 |
| Sky-viper | 5 | 50 | Land | 4 |
| ArduCopter | 10 | 50 to 5000 | Takeoff and Land | 7 |
| PX4 Solo | 10 | 50 to 5000 | Takeoff and Land | 9 |
| Paparazzi UAV | 10 | 50 to 5000 | Takeoff and Land | 8 |

**RQ3: Effectiveness of the attacks** From the above experiments, we found that the FDI attack can cause a deviation of 8 to 15 m (for a mission distance of 50 m, and mission time about 40 seconds) in an RV's mission trajectory. For long distance RV mission (5 km or more), the deviations caused by the FDI attack is more than 100 m. When the FDI attack is launched simultaneously on both the position and attitude controllers, the deviation increases to 160 m for the same mission. Typically, drones deployed in industrial use-cases such as package delivery, surveillance, etc., will operate autonomously for a mission duration of more than 30 minutes [8, 10], and cover a distance up to 20 km [29]. In such missions, the impact of the FDI attacks can be much more significant.

The SM attacks can also cause major repercussions, even for short missions. From our experiments (not presented in the table) we found that for a mission distance of 50 m, the SM attack prevented the drone from flying to the destination. Instead the drone loitered at a certain height. In another instance, the attack caused the drone to ignore the "land" command, and the drone kept gaining elevation (video can be found at [50]). Further, we were able to crash the drone by strategically launching the SM attack when the drone switched to the *Land* mode. When such attacks are launched against drones in industrial use-cases such as package delivery, they can cause damage to the drone and other nearby objects including the packages.

The artificial delay attack increased the mission duration by more than 65% for the R1 rover and by more than 30% for the drones. Although this attack does not directly deviate or damage the RV, it can have major performance and efficiency related consequences when launched against RVs in industrial use-cases. For example, drones are used for delivery of time critical items such as blood samples and drugs [8, 10], and rovers are used to increase the productivity in warehouses [61], where timeliness is important.

Table 3. Results of the attacks on different types of RVs and the impacts of the attacks. Note that SM attacks are only applicable to drones and not to rovers in our experiments.
MD: Mission Distance, FT: Flight Time, TH: Threshold, MW: Monitoring Window
FDI: False Data Injection, SM: Switch Mode, AD: Artificial Delay

| RV System | Attacks | MD (m) | FT (s) | Control Invariants (CI) | | | Extended Kalman Filter (EKF) | | | Attack Impacts |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | TH (degree) | MW (S) | Deviation (m) | TH (degree) | MW (s) | Deviation (m) | |
| ArduCopter | FDI | 50 | 45 | 60 (yaw angle | 2.0 | 11 | 52 (yaw angle) | 2.3 | 10 | RV landing at wrong location |
| | SM | 50 | 49 | | | 7 | | | 7 | Crash landing |
| | AD | 50 | 71 | | | - | | | - | 54% increase in mission time. |
| ArduRover | FDI | 50 | 42 | 72 (roll angle) | 2.6 | 14 | 60 (roll angle) | 2.3 | 11 | RV deviated from mission path |
| | AD | 50 | 72 | | | - | | | - | 56% increase in mission time. |
| PX4 Solo | FDI | 50 | 40 | 9 (roll rate) | 3.5 | 12.4 | 6.1 (roll rate) | 3.5 | 11 | RV landing at wrong location |
| | SM | 50 | 46 | | | 8 | | | 7 | Crash landing |
| | AD | 50 | 61 | | | - | | | - | 51% increase in mission time |
| PX4 Rover | FDI | 50 | 45 | 8.2 (roll rate) | 2.5 | 15 | 7 (roll rate) | 3.0 | 11.5 | RV deviated from mission path |
| | AD | 50 | 79 | | | - | | | - | 68% increase in mission time |
| Paparazzi UAV | FDI | 50 | 51 | 6.6 (roll rate) | 2.0 | 8.5 | 5 (roll rate) | 2.4 | 6 | RV deviated from mission path |
| | SM | 50 | 57 | | | 6 | | | 6 | RV landed at wrong location |
| | AD | 50 | 83 | | | - | | | - | 65% increase in mission time |
| Pixhawk/ArduCopter | FDI | 50 | 32 | 60 (yaw angle) | 2.0 | 11 | 45 (yaw angle) | 2.3 | 8 | RV deviated from the trajectory |
| | SM | 50 | 34 | | | 6 | | | 6 | Unstable landing at wrong location |
| | AD | 50 | 41 | | | - | | | - | 30% increase in mission time |
| Pixhawk/PX4 | FDI | 50 | 33 | 8.2 (roll rate) | 3.5 | 12.5 | 6.1 (roll rate) | 3.5 | 10 | RV deviated from the trajectory |
| | SM | 50 | 36 | | | 9 | | | 8 | Unstable landing at wrong location |
| | AD | 50 | 44 | | | - | | | - | 33% increase in mission time |
| Sky-viper Drone | FDI | 50 | 45 | 81 (roll angle) | 2.6 | 15 | 67 (roll angle) | 3.5 | 13 | RV deviated from the trajectory |
| | SM | 50 | 51 | | | 7 | | | 7 | Crash landing |
| | AD | 50 | 60 | | | - | | | - | 33% increase in mission time |
| Aion R1 Rover | FDI | 36 | 35 | 82 (roll angle) | 2.6 | 11.2 | 60 (roll angle) | 2.5 | 9 | RV followed arbitrary path |
| | AD | 36 | 59 | | | - | | | - | 65% increase in mission time. |

## 7 DISCUSSION

The main limitation of our attack approach is that the state estimation model, as well as the threshold and monitoring window values, vary for RVs using different hardware platforms (*e.g.*, the model derived from Pixhawk drone does not not apply to Sky-viper drone). Therefore, the attacker will have to expend the effort of repeating the steps in the *Attack Preparation Phase* for each class of RVs. In this section, we present the design of a self-learning malware program that will attack an RV adaptively without any human effort. Then, we discuss the other limitations of our attacks, followed by a discussion on how IDSes can be better designed for dynamic CPS such as RVs. Finally, we discuss some of the threats to the validity of our results.

### 7.1 Self-Learning Malware

We propose a technique though which an attacker with the same capabilities as in our attack model (Section 3) can automate the attack preparation phase through a self-learning malware. The attacker can design a program called `modelExtractor` to collect the sensor measurement and the mission profile data from position controller and attitude controller. The `modelExtractor` will work in tandem with the malicious library on the RV. For each mission, the `modelExtractor` will collect the data, and create an archive of mission profile data for various mission trajectories and mode of operations of the RV. As we said earlier (Section 5.1), the mission profile data from 5-7 missions is sufficient to derive an accurate state estimation model. After the RV has completed $n$ missions, the `modelExtractor` will trigger a system identification library [21, 47] to derive the state space matrices $A, B, C, D$ and the Kalman gain $K$.

From our experiments, we have found that the values of the monitoring window and the detection threshold based on EKF are typically smaller than those of CI. Therefore, the attacker can focus on deriving the EKF's threshold and monitoring window, by recording the time intervals at which EKF's state covariance matrix is updated. This way, the monitoring window can be extracted, and the error accumulated in this monitoring window will be the threshold. The `modelExtractor` program will pass these values to the attack algorithms (Algorithm 1, Algorithm 3), which will trigger the attacks based on the RVs mode of operation and mission state. While we have not implemented such a malware program and hence cannot measure its overhead, our preliminary experiments on performing these measurements and calculating the thresholds, indicate that the overhead of the malware program will likely be small.

## 7.2 Limitations

In FDI attacks, the value of the false data to be injected is calculated dynamically based on the threshold and the current state of the system in order to remain stealthy. However, in some situations the threshold values based on the CI and EKF models will not allow much room for performing FDI. For example, for an Erle-Rover, the CI model employs a threshold of 2.5 degrees (steering rate estimation) for a monitoring window of 4.2s [15]. We did not have access to a Erle-Rover to perform the attacks ourselves. Therefore, we performed experiments in the ArduRover simulator using the thresholds and monitoring window for Erle-Rover [15]. We found that for a mission distance of 50 meters, the deviation caused by the FDI attack was only 4 meters, which was considerably smaller than what we observed for the Aion R1 rover.

Further, in some of our experiments, we found that if the drone overshoots its trajectory because of the injected false values (for example, in a switch mode attack, where we injected large false values) the drone system activates the (hardware) fail-safe mode and forces the drone to land and abort the mission. This can be considered a limitation of our attack because the attack failed in achieving the desired outcome (i.e., deviating or delaying the RV). However, an attacker can take advantage of such a fail-safe landing mechanism, and force the drone to land at a location conducive to the attacker (different from the destination defined in the the drone mission).

## 7.3 Countermeasures

One way to mitigate the attacks in this paper is to design estimation models that demonstrate a high degree of convergence with the observed control outputs. This will enable an IDS to employ a smaller threshold value thereby limiting the room for sensor manipulation [25]. An improved version of CI and EKF techniques with adaptive thresholds and variable monitoring windows can be effective in limiting the stealthy attacks. The conventional methods for invariant extraction (both CI and EKF) use pre-defined fixed thresholds and monitoring windows for a subject RV. We exploit this notion of fixed bounds invariant analysis to trigger our attacks. If the IDS uses an adaptive threshold (*e.g.*, different threshold for steady state flight and $Land$ / $Takeoff$ modes), the leeway for injecting false values into sensor and actuator parameters will decrease, which in turn will reduce the impact of the FDI and switch mode attacks. Similarly, if the IDS employs variable-sized monitoring windows, the impacts of artificial delay attack will decrease. If the attacker injects a fixed size delay (as we do in our artificial delay attack), an IDS using a variable sized monitoring window may detect the attack. We defer detailed design of mitigation techniques for these attacks to future work.

## 7.4 Threats to Validity

We consider three threats to validity - i) Internal, ii) External and iii) Construct. An internal threat to our work is that we have considered only control-based attack detection techniques. Although we do not evaluate other attack detection techniques against our stealthy attacks, we posit that methods that follow a threshold based detection such as CI and EKF are vulnerable to our stealthy attacks. Another internal threat is that our experiments with

real vehicle were not very extensive. This is largely because of the logistical restrictions around using unmanned RVs over long distances, as well as time limitations. However, we mitigate this threat by performing extensive experiments on a simulator.

An external threat to work is that we have considered only two types of hardware platforms i.e., Pixhawk and Sky-viper. However, the RV's hardware platforms use flight controllers that include processors and sensors, and run a compatible auto-pilot software for control and navigation. The auto-pilot platforms typically use pre-designed libraries for control and sensor operations. Therefore, our stealthy attacks can be extended to other RV hardware platforms such as Navio [20] and Bebop [51] that deploy similar libraries.

Finally, a construct threat to our work is that if the detection threshold and monitoring windows are small, the effects of the stealthy attacks will not be as critical. However, as the detection threshold in CI and EKF methods is calculated using training traces, it is difficult to come up with small and precise threshold boundaries, for reasons such as sensor noise and environmental factors. Moreover, an aggressively calculated small detection threshold would result in high false positives, which is undesirable. Secondly, even with small detection windows, the artificial delay attack will still be able to cause undesirable consequences in the RV mission.

## 8 RELATED WORK

**Sensor spoofing attacks**. Previous work has demonstrated sensor spoofing attacks such as GPS spoofing [30][64] to misguide a drone's trajectory, optical spoofing [19] to gain an implicit control channel etc. It has also been shown that inaudible sound noise when injected at resonance frequency can adversely affect the MEMS gyroscopic sensor, which can cause the drone to crash [60]. Likewise, attackers can compromise the accelerometer of drones by injecting acoustic noise in a controlled manner [65]. However, these attacks are not necessarily stealthy, as they can be detected by the IDS depending on the degree of deviation they cause. In contrast, our attacks are designed to be stealthy.

Stealthy attacks such as false data injection have been demonstrated on industrial control systems to mislead state estimators and controllers [18, 37, 42]. Stealthy sensor spoofing attacks can induce the supervisor control layer into allowing the system to reach an unsafe state, thereby causing physical damage to the system [26]. Our attacks cause perturbations in the sensor measurements thereby inhibiting the RV from performing its task, and not necessarily causing physical damage (which is easier to detect).

**Malware attacks** Multiple instances of malware attacks on industrial control systems have been reported. A few prominent examples are the Stuxnet attack [32], and the attack on the power grid in Ukraine [35]. Malware with learning capabilities can derive an optimal attack vector and launch man-in-the-middle attacks [24]. Alamzadeh *et al.* [4] present a malware attack targeting a tele-operated surgical robot, where the malware identities an optimal attack time and injects faults. Similar attacks have been demonstrated on pacemakers [28]. Chung *et al.* [16] demonstrated feasibility of attacking water treatment systems using a self-learning malware. Subsequently, they extended their work to launch MITM attacks on surgical robots by exploiting the vulnerabilities in the underlying runtime environment (ROS) [17]. However, none of these attacks have been demonstrated on RVs protected with control-based IDS as in our work (to the best of our knowledge).

**Intrusion Detection Systems (IDS)** IDS have been proposed that uses physical invariants for each sensor and actuator to tackle attacks against different cyber-physical systems, including UAVs [45]. BRUIDS [46] is a specification based IDS that is adaptive based on the attacker type and environment changes. CORGIDS [2] derives correlations between physical properties using hidden Markov models, and uses these correlations as invariants. Adepu *et al.*[1] design an IDS for a water treatment plant by manually describing the invariants for a particular sensor in terms of the water level changes between two consecutive readings. ARTINALI [5] dynamically mines data, time and event invariants from an execution trace of the program and use data-time-event invariants to detect anomalies. Chen et al. [12] present an approach for automatically constructing invariants of CPS, by using

supervised ML on traces of data obtained from systematically mutated PLC programs. Ahmed *et al.*[3] propose a technique to fingerprint the sensor and process noise to detect sensor spoofing attacks. Kim *et al.*[34] proposed a technique for detecting control semantic bugs (input validation bugs) leveraging a control instability detector to detect RV malfunctions. Fei *et al.*[22] proposed retro-fitting UAVs controllers with reinforcement learning policies to recover from attacks. Kim *et al.*[33] presented a machine learning based method for detecting sensor spoofing attacks against RVs. Most of the above techniques [2, 22, 45, 46] use a threshold based technique to detect deviations from the invariants or models. Therefore, they are vulnerable to stealthy attacks like ours. That said, we did not consider these IDS in our evaluations as our attacks target control-based IDS techniques.

Quinonez *et al.* [53] present Savior, an EKF, and cumulative sum statistics (CUSUM) based technique for mitigating stealthy attacks against RVs. The difference between the EKF model considered in this paper and Savior is that Savior does not use a fixed monitoring window based error accumulation. Instead, Savior accumulates errors throughout the RV mission following the CUSUM algorithm [66], and raises an alert when the error is greater than a threshold. However, as Savior does not refresh the accumulated error based on a monitoring window, it can incur high false-positive rates in the presence of external noise [25]. We experimentally evaluated our FDI attack in the presence of Savior, and found that the attack can still cause significant deviations in RV missions without triggering any alarms (though it was only half as much as that with CI and EKF).

In recent work, Choi *et al.* present a technique to recover RVs from attacks by replacing the physical sensor measurements with those from software sensors once an attack is detected [14]. However, they use a linear state-space estimation model (similar to CI) to estimate the physical states of the RV, and determine whether the RV is under an attack based on a threshold analysis. Because the linear state-estimation technique fails to closely capture the RV's runtime behaviour and hence requires a high threshold, this recovery technique will only be effective against attacks that cause abrupt disruptions. Furthermore, the software sensors rely on parameters derived from physical sensor measurements (e.g., velocity), and thus a coordinated attack launched on multiple sensors may derail the recovery process.

## 9 CONCLUSION

In this paper, we highlight the vulnerabilities in control-theory based techniques namely CI (Control Invariants) and EKF (Extended Kalman Filter) used for attack detection in Robotic Vehicles(RVs). We find that these techniques use pre-defined detection threshold and monitoring window based invariant analysis techniques, and are hence susceptible to stealthy attacks. Moreover, both CI and EKF fail to achieve high accuracy in predicting RVs runtime control outputs which forces them to employ a large threshold in order to prevent false alarms.

To demonstrate how an attacker can exploit the vulnerabilities, we designed three stealthy attacks namely: false data injection, artificial delay attack and switch mode attack. We present algorithms that will automate the process of deriving the detection thresholds. Knowing the threshold, an attacker can perform stealthy sensor and actuator tampering attacks, thereby bypassing the detection mechanisms. We demonstrated the attacks in eight RV systems including three real systems, and on different auto-pilot software stacks. Though the attacks are stealthy in nature, and do not cause large-scale disruptions, we found that the consequences can still be quite severe such as: deviating a drone by more than 160 meters from its trajectory, increasing the mission duration of a rover and drone by more than 65% and 30% respectively, and causing a drone to crash while landing (and harming other objects). We also show that the attacks can be triggered against a diverse range of RV hardware platforms and auto-pilot software. Furthermore, we discuss the attacker's goals in the context of industrial use-cases, and discuss how the attacker can perform stealthy attacks to achieve his/her goals.

In our future work, we will explore the design of techniques that achieve high convergence in predicting RV's runtime behaviour across various modes of operations. High prediction convergence will allow employing a small detection threshold, which will limit the room for sensor manipulation, thereby mitigating stealthy attacks.

## ACKNOWLEDGEMENT

This research was supported by a research grant from the Natural Sciences and Engineering Research Council of Canada (NSERC), and a research gift from Intel. We thank Prof. Ryozo Nagamune, Department of Mechanical Engineering, University of British Columbia for his valuable feedback. We also thank the anonymous reviewers of ACSAC'19 for their comments which helped improve the paper.

## REFERENCES

[1] Sridhar Adepu and Aditya Mathur. 2016. Using process invariants to detect cyber attacks on a water treatment system. In *IFIP International Information Security and Privacy Conference*. 91–104.

[2] Ekta Aggarwal, Mehdi Karimibiuki, Karthik Pattabiraman, and André Ivanov. 2018. CORGIDS: A Correlation-based Generic Intrusion Detection System. In *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy (CPS-SPC '18)*. ACM, New York, NY, USA, 24–35. https://doi.org/10.1145/3264888.3264893

[3] Chuadhry Mujeeb Ahmed, Jianying Zhou, and Aditya P. Mathur. 2018. Noise Matters: Using Sensor and Process Noise Fingerprint to Detect Stealthy Cyber Attacks and Authenticate Sensors in CPS. In *Proceedings of the 34th Annual Computer Security Applications Conference (ACSAC '18)*. ACM, New York, NY, USA, 566–581. https://doi.org/10.1145/3274694.3274748

[4] H. Alemzadeh, D. Chen, X. Li, T. Kesavadas, Z. T. Kalbarczyk, and R. K. Iyer. 2016. Targeted Attacks on Teleoperated Surgical Robots: Dynamic Model-Based Detection and Mitigation. In *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. 395–406. https://doi.org/10.1109/DSN.2016.43

[5] Maryam Raiyat Aliabadi, Amita Ajith Kamath, Julien Gascon-Samson, and Karthik Pattabiraman. 2017. ARTINALI: Dynamic Invariant Detection for Cyber-physical System Security. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2017)*. ACM, New York, NY, USA, 349–361. https://doi.org/10.1145/3106237.3106282

[6] Amazon Prime [n. d.]. Amazon Prime Delivery. ([n. d.]). Retrieved January 24, 2019 from https://www.amazon.com/Amazon-Prime-Air/b?node=8037720011

[7] ArduPilot [n. d.]. Ardupilot - Software in the Loop. ([n. d.]). Retrieved May 24, 2018 from http://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html

[8] Aryn Baker. [n. d.]. Zipline Drone Delivery. ([n. d.]). Retrieved January 24, 2019 from http://www.flyzipline.com/

[9] P.-J. Bristeau, E. Dorveaux, D. VissiÃÍre, and N. Petit. 2010. Hardware and software architecture for state estimation on an experimental low-cost small-scaled helicopter. *Control Engineering Practice* 18, 7 (2010), 733 – 746. https://doi.org/10.1016/j.conengprac.2010.02.014 Special Issue on Aerial Robotics.

[10] Stephen Burns. [n. d.]. Drone meets delivery truck. ([n. d.]). Retrieved May 24, 2019 from https://www.ups.com/us/es/services/knowledge-center/article.page?name=drone-meets-delivery-truck&kid=cd18bdc2

[11] Alvaro Cardenas, Saurabh Amin, Bruno Sinopoli, Annarita Giani, Adrian Perrig, and Shankar Sastry. 2009. Challenges for Securing Cyber Physical Systems. In *Workshop on Future Directions in Cyber-physical Systems Security*. DHS.

[12] Y. Chen, C. M. Poskitt, and J. Sun. 2018. Learning from Mutants: Using Code Mutation to Learn and Monitor Invariants of a Cyber-Physical System. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Los Alamitos, CA, USA, 648–660. https://doi.org/10.1109/SP.2018.00016

[13] Grzegorz Chmaj and Henry Selvaraj. 2015. Distributed Processing Applications for UAV/drones: A Survey. In *Progress in Systems Engineering*, Henry Selvaraj, Dawid Zydek, and Grzegorz Chmaj (Eds.). Springer International Publishing, Cham, 449–454.

[14] Hongjun Choi, Sayali Kate, Yousra Aafer, Xiangyu Zhang, , and Dongyan Xu. 2020. Software-based Realtime Recovery from Sensor Attacks on Robotic Vehicles. In *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*. USENIX Association, San-Sebastian, Spain.

[15] Hongjun Choi, Wen-Chuan Lee, Yousra Aafer, Fan Fei, Zhan Tu, Xiangyu Zhang, Dongyan Xu, and Xinyan Deng. 2018. Detecting Attacks Against Robotic Vehicles: A Control Invariant Approach. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18)*. ACM, New York, NY, USA, 801–816. https://doi.org/10.1145/3243734.3243752

[16] Keywhan Chung, Zbigniew T. Kalbarczyk, and Ravishankar K. Iyer. 2019. Availability Attacks on Computing Systems Through Alteration of Environmental Control: Smart Malware Approach. In *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS '19)*. ACM, New York, NY, USA, 1–12. https://doi.org/10.1145/3302509.3311041

[17] Keywhan Chung, Xiao Li, Peicheng Tang, Zeran Zhu, Zbigniew T. Kalbarczyk, Ravishankar K. Iyer, and Thenkurussi Kesavadas. 2019. Smart Malware that Uses Leaked Control Data of Robotic Applications: The Case of Raven-II Surgical Robots. In *22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019)*. USENIX Association, Chaoyang District, Beijing, 337–351.

[18] G. Dan and H. Sandberg. 2010. Stealth Attacks and Protection Schemes for State Estimators in Power Systems. In *2010 First IEEE International Conference on Smart Grid Communications*. 214–219. https://doi.org/10.1109/SMARTGRID.2010.5622046

[19] Drew Davidson, Hao Wu, Rob Jellinek, Vikas Singh, and Thomas Ristenpart. 2016. Controlling UAVs with Sensor Input Spoofing Attacks. In *10th USENIX Workshop on Offensive Technologies (WOOT 16)*. USENIX Association, Austin, TX.

[20] Emlid. [n. d.]. Navio2. ([n. d.]). https://emlid.com/navio/

[21] ETH-Agile and Dexterous Robotics Lab. [n. d.]. Control Toolbox. ([n. d.]). https://ethz-adrl.github.io/ct/ct_doc/doc/html/index.html

[22] Fan Fei, Zhan Tu, Dongyan Xu, and Xinyan Deng. 2019. Learn-to-Recover: Retrofitting UAVs with Reinforcement Learning-Assisted Flight Control Under Cyber-Physical Attacks.

[23] Gene F. Franklin, J. David Powell, and Abbas Emami-Naeini. 2018. *Feedback Control of Dynamic Systems (8th Edition) (What's New in Engineering)*. Pearson. https://www.amazon.com/Feedback-Control-Dynamic-Systems-Engineering/dp/0134685717?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0134685717

[24] Luis Garcia, Ferdinand Brasser, Mehmet Hazar Cintuglu, Ahmad-Reza Sadeghi, Osama A. Mohammed, and Saman A. Zonouz. 2017. Hey, My Malware Knows Physics! Attacking PLCs with Physical Model Aware Rootkit. In *NDSS*.

[25] Ian Y. Garrett and Ryan M. Gerdes. 2020. On the Efficacy of Model-Based Attack Detectors for Unmanned Aerial Systems. In *Proceedings of the Second ACM Workshop on Automotive and Aerial Vehicle Security (AutoSec âĂŹ20)*. Association for Computing Machinery, New York, NY, USA, 11âĂŞ14.

[26] R. M. GÃşes, E. Kang, R. Kwong, and S. Lafortune. 2017. Stealthy deception attacks for cyber-physical systems. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. 4224–4230. https://doi.org/10.1109/CDC.2017.8264281

[27] J. Habibi, A. Gupta, S. Carlsony, A. Panicker, and E. Bertino. 2015. MAVR: Code Reuse Stealthy Attacks and Mitigation on Unmanned Aerial Vehicles. In *2015 IEEE 35th International Conference on Distributed Computing Systems*. 642–652.

[28] D. Halperin, T. S. Heydt-Benjamin, B. Ransford, S. S. Clark, B. Defend, W. Morgan, K. Fu, T. Kohno, and W. H. Maisel. 2008. Pacemakers and Implantable Cardiac Defibrillators: Software Radio Attacks and Zero-Power Defenses. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*.

[29] Andrew J. Hawkins. [n. d.]. UPS will use drones to deliver medical supplies in North Carolina. ([n. d.]). Retrieved May 24, 2019 from https://www.theverge.com/2019/3/26/18282291/ups-drone-delivery-hospital-nc-matternet

[30] Todd E. Humphreys. 2008. Assessing the Spoofing Threat: Development of a Portable GPS Civilian Spoofer. In *In Proceedings of the Institute of Navigation GNSS (ION GNSS*.

[31] JSMSim [n. d.]. JSBSim Open Source Flight Dynamics Model. ([n. d.]). Retrieved May 24, 2018 from "http://jsbsim.sourceforge.net/"

[32] S. Karnouskos. 2011. Stuxnet worm impact on industrial cyber-physical system security. In *IECON 2011 - 37th Annual Conference of the IEEE Industrial Electronics Society*. 4490–4494. https://doi.org/10.1109/IECON.2011.6120048

[33] Kyo Hyun Kim, Siddhartha Nalluri, Ashish Kashinath, Yu Wang, Sibin Mohan, Miroslav Pajic, and Bo Li. 2020. Security Analysis against Spoofing Attacks for Distributed UAVs. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS âĂŹ16)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2976749.2978388

[34] Taegyu Kim, Chung Hwan Kim, Junghwan Rhee, Fan Fei, Zhan Tu, Gregory Walkup, Xiangyu Zhang, Xinyan Deng, and Dongyan Xu. 2019. RVFuzzer: Finding Input Validation Bugs in Robotic Vehicles through Control-Guided Testing. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association, Santa Clara, CA, 425–442.

[35] Robert M. Lee, Michael J. Assante, and Tim Conway. 2016. Analysis of the Cyber Attack on the Ukrainian Power Grid. Technical Report. *Electricity Information Sharing and Analysis Center (E-ISAC)* (2016).

[36] J. Li and Y. Li. 2011. Dynamic analysis and PID control for a quadrotor. In *2011 IEEE International Conference on Mechatronics and Automation*. 573–578. https://doi.org/10.1109/ICMA.2011.5985724

[37] Yao Liu, Peng Ning, and Michael K. Reiter. 2009. False Data Injection Attacks Against State Estimation in Electric Power Grids. In *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS '09)*. ACM, New York, NY, USA, 21–32. https://doi.org/10.1145/1653662.1653666

[38] L. Ljung. 1979. Asymptotic behavior of the extended Kalman filter as a parameter estimator for linear systems. *IEEE Trans. Automat. Control* 24, 1 (February 1979), 36–50. https://doi.org/10.1109/TAC.1979.1101943

[39] K. Manandhar, X. Cao, F. Hu, and Y. Liu. 2014. Detection of Faults and Attacks Including False Data Injection Attack in Smart Grid Using Kalman Filter. *IEEE Transactions on Control of Network Systems* 1, 4 (Dec 2014), 370–379. https://doi.org/10.1109/TCNS.2014.2357531

[40] MATLAB. [n. d.]. System Identification Overview. ([n. d.]). https://www.mathworks.com/help/ident/gs/about-system-identification.html

[41] MATLAB. [n. d.]. System Identification Toolbox. ([n. d.]). https://www.mathworks.com/products/sysid.html

[42] S. McLaughlin and S. Zonouz. 2014. Controller-aware false data injection against programmable logic controllers. In *2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)*. 848–853. https://doi.org/10.1109/SmartGridComm.2014.7007754

[43] Lorenz Meier, Petri Tanskanen, Friedrich Fraundorfer, and Marc Pollefeys. 2011. Pixhawk: A system for autonomous flight using onboard computer vision. In *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2992–2997.

[44] MARS 2020 Mission. [n. d.]. MARS Exploration Rover. ([n. d.]). https://mars.nasa.gov/mer/mission/rover/

[45] Robert Mitchell and Ing-Ray Chen. 2012. Specification based intrusion detection for unmanned aircraft systems. *In Proceedings of the first ACM MobiHoc workshop on Airborne Networks and Communications* (2012), 31–36.

[46] Robert Mitchell and Ing-Ray Chen. 2014. Adaptive intrusion detection of malicious unmanned air vehicles using behavior rule specifications. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 44, 5 (2014), 2014.
[47] GNU Octave. [n. d.]. GNU Octave Scientific Programming Language. ([n. d.]). https://www.gnu.org/software/octave/
[48] Out of Control. [n. d.]. Artificial Delay Attack Demo Video. ([n. d.]). https://drive.google.com/open?id=1_CHITopKSraKZXnAIyeUoQZqki8fgUXe
[49] Out of Control. [n. d.]. False Data Injection Attack Demo Video. ([n. d.]). https://drive.google.com/open?id=1JgrCpwspsBiYdNvKUxeKnl-bZQ9WS_Cg
[50] Out of Control. [n. d.]. Switch Mode Attack Demo Video. ([n. d.]). https://drive.google.com/open?id=1yUSGa5GoBQYl0GTiTbcPFN5NnjBHXL-Y
[51] Parrot.com. [n. d.]. Bebop2. ([n. d.]). https://www.parrot.com/us/drones/parrot-bebop-2
[52] Gazebo Project. [n. d.]. Gazebo robot simulation. ([n. d.]). http://gazebosim.org/
[53] Raul Quinonez, Jairo Giraldo, Luis Salazar, Erick Bauman, Alvaro Cardenas, and Zhiqiang Lin. 2020. SAVIOR: Securing Autonomous Vehicles with Robust Physical Invariants. In *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Boston, MA.
[54] Hadi Ravanbakhsh, Sina Aghli, Christoffer Heckman, and Sriram Sankaranarayanan. 2018. Path-Following through Control Funnel Functions. *CoRR* abs/1804.05288 (2018). arXiv:1804.05288
[55] Aion Robotics. [n. d.]. R1 ArduPilot Edition. ([n. d.]). https://docs.aionrobotics.com/en/latest/r1-ugv.html
[56] Sky Rocket. [n. d.]. Sky Viper Journey. ([n. d.]). https://sky-viper.com/journey/
[57] H. Sakoe and S. Chiba. 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26, 1 (February 1978), 43–49.
[58] Nicolas Sheilds. [n. d.]. Walmart Drone Delivery. ([n. d.]). Retrieved December 09, 2018 from https://www.businessinsider.com/walmart-blockchain-drone-delivery-patent-2018-9
[59] Yasser Shoukry, Paul Martin, Paulo Tabuada, and Mani Srivastava. 2013. Non-invasive Spoofing Attacks for Anti-lock Braking Systems. In *Cryptographic Hardware and Embedded Systems - CHES 2013*, Guido Bertoni and Jean-Sebastien Coron (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 55–72.
[60] Yunmok Son, Hocheol Shin, Dongkwan Kim, Youngseok Park, Juhwan Noh, Kibum Choi, Jungwoo Choi, and Yongdae Kim. 2015. Rocking Drones with Intentional Sound Noise on Gyroscopic Sensors. In *24th USENIX Security Symposium (USENIX Security 15)*. USENIX Association, Washington, D.C., 881–896.
[61] Christopher Steiner. [n. d.]. Bot-In-Time Delivery. ([n. d.]). https://www.forbes.com/forbes/2009/0316/040_bot_time_saves_nine.html#68ee53d9b942
[62] Paparazzi Development Team. [n. d.]. Paparazzi - The Free Autopilot. ([n. d.]). https://wiki.paparazziuav.org/wiki/Main_Page
[63] Pixhawk Development Team. [n. d.]. Pixhawk AutoPilot. ([n. d.]). https://docs.px4.io/en/flight_controller/pixhawk_series.html
[64] Nils Ole Tippenhauer, Christina Pöpper, Kasper Bonne Rasmussen, and Srdjan Capkun. 2011. On the Requirements for Successful GPS Spoofing Attacks. In *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS '11)*. ACM, New York, NY, USA, 75–86.
[65] T. Trippel, O. Weisse, W. Xu, P. Honeyman, and K. Fu. 2017. WALNUT: Waging Doubt on the Integrity of MEMS Accelerometers with Acoustic Injection Attacks. In *2017 IEEE European Symposium on Security and Privacy (EuroS P)*. 3–18. https://doi.org/10.1109/EuroSP.2017.42
[66] David I. Urbina, Jairo A. Giraldo, Alvaro A. Cardenas, Nils Ole Tippenhauer, Junia Valente, Mustafa Faisal, Justin Ruths, Richard Candell, and Henrik Sandberg. 2016. Limiting the Impact of Stealthy Attacks on Industrial Control Systems. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16)*. ACM, New York, NY, USA, 1092–1105. https://doi.org/10.1145/2976749.2978388
[67] T. P. Vuong, G. Loukas, D. Gan, and A. Bezemskij. 2015. Decision tree-based detection of denial of service and command injection attacks on robotic vehicles. In *2015 IEEE International Workshop on Information Forensics and Security (WIFS)*. 1–6.
[68] Zhaolin Yang, Feng Lin, and B. M. Chen. 2016. Survey of autopilot for multi-rotor unmanned aerial vehicles. In *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*. 6122–6127. https://doi.org/10.1109/IECON.2016.7793820

# A RESEARCH METHODS

## A.1 State Estimation Model

The Algorithm 4 shows the process of deriving the state estimation model (*i.e.*, $A, B, C, D$ matrices) from the mission profile data. We collected time series data $M$ from $n$ missions, and we use Matlab's SI toolbox to generate the matrices [41]. For the first iteration, we randomly select $k$ mission profile data from the data set $M$ (Line 6). The time series data from $k$ missions is combined with `iddata` object, which consists of input and output value

matrices and a fixed sampling interval $t_s$ (Line 6 to 8). The control inputs $u$ and control outputs $y$ values collected in mission $i$ are represented as vectors. At line 20, the `tfest` function identifies the optimal coefficients for the model template from the RV's mission profile data. The transformation turns a time-domain function into the frequency domain and hence substantially reduces the complexity of fitting the profile data. At line 21, function `tfdata` accesses the resultant model: *num* and *den* that encode the model in the frequency domain. At line 22, `tf2ss` function converts a discrete-time transfer function into equivalent state-space representation. We test the accuracy of the state space model by comparing the model estimated values $(y(t), x'(t))$ with the recorded values. To improve the accuracy of the state estimation model, we perform system identification iteratively (Line 10 to 18) by adding 1 more mission profile data to the *iddata* object.

**Algorithm 4:** Generating State Space Model

```
1  M ← Mission profile data.
2  n: number of missions.
3  t_s: sampling interval.
4  N_p: poles.
5  N_z: zeroes.
6  while i < k do
7  |  data(i) = iddata (y(i), u(i), ts);
8  end
9  A, B, C, D ← systemIdentification(data, N_p, N_z);
10 while i < n − k do
11 |  if checkModelAccuracy(A, B, C, D) then
12 |  |  break;
13 |  else
14 |  |  data = iddata (y, u, ts);
15 |  |  A, B, C, D ← SystemIdentification(data, N_p, N_z);
16 |  |  checkModelAccuracy(A, B, C, D);
17 |  end
18 end
19 Function SystemIdentification(data, N_p, N_z):
20 |  tf = tfest(data, Np, Nz);
21 |  [num, den] = tfdata(tf);
22 |  [A, B, C, D] = tf2ss(num, den);
23 return A, B, C, D
```

## A.2 Kalman Gain

The state space matrices derived using system identification can be used to formulate a system model *sys*. The Matlab function `kalman` creates a state space model $K_{ss}$ of the Kalman estimator given the system model *sys*.

```
[Kss,K,P] = kalman(sys,Qn,Rn,Nn)
```

$K$ is the Kalman gain matrix, $P$ is the error covariance matrix, $Q_n, R_n, N_n$ are the noise covariance data.