Manipulating Drone Position Control

Wenxin Chen, Yingfei Dong
Department of Electrical Engineering
University of Hawaii
Honolulu, HI 96822

Zhenhai Duan
Department of Computer Science
Florida State University
Tallahassee, FL 32306

***Abstract*—Although consumer drones have been used in many attacks, besides specific methods such as jamming, very little research has been conducted on systematical methods to counter these drones. In this paper, we develop generic methods to compromise drone position control algorithms in order to make malicious drones deviate from their targets. Taking advantage of existing methods to remotely manipulate drone sensors through cyber or physical attacks (e.g., [1], [2]), we exploited the weaknesses of position estimation and autopilot controller algorithms on consumer drones in the proposed attacks. For compromising drone position control, we first designed two state estimation attacks: a *maximum False Data Injection (FDI) attack* and a *generic FDI attack* that compromised the Kalman-Filter-based position estimation (arguably the most popular method). Furthermore, based on the above attacks, we proposed two attacks on autopilot-based navigation, to compromise the actual position of a malicious drone. To the best of our knowledge, this is the first piece of work in this area. Our analysis and simulation results show that the proposed attacks can significantly affect the position estimation and the actual positions of drones. We also proposed potential countermeasures to address these attacks.**

***Index Terms*—position estimation, counter drone, drone navigation, Kalman filter**

I. Introduction

While consumer drones[1] have become extremely popular recently [3], they have been abused in many security incidents [4], [5]. In late Dec. 2018, unauthorized drones invaded the airspace of the second largest UK airport (Gatwick) for three days and disrupted about 1,000 flights with 140,000 passengers. Clearly, it is urgent to develop effective methods to disable malicious drones around important assets.

Existing Counter-Drone Methods. To protect a restricted airspace from drone invasions, e.g., at an airport or a critical infrastructure, as shown in Fig. 1, we need first discover the invasion of an unauthorized drone in the protected airspace and then apply countermeasures to disrupt, capture, or destroy it. For drone identification, existing methods usually utilize Radio Frequency (RF) communications, radars, acoustic monitoring, or image processing [6]–[8] to discover invading drones. Radar-based and RF-based detection methods are more reliable compared to other methods, e.g., in dealing with various light situations, noisy environments, or weather conditions. In this paper, we focus on the counter-drone step: to develop systematical countermeasures to compromise the navigation and control of invading drones, assuming that we are able to remotely manipulate the onboard sensors of the malicious drones.

[1]In the following, we use the term "drones" to represent consumer drones.

Although several counter-drone techniques have been developed (mostly by industry due to the acute demands), very little systematic investigation has been conducted on counter-drone technology. Existing techniques mostly apply *direct physical attacks*, such as capturing a drone with a net, or jamming its communication channels and GPS receivers to trigger its default fail-safe function after the loss of control channels or GPS signals. As a result, the limitations of these techniques are obvious: they are usually difficult to be automated for quick responses, and do not consider potential collateral damages, e.g., we do not want to shoot it down (or make it land) when a drone is carrying biological agents over a stadium. Therefore, more effective counter-drone techniques must be developed. In this paper, assume that cyber or physical attacks can help us remotely tamper onboard sensors (e.g., two existing attacks on gyroscopes [1] or accelerators [2]), we then develop generic counter-drone methods by compromising position estimation and autopilot-based navigation, e.g., deceiving a drone to a specific location.

By carefully examining existing literature, popular open-source flight control systems (*ArduPilot* and *Paparazzi*), and several proprietary firmware (3DR and Parrot), we have discovered multiple weaknesses in common control and navigation algorithms; e.g., we can easily learn their state

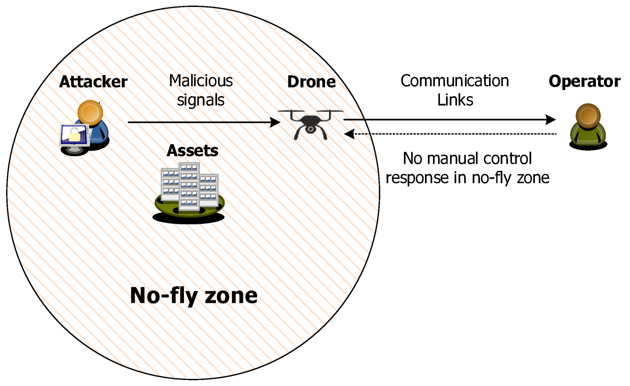


Fig. 1: Attack Model.

estimation methods, understand their control and navigation algorithms for autopilot, and obtain their default common parameters. Therefore, we can identify the limits of these algorithms. Consequently, we have compromised multiple types of drones through cyber attacks or physical attacks.

Our Research Focus. In this paper, we focus on remotely manipulating the actual position of a drone, by compromising state estimation algorithms and autopilot controllers (such as heading, velocity, or attitude). As sensors are usually subject to occasional errors, most control systems use state estimation algorithms to deal with such errors. With an extensive literature review, we confirm that Kalman Filter (KF) and its variants are the most popular estimation algorithms in drone control systems. Therefore, we choose it as our attack target. Furthermore, based on the position estimation, a drone navigation system usually uses an autopilot controller to manage its actual position in real time such that it follows a given flight path, and automatically guides it to a desired destination. Therefore, we further investigate drone autopilot controllers to figure out generic methods to manipulate target drones' actual positions. To the best of our knowledge, this is the first work in this area.

Paper Organization and Our Main Contributions. The remainder of this paper is organized as follows. We first introduce related work in Sec. 2. In Sec. 3, we introduce the common control loop in drone systems, especially the KF-based position estimation methods and autopilot controller in detail. In Sec. 4, we propose several attacks on drone position estimation methods and autopilot controllers. First, we propose a maximum False Data Injection (FDI) attack on the most popular KF-based estimation approaches; we then provide a detailed analysis of the proposed attack, and further design a generic FDI attack. Furthermore, we design two attacks on autopilot-based navigation, which ultimately affects the actual position of a drone. In Sec. 5, we evaluate the proposed attacks via simulations. Our simulation results have shown that the proposed attacks can significantly affect the position estimation and the actual position of a drone. In Sec. 6, we conclude this paper and point out our future research directions.

II. Related Work

A. Position Estimation Algorithms

Position estimation is critical to drone control and navigation systems. The most common sensors on consumer drones include standard IMUs (accelerometers, gyroscopes, and magnetometers in some cases), GPS, and barometers. As sensors may experience errors, drone control systems usually use state estimation methods to provide more accurate estimations. In general, KF and its variations (e.g., Extended KF, or simply EKF) are the most commonly used state estimation methods [9], [10]. These methods usually give us fairly accurate state estimations. A recent advanced KF framework [11], "Sigma-Point Kalman Filter", provides both a higher accuracy and lower computational overhead, compared to the widely-used EKF. A few other methods are also proposed to obtain altitude estimations based on various sensors (e.g., using camera images [12], or differential GPS and ultrasonic sensors [13]). However, because they are not commonly used on consumer drones, these methods are out of the scope of this research.

B. Existing False Data Injection (FDI) on State Estimation

FDI attacks have been proposed on dynamic systems, through manipulating state estimations via modifying corresponding measurements without being detected by *bad data detectors*. FDI attacks against state estimation have been examined in various control systems. Liu et al. first proposed a FDI attack against the state estimation in power grids [14]. Several following-up efforts have been made in this area [15], [16]. However, such state estimation model is different from the one in drone navigation systems. Recently theoretical analyses have been conducted in the area of FDI attack against KF-based state estimation [17]. However, their results cannot be applied to our research, because their model has very strict assumptions.

C. Autopilot Controller in Drones

An autopilot controller is essential for drone navigation systems. Common control strategies include the following methods [18]. First, the *PID control* is the most popular control strategy, which uses a control feedback mechanism to dynamically adjust control inputs according to the differences between outputs and setpoints. It is also one of the simplest control strategies and usually achieves an acceptable performance. However, it is not very robust, and its parameter adjustments may require extra efforts. Second, *Neural-Network (NN)-based adaptive control* uses neural networks to direct adaptive control system, e.g., an NN-based autopilot algorithm is presented for unmanned helicopter control [19], which also works for fixed-wing UASs. Furthermore, *Fuzzy-based autopilot* takes advantage of the recent development of fuzzy logic control systems, e.g., three fuzzy controllers are proposed to make a drone follow a pre-set flight path under wind disturbance [20]. Lastly, *LQG/LTR & H_∞ based autopilot* are also proposed. Although the PID control and NN-based adaptive control do not require an accurate system model, they may be not very robust. As a result, model-based controllers have been developed. Because drone flight control systems are usually non-linear, we often use linear models to approximate the non-linear models. While a model-based controller with linear quadratic regulator (LQR) Controller is proposed [21] for effective drone altitude control, a controller using a combination of dynamic inversion and H_∞ loop shaping is proposed to solve the same problem [22].

III. Models and Problems Formulation

A. Control Loop in Drone Systems

The control loop in a drone is illustrated in Fig. 2, which is a real-time automatic control without manual inputs.

Specifically, the system first performs state estimation based on sensor measurements. Then according to the current state estimation, the autopilot controller takes control actions to automatically adjust the drone movement. Given the auto-control actions, the actuators take corresponding adjustments.

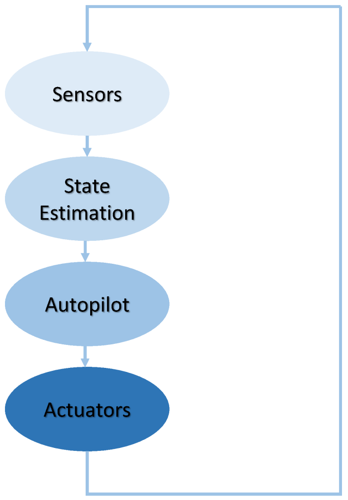


Fig. 2: Drone Control Loop.

In this paper, our goal is to deceive the drones in a controlled manner, e.g., redirecting it to a specific location. According to the control loop, we divide the *deception attacks* into three phases: the **first-phase attacks** focus on compromising sensor readings. Once attackers are able to manipulate sensor readings, in the **second-phase attacks**, they will try to fool state estimation algorithms. As long as the state estimation is successfully compromised, the attackers can then manipulate the drone movement determined by the autopilot controllers in the **third-phase attacks**. In this paper, we assume that we can compromise sensors via various attacks in order to deceive drones; we do not assume that we can remotely tamper actuators. Attacking sensor readings has been an active research area in recent years. The state-of-the-art work achieves good results in manipulation of IMU sensors [1], [2], GPS [23], [24], etc. Therefore, in this paper, we assume that such first-phase attacks help us compromise sensors, and will focus on the **second** and **third** phase attacks.

B. KF-based State Estimation Methods

In this paper, we use algorithms employed by ArduPilot as our targets, which is one of the most popular open source flight control systems. It uses EKF to estimate 24 states related to drone navigation, including velocity, position, accelerator bias, etc. The 24 states are divided into three categories: velocity and position state estimations, magnetic state estimations, and airspeed estimations. In this paper, we will investigate attacks on position state estimations.

The procedure of KF-based position estimation in ArduPilot is as follows:

Prediction:

$$\begin{aligned} \mathbf{x}_k^- &= \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_{k-1}, \\ \mathbf{P}_k^- &= \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^T + \mathbf{Q}_{k-1}, \end{aligned} \tag{1}$$

Update:

$$\begin{aligned} \Delta_k &= \mathbf{z}_k - \mathbf{H}\mathbf{x}_k^-, \\ \mathbf{K}_k &= \mathbf{P}_k^-\mathbf{H}^T(\mathbf{H}\mathbf{P}_k^-\mathbf{H}^T + \mathbf{R}_k)^{-1}, \\ \mathbf{x}_k &= \mathbf{x}_k^- + \mathbf{K}_k\Delta_k, \\ \mathbf{P}_k &= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^-. \end{aligned} \tag{2}$$

Here the state is defined as:

$$\mathbf{x}_k = \begin{pmatrix} \text{EastSpeed} \\ \text{NorthSpeed} \\ \text{UpSpeed} \\ \text{EastPosition} \\ \text{NorthPosition} \\ \text{Altutide} \end{pmatrix},$$

and we define the measurement vector as:

$$\mathbf{z}_k = \begin{pmatrix} \text{East position measurement} \\ \text{North position measurement} \\ \text{Altitude measurement} \end{pmatrix},$$

where the north and east position measurements come from the GPS, and the altitude measurements comes from either GPS or barometer.

$\mathbf{A}$ is commonly referred to as the state transition matrix, and $\mathbf{H}$ is the observation (or measurement) matrix, which maps system states to sensor measurements. The superscript $^-$ indicates the corresponding vector or variable that is predicted based on the estimated system states in the previous time interval. The state transition matrix $\mathbf{A}$ here is

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ \Delta t & 0 & 0 & 1 & 0 & 0 \\ 0 & \Delta t & 0 & 0 & 1 & 0 \\ 0 & 0 & \Delta t & 0 & 0 & 1 \end{pmatrix}.$$

Also, $\mathbf{u}_{k-1}$ is the acceleration vector from an accelerometer:

$$\mathbf{u}_{k-1} = \begin{pmatrix} \text{East Acceleration} \\ \text{North Acceleration} \\ \text{Up Acceleration} \end{pmatrix}.$$

$\mathbf{B}$ is define as

$$\mathbf{B} = \begin{pmatrix} \Delta t & 0 & 0 \\ 0 & \Delta t & 0 \\ 0 & 0 & \Delta t \\ \frac{\Delta t^2}{2} & 0 & 0 \\ 0 & \frac{\Delta t^2}{2} & 0 \\ 0 & 0 & \frac{\Delta t^2}{2} \end{pmatrix}.$$

We can also easily derive $\mathbf{H}$ from the relationship between $\mathbf{x}_k$ and $\mathbf{z}_k$:

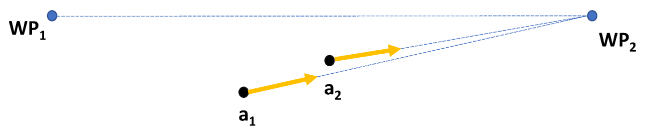


Fig. 3: Basic autopilot algorithm.

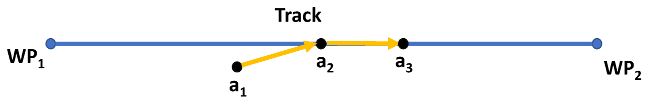


Fig. 4: Fly along a linear track.

$$\mathbf{H} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

$\mathbf{Q}_k$ and $\mathbf{R}_k$ are the process and measurement noise covariances, respectively. $\mathbf{R}_k$ are determined by the magnitude of navigation acceleration.

C. Autopilot Controller in ArduPilot

In this section, we introduce the basic mechanism used by the autopilot controllers in the control loop. Given the state estimations, the autopilot controllers need to adjust the actions of drones automatically to make sure that the drone follows a pre-set flight path or fly through specific waypoints.

In this paper, we consider the popular open-source ArduPilot in our investigation. In ArduPilot, they use the PID method as the autopilot controller. In particular, they provide two autopilot algorithms for two different goals as follows:

(1) The *basic autopilot algorithm* is the fundamental auto-navigation control algorithm. As Fig. 3 shows, in this method, the controller calculates the target bearing in every time step based on the current position of the drone and the target position. On receiving the target bearing information, the controller will navigate the drone to move towards the target with this bearing and a desired velocity.

(2) A *linear track-based navigation algorithm* leads a drone to a target location following the linear track from an origin to a destination, as shown in Fig. 4. To make sure that the drone flies along the track, the controller will calculate the track error (the distance between the drone's current position and the track) in every time step, and restrict the error within a small range. As a result, a drone will not deviate from the track too far. Usually, the rate to correct the track error (i.e., moving the drone back to the track) is stable as a constant. To lead the drone to reach the destination, the controller will calculate the drone's speed along the track and stabilize it at a desired one.

IV. Proposed Attacks

In this section, we focus on designing general methods to compromise drone positions. In particular, we first present two **second-phase attacks** on KF-based position estimation. Based on these attacks, we propose two **third-phase attacks** on the two autopilot algorithms used in ArduPilot, which allows the attackers to compromise the positions of a drone. In addition, we also propose countermeasures for the second-phase attacks, which can then prevent the third-phase attacks.

We assume that other first-phase attacks (cyber attacks or physical attacks) can help us compromise drone sensor readings, such that we can feed "manipulated" data to drone control algorithms in order to disrupt drone navigation control. Recently, several attacks have been developed to compromise the readings of MEMS sensors, by exploiting physical vulnerabilities. In [1], the authors showed that an adversary can incapacitate the MEMS gyroscopes of a drone, by crafting acoustic signals with the same resonant frequencies of gyroscopes to degrade their performance. In [2], the authors further developed acoustic injection attacks on MEMS accelerometers. Moreover, manipulating GPS readings [23], [24] has been conducted in field tests. We are fully aware that this assumption is fairly strong and itself could be a significant effort. Therefore, in this paper, we focus on the aftermath of such attacks, i.e., how to exploit the weakness of common navigation and control algorithms. In the meantime, we are also investigating potential attacks on GPS and MEMS barometer sensors used on drones, following other MEMS research [25].

A. Second-Phase Attacks on KF-based Position Estimation

As in the popular open-source ArduPilot, assume the KF-based position estimation uses an anomaly detection algorithm to check if the following condition is true:

$$innovation(i)^2/varInnov(i) \leq \tau, \tag{3}$$

for i= 1, 2, 3 (for three dimensions of $\mathbf{z}_k$). Here τ is a pre-set threshold, $innovation(i) = \mathbf{z}_k(i) - \mathbf{H}\mathbf{x}_k^-(i)$ is the difference between a prediction and a measurement, and $varInnov(i)$ is the variance of $innovation(i)$. We assume that the system has reached a steady state before we perform the attack. Therefore, we can consider $varInnov(i)$ as a constant c_{var}. Then, eq. (3) can be simplified as the following:

$$\left|\mathbf{z}_k(i) - \mathbf{H}\mathbf{x}_k^-(i)\right| \leq \lambda_{max}, \tag{4}$$

where $\lambda_{max} = \sqrt{c_{var} \cdot \tau}$ is a constant parameter. With the above settings, our goal is to maximize the deviation of state estimations without being detected. We call this attack as *maximum False Data Injection (FDI) attack*. As one realization of this attack, the attacker can make the compromised estimation as large as possible.

To achieve this attack, in every time slot k, the malicious measurements $\mathbf{z}'_k(i)$ is set as follows:

$$\mathbf{z}'_k(i) = \mathbf{H}\mathbf{x}_k^-(i) + \lambda_{max}. \tag{5}$$

This is a simple attack with a specific goal. In the following, we will analyze the attack effects. Based on the analysis, we develop the second attack – *generic FDI attack*.

Since it has reached the steady state, $\mathbf{K}$ has converged to a constant. Then, in the maximum FDI attack, according to eq. (1),(2), and (5), we have

$$\begin{aligned}\mathbf{x}_k(4:6) =& \mathbf{A}\mathbf{x}_{k-1}(4:6) + \mathbf{B}\mathbf{u}_{k-1}(4:6) \\ &+ \mathbf{K}\begin{pmatrix}\lambda_{max}\\ \lambda_{max}\\ \lambda_{max}\end{pmatrix}(4:6).\end{aligned} \tag{6}$$

To simplify our analysis, we consider the accelerometer readings $\mathbf{u}_k$ as constants. Then, we have

$$\begin{aligned}\mathbf{x}_k - \mathbf{x}_{k-1} &= \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_{k-1} + \mathbf{K}\lambda_{max} - \mathbf{x}_{k-1}, \\ &= \begin{pmatrix}0\\0\\0\\ \mathbf{u}_k(1)\cdot(k-1)\Delta t\cdot\Delta t\\ \mathbf{u}_k(2)\cdot(k-1)\Delta t\cdot\Delta t\\ \mathbf{u}_k(3)\cdot(k-1)\Delta t\cdot\Delta t\end{pmatrix} + \mathbf{K}\begin{pmatrix}\lambda_{max}\\ \lambda_{max}\\ \lambda_{max}\end{pmatrix} + \mathbf{B}\mathbf{u}_{k-1},\end{aligned} \tag{7}$$

where $\mathbf{B}\mathbf{u}_k$, Δt, $\mathbf{K}$, λ_{max} are approximated as constants. In the above equation, we remove the notation "$(4:6)$" from each term for clarity.

This result shows that $\mathbf{x}_k - \mathbf{x}_{k-1}$ is linear, so $\mathbf{x}_k$ would be quadratic. Eq. (7) also gives us the expression for $\mathbf{x}_k$. In particular, from eq. (7), we have:

$$\begin{aligned}\mathbf{x}_k - \mathbf{x}_{k-1} &= \begin{pmatrix}\cdots\\ \mathbf{u}_k(1)\cdot(k-1)\Delta t\cdot\Delta t\\ \mathbf{u}_k(2)\cdot(k-1)\Delta t\cdot\Delta t\\ \mathbf{u}_k(3)\cdot(k-1)\Delta t\cdot\Delta t\end{pmatrix} + \mathbf{K}\begin{pmatrix}\lambda_{max}\\ \lambda_{max}\\ \lambda_{max}\end{pmatrix} \\ &+ \mathbf{B}\mathbf{u}_{k-1}; \\ \mathbf{x}_{k-1} - \mathbf{x}_{k-2} &= \begin{pmatrix}\cdots\\ \mathbf{u}_{k-1}(1)\cdot(k-2)\Delta t\cdot\Delta t\\ \mathbf{u}_{k-1}(2)\cdot(k-2)\Delta t\cdot\Delta t\\ \mathbf{u}_{k-1}(3)\cdot(k-2)\Delta t\cdot\Delta t\end{pmatrix} + \mathbf{K}\begin{pmatrix}\lambda_{max}\\ \lambda_{max}\\ \lambda_{max}\end{pmatrix} \\ &+ \mathbf{B}\mathbf{u}_{k-2}; \\ &\cdots \\ \mathbf{x}_1 - \mathbf{x}_0 &= \begin{pmatrix}\cdots\\0\\0\\0\end{pmatrix} + \mathbf{K}\begin{pmatrix}\lambda_{max}\\ \lambda_{max}\\ \lambda_{max}\end{pmatrix} + \mathbf{B}\mathbf{u}_0.\end{aligned}$$

Summing up the above equations, we have:

$$\begin{aligned}\mathbf{x}_k &= \mathbf{x}_0 + \sum_{i=1}^{k}\mathbf{x}_i - \mathbf{x}_{i-1}, \\ &= \mathbf{x}_0 + \begin{pmatrix}\cdots\\ \frac{k^2}{2}\Delta^2 t\cdot\mathbf{u}(1) + k\cdot\mathbf{K}\begin{pmatrix}\lambda_{max}\\ \lambda_{max}\\ \lambda_{max}\end{pmatrix}(4)\\ \frac{k^2}{2}\Delta^2 t\cdot\mathbf{u}(2) + k\cdot\mathbf{K}\begin{pmatrix}\lambda_{max}\\ \lambda_{max}\\ \lambda_{max}\end{pmatrix}(5)\\ \frac{k^2}{2}\Delta^2 t\cdot\mathbf{u}(3) + k\cdot\mathbf{K}\begin{pmatrix}\lambda_{max}\\ \lambda_{max}\\ \lambda_{max}\end{pmatrix}(6)\end{pmatrix}.\end{aligned} \tag{8}$$

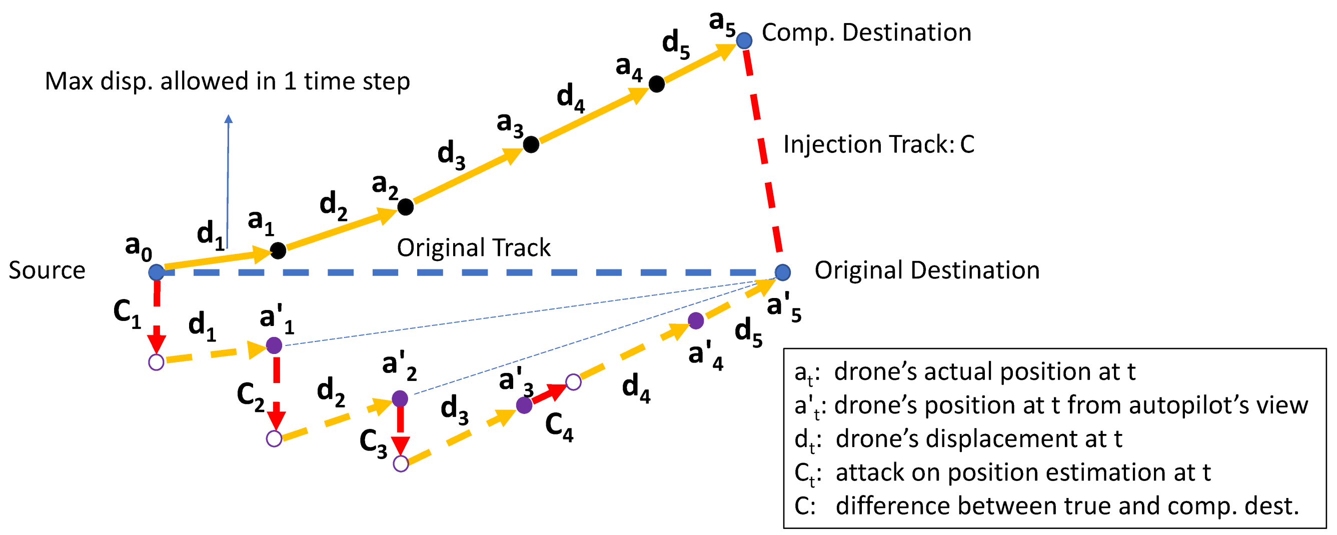
With this result, we can achieve a more general purpose attack – *generic FDI attack*: in this attack, assuming the attacker starts performing the attack at time t. Our goal is to make the state value $\mathbf{x}_{t+n}$ to be certain value ξ at time $t+n$, after n time cycles.

To further simplify the attack, we assume starting time t to be 0. In eq. (8), $\mathbf{x}_0$ and $\mathbf{K}$ are known to attackers, Δt is usually set to be one time unit, e.g., 100 milliseconds. Then we have two options to achieve the attack goals: compromising $\mathbf{u}$ or λ_{max}. Modifying λ_{max} relies on manipulating GPS or barometer readings (according to eq. (5)), and modifying $\mathbf{u}$ means the manipulation of accelerometer readings. As a result, by modifying GPS/barometer readings and/or accelerometer readings, we can achieve the generic FDI attack based on eq. (8).

If the starting time t is not zero, we can achieve the generic FDI attack based on the following equation:

$$\begin{aligned}\mathbf{x}_{t+n} &= \mathbf{x}_t + \sum_{i=1}^{n}\mathbf{x}_{t+i} - \mathbf{x}_{t+i-1}, \\ &= \mathbf{x}_t + \sum_{i=1}^{n} \\ &\left\{\begin{pmatrix}\cdots\\ \mathbf{u}(1)\cdot(t+i-1)\Delta^2 t\\ \mathbf{u}(2)\cdot(t+i-1)\Delta^2 t\\ \mathbf{u}(3)\cdot(t+i-1)\Delta^2 t\end{pmatrix} + \mathbf{K}\begin{pmatrix}\lambda_{max}\\ \lambda_{max}\\ \lambda_{max}\end{pmatrix} + \mathbf{B}\mathbf{u}\right\}, \\ &= \mathbf{x}_t + \begin{pmatrix}\cdots\\ \frac{(2t+n)n}{2}\Delta^2 t\cdot\mathbf{u}(1) + n\cdot\mathbf{K}\begin{pmatrix}\lambda_{max}\\ \lambda_{max}\\ \lambda_{max}\end{pmatrix}(4)\\ \frac{(2t+n)n}{2}\Delta^2 t\cdot\mathbf{u}(2) + n\cdot\mathbf{K}\begin{pmatrix}\lambda_{max}\\ \lambda_{max}\\ \lambda_{max}\end{pmatrix}(5)\\ \frac{(2t+n)n}{2}\Delta^2 t\cdot\mathbf{u}(3) + n\cdot\mathbf{K}\begin{pmatrix}\lambda_{max}\\ \lambda_{max}\\ \lambda_{max}\end{pmatrix}(6)\end{pmatrix}.\end{aligned} \tag{9}$$

This equation shows that the compromised estimation $\mathbf{x}_{t+n}$ will also depend on the starting time t.



Max disp. allowed in 1 time step
Comp. Destination
Injection Track: C
Original Track
Source
Original Destination
a0
a1
a2
a3
a4
a5
d1
d2
d3
d4
d5
C1
C2
C3
C4
a'1
a'2
a'3
a'4
a'5
a_t: drone's actual position at t
a'_t: drone's position at t from autopilot's view
d_t: drone's displacement at t
C_t: attack on position estimation at t
C: difference between true and comp. dest.

Fig. 5: Illustration of attacking the basic autopilot algorithm.

B. *Third-Phase Attacks on Autopilot-based Navigation*

In this section, assume we can successfully performed the aforementioned generic FDI attack on state estimation, i.e., we can manipulate the position estimation to any specific value within a range. We then develop the following attacks on autopilot-based navigation, which can ultimately affect the target drone's actual position.

1) Attacking the Basic Autopilot Algorithm: Although the strategy of the basic autopilot algorithm is straightforward, it is not easy to compromise in comparison with the advanced track-based autopilot. The reason is that in each time step when a drone moves, it calculates a new bearing information, which is difficult to predict and manipulate. In this paper, we use a novel algorithm to find the strategy of attacking the basic autopilot algorithm. The main idea of this attack is: by attacking the position estimation, we continuously adjust the moving direction of the drone little by little, until that the drone exactly points to the compromised destination. As long as the bearing has been determined, what the attackers need to do is to adjust the remaining flight distance so that it can reach the compromised destination precisely. In the following, we use Fig. 5 to show the proposed attack algorithm in detail.

As shown in Fig. 5, we capture the 2D-plane that contains the source, the original destination, and the compromised destination. Since the compromised destination is above the original destination in this case, we apply a downward position injection such that the direction of the displacement points to the above. Here the optimal strategy is to make the position injection vector $\mathbf{C}_1$ vertically downward, with the magnitude of the maximum value that it can achieves in the second-phase attack. This strategy will maximize the bearing change. After this time step, the autopilot finds the drone at a new position. Then it calculates the current bearing, and moves towards the "destination" with the maximum displacement allowed in one time step. Now the drone believes that it is at $\mathbf{a}'_1$, but actually at $\mathbf{a}_1$. In the following, the attackers repeat the above process until the drone is exactly towards the compromised destination. Note that the magnitude of the last injection should be tuned such that we get the perfect bearing for the drone. Next, we need an additional injection $\mathbf{C}_4$ with the same or opposite moving direction of the drone, to cover the remaining flight distance. More specifically, $\mathbf{C}_4 = \mathbf{C} - \mathbf{C}_1 - \mathbf{C}_2 - \mathbf{C}_3$ (note that here is the vector arithmetic). In the end, the drone reaches the compromised destination $\mathbf{a}_5$, but it believes that it arrives at the original destination $\mathbf{a}'_5$.

2) Attacking the Linear Track-based AutoPilot Algorithm: If the desired flight track is linear, the position error caused by the proposed FDI attacks is easy to figure out: since any "position drift" (due to the injection on the estimation by the attack) will be corrected by the autopilot controller, the sum of the total injections on position estimation should be equal to the inverse of the injection track (the difference between the original destination and the compromised destination). We discuss the algorithm that can manipulate the drone to a specific compromised destination in the following.

As shown in Fig. 6, we capture the 2D-plane that contains the source, the original destination and the compromised destination. In our attack, we first apply a position estimation injection $\mathbf{C}_1$ in time step 1. Without loss of generality, we let the magnitude of the injections $\mathbf{C}_t$ (except the last injection) be the maximum value that it can achieve in the second-phase attacks, and all the injection vectors be parallel to the injection track. After the injection, the autopilot controller believes that the drone has drifted away from the track and will try to make it return to the track again. It will let the drone make an adjustment $\mathbf{d}_1$, which is the sum of the maximum allowed displacement along the track and the maximum allowed displacement correction towards the track in one time step. As a result, the drone reaches $\mathbf{a}_1$ but believes that it is at $\mathbf{a}'_1$. Then, we apply other two injections $\mathbf{C}_2$ and $\mathbf{C}_3$ on the position estimation such that $\mathbf{C} = \mathbf{C}_1 + \mathbf{C}_2 + \mathbf{C}_3$ (note that here is the vector arithmetic), and the controller will try to correct the drone's position with displacement $\mathbf{d}_2$ and $\mathbf{d}_3$. By now, we have finished all the injections

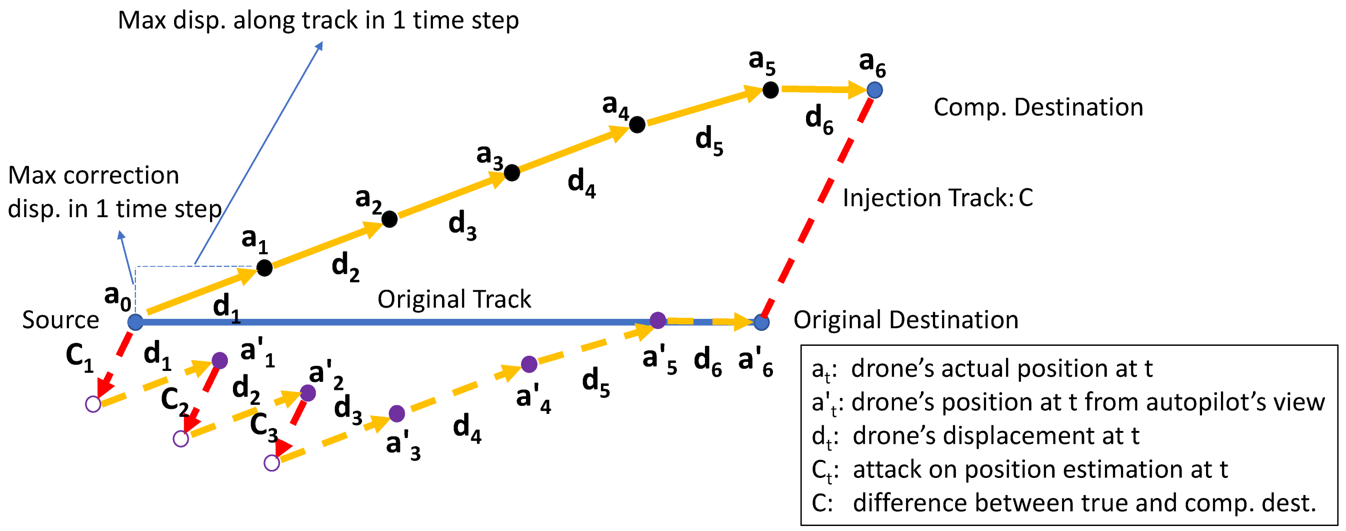

Max disp. along track in 1 time step
Max correction disp. in 1 time step
Source
Comp. Destination
Injection Track: C
Original Track
Original Destination
a_0 a_1 a_2 a_3 a_4 a_5 a_6
d_1 d_2 d_3 d_4 d_5 d_6
C_1 C_2 C_3
a'_1 a'_2 a'_3 a'_4 a'_5 a'_6
a_t: drone's actual position at t
a'_t: drone's position at t from autopilot's view
d_t: drone's displacement at t
C_t: attack on position estimation at t
C: difference between true and comp. dest.

Fig. 6: Illustration for attacking the linear track autopilot algorithm.

on position estimation. Therefore, we will let the autopilot controller continuously correct the drone's flight path and move along the "track" to the "destination". Finally, the drone arrives at the compromised destination $\mathbf{a}_6$, but believes that it reaches the original destination $\mathbf{a}'_6$.

C. Countermeasures

As discussed in the above, the anomaly detection algorithm in ArduPilot cannot detect the proposed FDI attacks on the KF-based position estimation. Another advanced detection method - chi-squared test, has been widely applied to defend against FDI attacks on KF-based estimation. This type of detector may work well against the proposed FDI attacks. However, it usually requires significant computational overheads and makes it impractical to be applied in simple drone flight control systems. In this subsection, we propose a novel detector that can effectively detect the proposed FDI attacks against position estimation with low computational overheads, which can then prevent the attacks on autopilot algorithms.

The new detector is designed based on the statistical characteristics among the innovations in different time steps. In particular, we know that the innovation Δ_k in time step k follows Gaussian distribution:

$$\Delta_k(i) = \mathbf{z}_k(i) - \mathbf{H}\mathbf{x}_k^-(i) \sim N(0, \mathbf{R}_k(i,i)), \tag{10}$$

where $\mathbf{R}_k$ is the measurement covariance matrix (the innovation Δ_k under the FDI attacks, however, usually not follows this distribution). In addition, the innovations in different time steps can be regarded as independent. Then by the Chebyshev's Inequality, for each dimension in Δ_k, we have:

$$P(\left|\frac{\sum_{i=1}^{k}\Delta_i(i)}{k}\right| < \epsilon) \geq 1 - \frac{\mathbf{R}_k(i,i)}{k^2 \cdot \epsilon^2}, \tag{11}$$

for i=1, 2, 3 (which corresponds to three dimensions of the innovation vector Δ_k, respectively). If we let ϵ be $\frac{\tau_{pd}}{k}$, then eq. (11) becomes:

$$P(\left|\frac{\sum_{i=1}^{k}\Delta_i(i)}{k}\right| < \frac{\tau_{pd}}{k}) \geq 1 - \frac{\mathbf{R}_k(i,i)}{\tau_{pd}^2}, \tag{12}$$

where τ_{pd} is a pre-set threshold. As long as τ_{pd} is large enough, $P(\left|\frac{\sum_{i=1}^{k}\Delta_i(i)}{k}\right| < \frac{\tau_{pd}}{k}) \approx 1$. Based on this observation, we let the proposed detector check whether the following statement is true:

$$\left|\frac{\sum_{i=1}^{k}\Delta_i(i)}{k}\right| < \frac{\tau_{pd}}{k}. \tag{13}$$

To implement this detector, in each time step in the estimation procedure, we calculate an additional variable - the mean of the innovation $\bar{\Delta}_k$ as:

$$\bar{\Delta}_k(i) = \begin{cases} \Delta_1(i) & \text{if } k = 1; \\ \frac{\bar{\Delta}_{k-1}(i)*(k-1)+\Delta_k(i)}{k} & \text{if } k \geq 2. \end{cases} \tag{14}$$

In this implementation, the computational overhead increases very little.

In the following, we use the maximum FDI attack as a simple example to test the effectiveness of the proposed detector. According to eq. (5), $\Delta_i = \lambda_{max}$ for $i = 1, 2, \ldots$. Then at time step k, the proposed detector checks whether the following statement is true:

$$\left|\frac{\sum_{i=1}^{k}\Delta_i(i)}{k}\right| = \lambda_{max} < \frac{\tau_{pd}}{k}. \tag{15}$$

In this case, when the time $k \geq \frac{\tau_{pd}}{\lambda_{max}}$, the detector will be alerted.

V. Simulation Evaluation

In this section, we evaluate the proposed second-phase and third-phase attacks with simulations.

A. Second-Phase Attacks on KF-based Position Estimation

In this subsection, we use ArduPilot to evaluate the proposed attacks on KF-based position estimation. Without loss

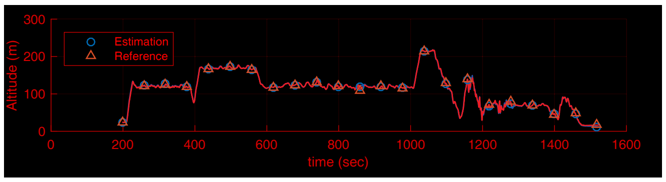


Fig. 7: Altitude estimation before attack.

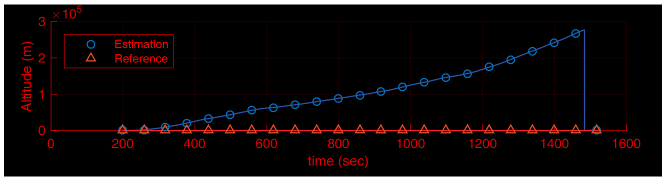


Fig. 8: Altitude estimation after attack.

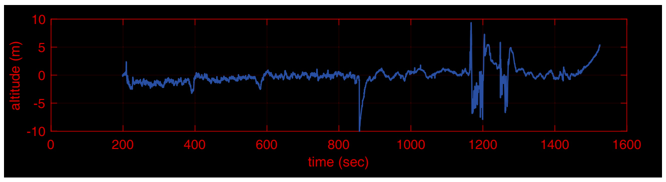


Fig. 9: Innovation before attack.

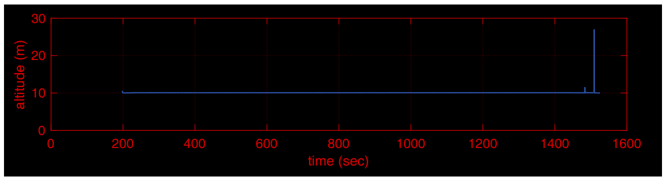


Fig. 10: Innovation after attack.

of generality, we choose one dimension of drone poistion - **altitude** as the attack target.

Simulation Settings: The simulations are based on the altitude estimation algorithm of the ArduPilot project [26]. We mainly compare the altitude estimations before and after the proposed attacks.

Our simulations are performed on `TestLog3` data given in [26]. In the experiments, without loss of generality, we perform the maximum FDI attack to make the attack effects more legible. The default threshold τ in the ArduPilot code is pre-set to be 5. In addition, all the sensors remain operational except the airspeed module, which is disabled manually to show the attack effects.

In this simulation, for simple illustration, we set the vertical acceleration to be 0, where the altitude estimation is expected to increase linearly according to eq. (8). In addition, we force the flight control system to perform altitude estimation at each time interval. (Under default settings, the estimation may not proceed at certain time steps if some measurements are missing.) In the simulations, we performed the attacks through the entire estimation process.

Simulation Results: Fig. 7 and Fig. 8 depict the altitude estimations before and after the maximum FDI attack, respectively. The x-axis shows the simulation time and the y-axis shows the altitude and the altitude estimate. In the figures, the blue line shows the estimations of altitude while the red line represents the actual values. In Fig. 7, without attacks, two lines mostly overlap. However, in Fig. 8, with the attacks, the estimation of altitude surges. In addition, the compromised estimations increase almost linearly. The reason that they are not strictly linear is: even though we set the vertical acceleration measurements as 0, its estimation may not be necessarily 0, which causes small fluctuations. Furthermore, we can also validate the effectiveness of our attack from the altitude innovation data. From Fig. 9 and Fig. 10, we can see that the innovation of the altitude remains constant during the entire estimation process when under attack, which is as expected.

B. Third-Phase Attacks on Autopilot Controllers

In this subsection, we will evaluate the proposed attacks on autopilot controllers. Specifically, we conduct two simulations to verify if the proposed attacks on the basic autopilot algorithm and the linear track-based autopilot algorithm can achieve the desired goal.

Simulation Settings: The attack simulator is written in MATLAB. In this simulation, we assume that we are able to manipulated the position estimation. The injection on the position estimation is set to be at most 5 m/timestep. In addition, the noise of each state estimation is assumed to follow $\mathcal{N}(0, 0.01)$. In each attack, the source and the original destination is set to be (50 m, 100 m) and (150 m, 100 m) in a 2D plane. We would like to guide the target drone to the compromised destination (150 m, 75 m). For the attack on the basic autopilot algorithm, the target drone's maximum speed allowed is assumed to be 10 m/timestep; and for the attack on the track-based autopilot algorithm, we assume that the maximum speed along the track and towards the track for track error correction are 10 m/timestep and 2.5 m/timestep, respectively.

Simulation Results: Fig. 11 shows the simulation results of attacks on the basic autopilot algorithm. The red line with circles shows the flight path from the system's view while the black line with triangles represents the actual path. At first, the attackers put a maximum positive injection along the y-axis. Then the autopilot controller will direct the drone to fly towards the "destination", which makes it move towards bottom right. However, the current bearing is not large enough for the compromised destination. Therefore, the attackers will keep the same action in the next four time steps. When the drone eventually aims exactly at the compromised destination after the $5th$ time step, we will stop injections, and let the drone fly directly towards the destination without changing bearings in the following time steps. In the end, we can find that the drone have almost reached the compromised destination, which is as expected. Similarly, Fig. 12 shows the attack results on the linear track-based autopilot algorithm: we can also direct the target drone arriving at the desired location.

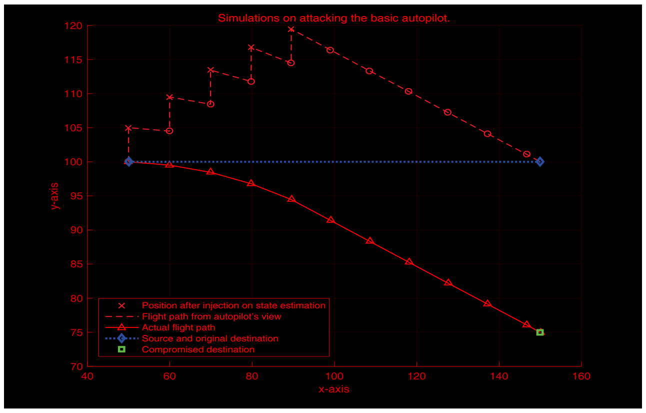


Fig. 11: Simulation of attacking the basic autopilot algorithm.

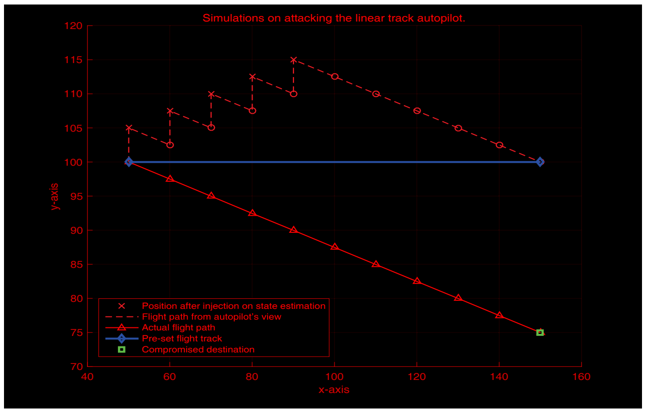


Fig. 12: Simulation of attacking the linear track autopilot algorithm.

VI. Conclusion

In this paper, we have carefully investigated position estimation methods and autopilot algorithms used in the popular open-source ArduPilot and identified potential attacks to compromise position estimations and manipulate drone actual positions. We have developed two attacks on the popular estimation techniques and further proposed two attacks on the autopilot controller to manipulate drone positions. Our simulation results have shown that the proposed attacks can significantly affect the position estimation and the actual positions of a drone. We are currently working formulating a theoretical framework to conduct thorough investigation such attacks and analyze its capabilities and limitations.

Acknowledgement: We like to thank the support of NSF 1662487 for this research. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] Y. Son, H. Shin, D. Kim, Y.-S. Park, J. Noh, K. Choi, J. Choi, Y. Kim *et al.*, "Rocking drones with intentional sound noise on gyroscopic sensors."

[2] T. Trippel, O. Weisse, W. Xu, P. Honeyman, and K. Fu, "Walnut: Waging doubt on the integrity of mems accelerometers with acoustic injection attacks."

[3] J. Vanian, "Drone registrations are still soaring," *Fortune, http://fortune.com/2017/01/06/drones-registrations-soaring-faa/*, Jan. 06, 2017.

[4] M. A.H. and D. Gettinger, "Analysis of new drone incident reports," May 8, 2017, http://dronecenter.bard.edu/analysis-3-25-faa-incidents/.

[5] M. Schmidt and M. Shear, "A drone, too small for radar to detect, rattles the white house," *New York Times, https://www.nytimes.com/2015/01/27/us/white-house-drone.html*, Jan. 26, 2015.

[6] M. Benyamin and G. Goldman, "Acoustic Detection and Tracking of a Class I UAS with a Small Tetrahedral Microphone Array," ARL, Tech. Rep. ARL-TR-7086, 2014.

[7] D. Labs, "Drone detector," http://www.dronedetector.com/how-drone-detection-works/, 2016.

[8] DeDrone, "Secure your airspace now," http://www.dedrone.com/en/dronetracker/drone-protection-software, 2016.

[9] T. Delbrügger, "Altitude sensor fusion," https://timdelbruegger.wordpress.com/2016/01/05/altitude-sensor-fusion/.

[10] P. Gasior, S. Gardecki, J. Goslinski, and W. Giernacki, "Estimation of altitude and vertical velocity for multirotor aerial vehicle using kalman filter," in *Recent Advances in Automation, Robotics and Measuring Techniques*. Springer, 2014, pp. 377–385.

[11] R. Van Der Merwe, E. A. Wan, S. Julier *et al.*, "Sigma-point kalman filters for nonlinear estimation and sensor-fusion: Applications to integrated navigation," 2004.

[12] D. Eynard, P. Vasseur, C. Demonceaux, and V. Fremont, "Real time uav altitude, attitude and motion estimation from hybrid stereovision," *Autonomous Robots*, vol. 33, no. 1-2, pp. 157–172, 2012.

[13] X. Lei and J. Li, "An adaptive altitude information fusion method for autonomous landing processes of small unmanned aerial rotorcraft," *Sensors*, vol. 12, no. 10, pp. 13 212–13 224, 2012.

[14] Y. Liu, P. Ning, and M. K. Reiter, "False data injection attacks against state estimation in electric power grids," in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, ser. CCS '09. New York, NY, USA: ACM, 2009, pp. 21–32. [Online]. Available: http://doi.acm.org/10.1145/1653662.1653666

[15] F. Pasqualetti, R. Carli, and F. Bullo, "A distributed method for state estimation and false data detection in power networks," in *Smart Grid Communications (SmartGridComm), 2011 IEEE International Conference on*. IEEE, 2011, pp. 469–474.

[16] T. T. Kim and H. V. Poor, "Strategic protection against data injection attacks on power grids," *IEEE Transactions on Smart Grid*, vol. 2, no. 2, pp. 326–333, 2011.

[17] R. Niu and L. Huie, "System state estimation in the presence of false information injection," in *Statistical Signal Processing Workshop (SSP), 2012 IEEE*. IEEE, 2012, pp. 385–388.

[18] H. Chao, Y. Cao, and Y. Chen, "Autopilots for small unmanned aerial vehicles: a survey," *International Journal of Control, Automation and Systems*, vol. 8, no. 1, pp. 36–44, 2010.

[19] E. Johnson and S. Kannan, "Adaptive flight control for an autonomous unmanned helicopter," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2002, p. 4439.

[20] M. Kumon, Y. Udo, H. Michihira, M. Nagata, I. Mizumoto, and Z. Iwai, "Autopilot system for kiteplane," *IEEE/ASME Transactions on Mechatronics*, vol. 11, no. 5, pp. 615–624, 2006.

[21] C. Hajiyev and S. Y. Vural, "Lqr controller with kalman estimator applied to uav longitudinal dynamics," *Positioning*, vol. 4, no. 1, p. 36, 2013.

[22] M. Sadraey and R. Colgren, "2 dof robust nonlinear autopilot design for a small uav using a combination of dynamic inversion and h-infinity loop shaping," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2005, p. 6402.

[23] N. O. Tippenhauer, C. Pöpper, K. B. Rasmussen, and S. Capkun, "On the requirements for successful gps spoofing attacks," in *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 2011, pp. 75–86.

[24] A. J. Kerns, D. P. Shepard, J. A. Bhatti, and T. E. Humphreys, "Unmanned aircraft capture and control via gps spoofing," *Journal of Field Robotics*, vol. 31, no. 4, pp. 617–636, 2014.

[25] D. E. Bolanakis, "Mems barometers in a wireless sensor network for position location applications," in *Applications of Commercial Sensors (VCACS), 2016 IEEE Virtual Conference on*. IEEE, 2016, pp. 1–8.

[26] P. Riseborough, "Inertial navigation filter," https://github.com/priseborough/InertialNav.