Accurately Redirecting a Malicious Drone

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

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

***Abstract*—Although some existing counterdrone measures can disrupt the invasion of certain consumer drone, to the best of our knowledge, none of them can accurately redirect it to a given location for defense. In this paper, we proposed a *Drone Position Manipulation (DPM)* attack to address this issue by utilizing the vulnerabilities of control and navigation algorithms used on consumer drones. As such drones usually depend on GPS for autopiloting, we carefully spoof GPS signals based on where we want to redirect a drone to, such that we indirectly affect its position estimates that are used by its navigation algorithm. By carefully manipulating these states, we make a drone gradually move to a path based on our requirements. This unique attack exploits the entire stack of sensing, state estimation, and navigation control together for quantitative manipulation of flight paths, different from all existing methods. In addition, we have formally analyzed the feasible range of redirected destinations for a given target. Our evaluation on open-source *ArduPilot* system shows that DPM is able to not only accurately lead a drone to a redirected destination but also achieve a large redirection range.**

***Index Terms*—drone security, UAS, navigation algorithm**

I. Introduction

Urgent Concerns. While consumer drones help us support many new applications, they have also been abused in many attacks. Therefore, it is urgent to develop effective drone countermeasures. We focus on consumer drones because of their low cost and broad deployment; we do not consider high-end drones due to their different resources and requirements. So, in this paper, we use *drones* to refer to *consumer drones*.

Motivations. Several direct physical methods for drone defense have been developed by industry in recent years, e.g., jamming a drone's control channels to trigger its fail-safe mode to land, or capturing it with a net. While such simple physical methods work well in many cases, they usually can not handle collateral damages. When a drone carries a bomb, we really do not want it to land in a protected area; instead, we would like to redirect it to a designated area for safe handling. While several projects [1]–[5] have demonstrated the feasibility of such attacks, none of them is able to provide concrete *quantitative control* on practical systems, which is the focus of this paper. Furthermore, as more robotic vehicles are developed for new applications, many similar security issues become serious concerns. In this paper, we propose to systematically address such issues by investigating the entire control stack to achieve accurate quantitative control for specific goals.

Ideally, we like to gain the complete control of an invading drone, e.g., by hacking into its control software or communications. While several methods have been developed to exploit specific drone settings, they require to compromise drone software, sensors, or communication channels [6]–[8], which are difficult to achieve in practice. While these methods can deal with weak systems with known vulnerabilities, we cannot solely rely on such methods, because the vulnerabilities may be easily patched.

Our method. Therefore, different from these methods, we will focus on a new challenge in this paper: *we would like to accurately control a drone without depending on compromising its software or hardware.* To achieve this goal, we propose a holistic approach to explore the entire stack of sensing, state estimation, and navigation together. First, almost all consumer drones depend on guidance inputs, e.g., civilian GPS. We can utilize existing software-defined radio (SDR) tools to spoof the signals to achieve our attacks. Furthermore, assume we can identify the type and model of an invading drone [9], [10], we can then find out its state estimation and navigation algorithms. As these algorithms are designed mainly for control without considering security concerns, we have carefully analyzed them and identified their guidance inputs as the attack surface. Therefore, we focus on carefully constructing spoofed GPS inputs to exploit both state estimation and navigation algorithms for manipulating a drone's position. Such a holistic solution allows us to integrate the vulnerabilities at three levels together and achieve accurate quantitative position control.

Although spoofing GPS to attack drones had been exploited in two other projects [1], [5], neither of them exploited the entire control stack as in this paper. While their methods indeed gained some control of the drone but did not achieve the accurate position control as the proposed *Drone Position Manipulation (DPM)* attack. Furthermore, although anecdotes on military GPS spoofing attacks have been reported, the details have never been revealed. So, we consider these attacks on military drones beyond the scope of this paper; our focus is the civilian GPS system on consumer drones.

Contributions. Our novel contributions in this paper include: (1) We develop a theoretical model to help us achieve accurate manipulation of a drone's position for specific goals, while existing methods were able to disrupt a drone's mission but did not define a clear model or achieve quantitative control. (2) The proposed attack exploits the entire stack of sensing,

state estimation, and navigation control, while existing methods mostly focused on one or two layers. (3) The proposed attack is validated on ArduPilot, arguably the most popular open-source flight control system, to show its effectiveness in practical settings; while existing methods are mostly evaluated on theoretical platforms. (4) The proposed attack does not require to compromise the software or hardware of a drone as some existing methods required.

To demonstrate the proposed attack, we evaluate it on the Software-in-the-loop (SITL) module of ArduPilot [11], which runs the same code as a firmware on a real drone. Our evaluation shows that the proposed attack is able to accurately lead a drone to a redirected destination. In addition, we have identified the range of feasible redirected destinations for a target to show the strong capability of DPM.

The proposed attack have broad impacts. First, existing projects and our own experiments have shown that we can use cheap SDR cards (e.g., BladeRF) to spoof a drone's GPS inputs. Although an enhanced GPS receiver may detect such spoofing attacks with extra hardware and software improvement, a common GPS receiver cannot detect carefully-crafted GPS spoofing. Furthermore, because the state estimation and navigation control algorithms on consumer drones are also broadly used in many other autonomous systems, understanding their weaknesses is also critical to secure those systems.

The remainder of this paper is organized as follows. In Section II, we will introduce our problem statement, common drone control algorithms, and GPS spoofing methods. We will then propose the DPM attack in Section III. We will present the performance evaluation in Section IV. We will discuss related work in Section V, and conclude this paper in Section VI.

II. Problem Statement and Drone Background

A. Problem Statement

As shown in Figure 1, we set up a restricted area around a critical asset to protect it from drone invasions. When a drone flies towards the critical asset, we need to redirect it away from the asset (its target destination) to a redirected destination for safe handling, e.g., guiding it into a blast containment chamber. Because we can usually recognize an invading drone with existing approaches (via radar, radio or traffic profiling, or image processing [9], [12]), we are able to identify its sensors (e.g., GPS) and firmware (including its state estimation and navigation algorithms). Assume that we have obtained the same model of drone and analyzed it ahead of time; utilizing the weaknesses of its GPS receiver, state estimation, and navigation algorithms, we propose a method to carefully spoof its GPS inputs to manipulate its position states such that its navigation control will change its path towards a redirected destination. As a result, our method does not require to compromise its software or hardware as some existing methods required.

To address this challenge, we need to carefully exploit the entire sensing, state estimation, and navigation control stack to achieve quantitative control on a practical system, different from existing methods [1], [5]. In particular, we need to solve

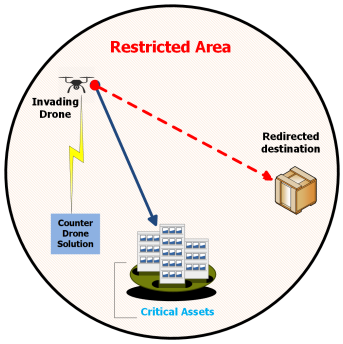

Fig. 1. Restricted Area around a critical asset.

the following problems: (1) We need to make a drone lock on our spoofed GPS signals. We will rely on existing methods to achieve this, e.g., using the covert attack proposed in [1]. In this paper, we show a method to spoof GPS inputs on the SITL simulation platform in order to understand how to spoof GPS signals without being detected. (2) We need to determine how to construct the spoofed GPS position inputs based on the drone's original flight path and the redirected destination, within a given attack duration. *This is the focus of this paper* as presented in the following.

We further divide the process of determining spoofed GPS signals into two steps: First, we will determine the shifting distance of the drone's position in each GPS cycle (i.e., 0.1 second) in order to make the navigation algorithm adjust its path towards the redirected destination during a given attack period. Second, we will construct the spoofed GPS inputs based on the shifting distance in a GPS cycle. The challenge here is: because the maximum spoofing range in a cycle is limited by a bad data detection threshold (see Section II-B) and the physical limitation of its GPS receiver, we have to carefully determine the spoofing signals within proper ranges in order to shift the drone position as much as we can, without triggering GPS-failure alarms. Obviously, we cannot arbitrarily redirect a drone to any destination due to constraints such as the maximum redirection distance per cycle and the attack duration. We will present the detailed attack steps and analyze the feasible range of the attack under given constraints in Section III.

B. Drone Control Background

Here we introduce the related drone control background and discuss how we will address the above research problems. Without loss of generality, to simplify our model, we assume that an invading drone is on autopilot, because of two reasons: First, because the drone operator usually does not want to expose itself, it has to turn off the control channel to avoid being triangulated based on its control signals. Second, because the control channel can be easily disrupted by the defense, the operator cannot depend on the channel to control the drone in the restricted area. So, autopilot is a natural choice.

As shown in Figure 2, autopilot is often achieved in four steps as many feedback-control systems. Starting with sensor measurements, the system estimates related states and then passes the states to its navigation algorithms to determine how

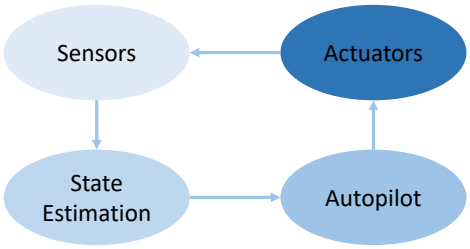

Fig. 2. Common Drone Control Loop.

to adjust actuators for real-time control. Such a loop is usually completed in a fixed period, e.g., the default state update on ArduPilot is set to every 10 ms when IMU sensors generate new readings. We introduce the most popular state estimation algorithms and the most common navigation algorithm in the following.

Common State Estimation Algorithms. A consumer drone conducts state estimation based on sensor readings, e.g., from accelerometers, gyroscopes, magnetometers, GPS signals, and barometers. Extended Kalman Filters (EKF) and its variants are the most commonly-used state estimation methods in current systems [13], [14], providing fairly accurate state estimations. A few enhanced methods are proposed for critical altitude estimations based on other sensors (e.g., ultrasonic sensors). However, because they usually need more resources that are not available on low-end consumer drones, we focus on the common EKF-based state estimation in this paper. The EKF on ArduPilot 3.6 estimates 24 states for drone control [15]; we focus on the position and velocity estimation because our goal is to manipulate drone positions to affect flight paths. Readers interested in the details of these EKF algorithms can refer to our previous paper [16].

We adopt several common assumptions in this paper. First, we assume that we are able to determine the parameters used in the control system, e.g., the bad-data detection threshold τ (see below). Many parameters on a drone are usually configured to default values based on experiences and its physical properties (e.g., weight or acceleration limit), which can be easily learned by examining the open-source code or reverse-engineering the firmware of the same model. In addition, we consider that the proposed attack is performed when the system is in *a steady state*, such that we can draw concrete conclusions to illustrate the attack effects. This is a common assumption in examining EKF-based control systems, and usually achieved in real systems [13], [17], [18].

Bad Data Detection. The EKF on ArduPilot uses a common anomaly detection algorithm to determine if a sensor measurement is acceptable, e.g., for position measurements, by checking if the following condition is true:

$$inn_N{}^2 + inn_E{}^2 \leq (var_N^{inn} + var_E^{inn}) \cdot \tau, \qquad (1)$$

where τ is a pre-set threshold, inn_N and inn_E are the innovation between a prediction and a measurement of a position state in the North and East directions, var_N^{inn} and var_E^{inn} are the variances of inn_N and inn_E, respectively. The values of var_N^{inn} and var_E^{inn} are calculated based on the covariance matrix of the EKF and the GPS position accuracy information; they can be regarded as constants in a steady-state system, based on existing results [1], [13].

Navigation Adjustment. The navigation of ArduPilot uses a *linear-track-based algorithm*, which is the most popular path-following algorithm on drones, more accurate than others [19]. Because a drone may drift away slightly from its scheduled flight track, due to various factors (e.g., wind disturbances), it usually runs a path-following algorithm to trace the track, as shown in Figure 3. Here, as the drone's track is from the bottom left to the upper right, the algorithm should keep the drone close to the track, i.e., its position states (estimated via EKF) should be close to the track. In order to achieve this, the navigation frequently adjusts the drone's movements to make it close to the track.

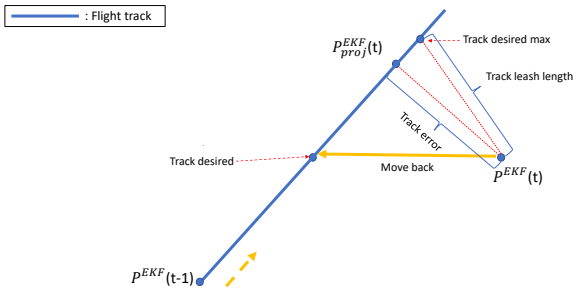

Fig. 3. Adjustment in path-following.

In each time interval, the navigation calculates a position called *track_desired* for the next interval based on its current position estimation and its scheduled velocity. Assume its position estimation is $P^{EKF}(t-1)$ in the interval $(t-1)$. In interval t, the navigation finds itself at the position $P^{EKF}(t)$ away from the track (based on GPS), and it then performs the following adjustment. First, the distance between $P^{EKF}(t)$ and its projection on the track $P^{EKF}_{proj}(t)$ is defined as a *track_error*. Next, the algorithm determines a *track_leash_length* based on its velocity, acceleration, and current position, and uses it to choose how the drone should fly back to the original track as follows. Specifically, the algorithm first identifies a position on the track called *track_desired_max* (which is the farthest distance along the track that the leash will allow), and then compares it with *track_desired* position to decide which position the drone should fly back to. In particular, if *track_leash_length* $\leq$ *track_error*, then *track_desired_max* is the projection of the current position $P^{EKF}(t)$ on the track; otherwise, *track_desired_max* is the position on the track that has the distance of *track_leash_length* from the current position $P^{EKF}(t)$, as shown in the figure. Furthermore, if *track_desired_max* is closer to the destination than *track_desired*, as in this example, the drone will fly to *track_desired*; otherwise, the drone will fly to *track_desired_max*. Our attack exploits this adjustment to achieve our drone position manipulation.

C. Attack Methods: Smart GPS Spoofing

After an invading drone enters the restricted zone, we use GPS spoofing to compromise its GPS position and velocity readings, to carefully feed crafted inputs to the drone state estimation algorithms in order to mislead its navigation control. In this paper, we will manipulate the drone position in a 2D plane of longitude and latitude. In ArduPilot simulations, we pass the GPS inputs with MAVLink *GPS_INPUT* messages to emulate a covert spoofing, such that we can test our attacks on the navigation control, which are the focus of this paper.

In addition, we also verified the feasibility of practical GPS spoofing on real drones. While we assume we can use existing tools to achieve covert GPS spoofing, we have conducted GPS spoofing on a SkyViper GPS drone for testing, whose firmware is a variant of ArduPilot. We overpowered the civilian GPS signals using a BladeRF A9 card to transmit shifted GPS signals from a moderate distance [20]. The drone gained a lock on the spoofed GPS signals after a short delay. We verified this by reading the GPS raw inputs from the drone via its MAVLink interface.

III. Proposed DPM Attack

In this section, we will introduce the *Drone Position Manipulation (DPM)* attack that directly uses the estimated position states of a drone to craft spoofed GPS inputs. Although obtaining EKF states is impractical on a real system, this method helps us better understand the proposed attack on a complicated control system and build a baseline analysis model of the attack. We will further develop a practical solution for obtaining drone positions in our following work. In the following, we will present the DPM attack and then illustrate its capability by formally analyzing its maximum feasible redirection range for a given original destination.

A. Theoretical Foundation of DPM

Let us first present the theoretical foundation of DPM, consisting of three important propositions.

Notations. We first define related notations for our discussion. Because we focus on the drone position in a horizontal 2D plane, we consider the drone's *real velocity* as a 2D vector $V^r = (V^r_N, V^r_E)$, with a sub-component to the North, V^r_N meter/second (m/s), and a sub-component to the East, V^r_E m/s. For example, when a drone moves at 4 m/s to the Northeast, we observe a velocity vector V^r as (2.81, 2.81) m/s. In DPM, for each GPS cycle, we build the spoofed GPS position inputs by first obtaining the estimated position state and then adding a shift to it with an injection vector of $I = (I_N, I_E)$ m/s, which has a sub-component to the North I_N, and a sub-component to the East I_E. Now when applying $I = (0, 10)$ m/s (or $(0, 1)$ m/GPS cycle) to a drone flying to the Northeast with V^r = (2.81, 2.81) m/s, i.e., injecting 0 m/s to the GPS position in the North direction and 10 m/s (or 1 m/GPS cycle) to the GPS position in the East, we observe that the drone's real velocity V^r quickly entered a stable velocity of (2.81, 2.37) m/s. In other words, the drone drifts away at a stable velocity V^r_{drift} at (0, -0.44) m/s, i.e., the drone drifts to the West at 0.44 m/s as the result of the injections. With these notations, we introduce the following propositions:

Proposition 1. Drift velocity V^r_{drift} is proportional to injection rate I under a DPM attack with an attack coefficient C^a defined as follows:

$$C^a = (C^a_N, C^a_E) = \left(\frac{V^r_{drift,N}}{I_N}, \frac{V^r_{drift,E}}{I_E}\right) \tag{2}$$

for $I_N \neq 0$ and $I_E \neq 0$. If I_N (or I_E) == 0, C^a_N (or C^a_E) = 0.

Proposition 2. For attacks on the same drone in the same environment, C^a keeps unchanged for any proper injection size in any direction.

In this DPM attack, we choose the injection rate I as a fixed value in each GPS cycle, which results in nearly fixed innovations in each EKF position estimation cycle, denoted as Δ (because $I \approx \Delta$). Since the Kalman gain K_k usually quickly becomes a constant in a steady state, the deviation of position estimation $V^{EKF}_{drift} = K_k \cdot \Delta$ becomes constant. Because the deviation will be corrected in each cycle, $V^r_{drift} = -V^{EKF}_{drift}$ becomes a constant in the cycle as well. In addition, $C^a \approx -K_k$, which can be regarded as the same constant for attacks on the same drone in the same environment.

We have further validated these propositions with ArduPilot SITL simulations. Figure 4 shows the relationship between the injection rate and the drift velocity observed during DPM attacks on SITL. With the injection rate (x-axis) increasing from 4 m/s to 40 m/s, we observe that the drift velocity (y-axis) increases in the opposite direction proportionally to the injection rate with coefficient C^a. Furthermore, we have validated that Propositions 1 and 2 hold for injections in any direction in the 2D plane, when we apply a feasible constant injection rate. We measured that $C^a_E \approx -0.0455$ and $C^a_N \approx -0.0491$ in these simulations.

Proposition 3. When we apply a 2D injection rate $I = (I_N, I_E)$, the effect is equivalent to the combined effects of attacking only in the North direction with $I_1 = (I_N, 0)$ and attacking only the East direction with $I_2 = (0, I_E)$.

Proposition 3 (Decomposition Proposition) holds because

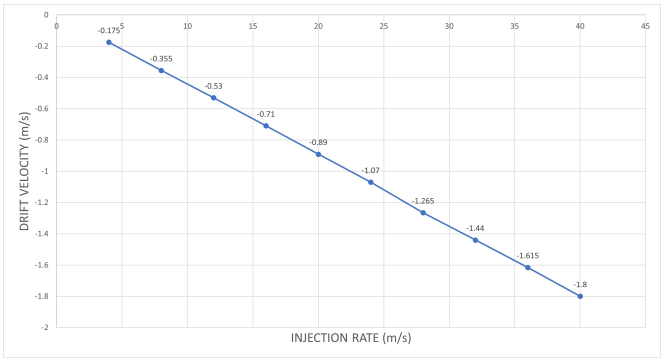

Fig. 4. Relationship between the injection rate x-axis and the drift velocity y-axis. As the injection rate increases, we can see the drift velocity increases in the opposite direction proportionally, and the attack coefficient C^a stays roughly constant.

the drone state estimation algorithms usually decompose the 3D positions and velocities into North, East, and Down subcomponents [15]. Based on Propositions 2 and 3, we can simplify the analysis of the attack result under an injection rate I in any direction in the 2D plane, by decomposing the injection rate $I = (I_N, I_E)$ into $I_1 = (I_N, 0)$ and $I_2 = (0, I_E)$. Using measured attack coefficients C_E^a and C_N^a, we can find the drift velocities of the drone in the North and East directions: $V^r_{drift,N} = I_N \cdot C_N^a$ and $V^r_{drift,E} = I_E \cdot C_E^a$; then we can find the attack result with the injection rate I by combining the drift velocities in the two directions.

B. DPM Attack

We illustrate how the DPM attack redirects a drone to a specific destination in a 2D plane, as shown in Figure 5. Given the redirected destination $R = (R_N, R_E)$, where we would like to guide the drone to reach, we define a redirection vector $\mathbf{DR} = (DR_N, DR_E)$ as $((R_N, R_E) - (D_N, D_E))$, the vector difference between the redirected destination (R_N, R_E) and the original destination (D_N, D_E). Assume we perform a DPM attack on a drone's position for n cycles to achieve **DR**; we evenly distribute the required injection in each cycle, i.e., in cycle t, we apply an injection $I(t) = (\frac{DR_N}{n \cdot C_N^a}, \frac{DR_E}{n \cdot C_E^a})$ on the current position state $P(t)$ to build its position input $P'(t)$, $0 \le t < n$. As a result, the navigation algorithm observes that the drone has drifted away from the track, and it will make an adjustment to move it back to the track. After the adjustment, the system considers the drone has returned to the track at $P(t+1)$, but its real position is actually at $P^r(t+1)$. In this example, we only need 5 injection cycles to achieve the required redirection; after 5 cycles, we stop injections and the drone will fly towards the redirected destination in a path parallel to the original track.

Similar to the above illustration, we introduce the main steps of DPM in Algorithm 1. Given the flight track of a drone, a redirected destination, and drone position states, the attack builds a position injection in each cycle in order to lead the drone away from its original track to achieve the redirection, as explained in the above. The maximum injection rate $I^{max} = (I_N^{max}, I_E^{max})$ per cycle can be determined based on the parameters associated with the bad-data detector in theory [16] and it also can be measured in practice. Attack coefficient $C^a = (C_N^a, C_E^a)$ is defined in Proposition 1, and can be measured in advance. A video demonstration of a DPM attack on ArduPilot SITL is at Youtube, https://youtu.be/kE0T4sFJZ7o, and we will evaluate the accuracy and feasible redirection range of DPM in Section IV.

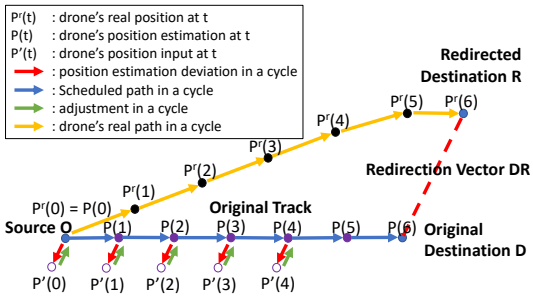

Fig. 5. Illustration for the DPM attack.

Algorithm 1: DPM Attack Algorithm.

input: Original track from (O_N, O_E) to (D_N, D_E);
Redirected destination (R_N, R_E);
Drone position state estimation $P^{EKF}(t)$.

Initialization: $\{I(t)\} \longleftarrow \emptyset$, $t \longleftarrow 0$;
n_0 = the remaining number of cycles on the original track;
DR $= (DR_N, DR_E) \longleftarrow (R_N - D_N, R_E - D_E)$;
$n \longleftarrow max(\lceil \frac{DR_N}{I_N^{max} \cdot C_N^a} \rceil, \lceil \frac{DR_E}{I_E^{max} \cdot C_E^a} \rceil)$;
find the total no. of injection cycles;
while $t \le n_0$ **do**
  if $t < n$ **then**
    $I(t) \longleftarrow (\frac{DR_N}{n \cdot C_N^a}, \frac{DR_E}{n \cdot C_E^a})$; injecting until $t \ge n$;
    $P^{GPS}(t) = I(t) + P^{EKF}(t)$; build fake position inputs;
  else
    $P^{GPS}(t) = P^{EKF}(t)$; fly towards R, not add injection;
  send $P^{GPS}(t)$ as GPS position inputs;
  $t \longleftarrow t + 1$;

C. Feasible Range of Redirected Destination

Obviously, the above attack cannot redirect a drone to an arbitrary destination due to many factors such as the attack duration and the maximum redirection per cycle. Therefore, we need to further figure out if a given redirected destination is feasible. In the following, we will analyze such a feasible range of the redirected destination to show the overall capability of DPM, which can help us determine if a redirected destination is reachable or not.

Eq. 1 is commonly used for EKF-based bad data detection. Since $(var_N^{inn} + var_E^{inn}) \cdot \tau$ can be regarded as a constant λ in a steady state, and inn_N and inn_E are roughly equal to I_N and I_E. Plugging them into Eq. 1, we have

$$I_N^2 + I_E^2 = \lambda. \tag{3}$$

Then, based on Eq. 2 on the previous page, we have

$$(\frac{V^r_{drift,N}}{C_N^a})^2 + (\frac{V^r_{drift,E}}{C_E^a})^2 = \lambda, \tag{4}$$

or

$$(V^r_{drift,N})^2 + (\frac{V^r_{drift,E}}{C_E^a / C_N^a})^2 = \lambda \cdot (C_N^a)^2. \tag{5}$$

Eq. 5 shows the feasible range of the redirected destination in one attack cycle is an ellipse with its center at the original destination and eccentricity $\sqrt{1 - \frac{(C_E^a)^2}{(C_N^a)^2}}$ (close to 0 in practice). After n attack cycles, the feasible range of the redirected destination will be

$$(\frac{R_x - D_x}{C_N^a})^2 + (\frac{R_y - D_y}{C_E^a})^2 = n^2 \cdot \lambda, \tag{6}$$

We will show concrete feasible ranges in Section IV.

IV. Performance Evaluation

A. System Instrumentation

We have conducted extensive analysis and testing of ArduPilot Copter code to understand the state estimation and navigation control algorithms. Our evaluation is performed on the SITL module of ArduPilot. For system control, the SITL module runs the same code as a real firmware to simulate a flight with a large set of common parameters. As a SITL drone runs in the same way as a real drone, it can take different types of GPS input formats, e.g., popular UBLOX and NMEA formats. In its default setting, it simply uses the simulated (physical) position of a drone as its GPS position input. So, we can perform covert GPS spoofing by switching its GPS input to take MAVLink GPS_INPUT messages, in the same way as it receives GPS messages from a GPS-capable device.

We uploaded a video of a covert DPM attack on the above platform at Youtube (https://youtu.be/kE0T4sFJZ7o). With the source location as (0, 0), the drone's destination is set to the waypoint of (500 meters, 500 meters) in the Northeast with a velocity of 4 meter/second. Waiting for 20 seconds into the mission after the system entered a steady state, we started to covertly spoof GPS inputs with an injection of 4.07 meter/per GPS cycle to the North. (It takes less than 20 seconds for the drone to enter a steady state.) As shown in Figure 6.(b), we then saw that the drone state (shown as the upper drone icon) is still on its original track (in pink) to the Northeast; but its real position (shown as the lower drone icon) is shifted down to the South, below its original track. The drone continued with its mission, without noticing its real position is gradually away from its original track. When the drone position state is close to the original destination, the real drone position is about 100 meters South to the original destination, as shown in Figure 6.(c).

With significant efforts in the past years, we were able to build this in-depth instrumentation platform for examining the control algorithms and the proposed attacks in details, which also facilitated the evaluation presented in the following.

B. Evaluation of DPM

1) Simulation Settings: In each attack simulation, to show the pure attack effect, we did not launch the attack until the system entered a steady state, i.e., we waited for 20 seconds after it reached the takeoff altitude and began to fly to a preset waypoint. We used the common settings of consumer drones as key parameters in the evaluation, e.g., a GPS update cycle is set to 0.1 second; the horizontal position accuracy of GPS input is 0.1 meter; the velocity accuracy of GPS input is 0.1 meter/second; a default drone starting velocity is 4 meter/second. We have repeated the simulations many times to observe and measure the coefficients for our model presented in Section III: for example, we used linear regression to find the attack coefficients $C_a^N = -0.0491$, and $C_a^E = -0.0455$.

2) Accuracy of DPM: As our goal is to divert a drone to a redirected destination, we first evaluated the accuracy of DPM, i.e., the difference between the expected redirected destination and the actual destination. For easy illustration, we first set the injection direction to the East and the total attack duration to 50 seconds. Consider the home position as the original point (0, 0), the original destination was set to (500 meters, 500 meters) in the Northeast in a local frame. To evaluate the accuracy under different attack sizes, we varied the size of **DR** (the vector difference between the redirected destination (R_x, R_y) and the original destination (D_x, D_y)) from 20 to 100 meters. Table I shows the attack error rates under different injection rates. The 1st row shows the size of intended redirection vector from 20 to 100 meters. The 2nd row is the corresponding injection size derived based on the size of redirection vector using Algorithm I. The 3rd row is the size of the actual redirection vector obtained from the simulation. In the 4th row, we can see that: for different redirection sizes, the DPM attack achieved very small errors (under 1.5%), i.e., it can accurately redirect a drone to the intended destination.

TABLE I
DPM Attack error under different Injection rates.

expected **DR** size (m)	20	40	60	80	100
Injection (m/GPS-cycle)	0.88	1.76	2.64	3.52	4.40
Actual **DR** size (m)	20.22	39.81	60.01	81.14	100.36
Error Rate	**1.10%**	**-0.475%**	**0.017%**	**1.425%**	**0.36%**

Next, keeping the same source and destination as the above, we evaluated the DPM's accuracy in eight directions: North, Northeast, East, Southeast, South, Southwest, West, Northwest. The total attack duration is set to 50 seconds as the above, and the redirection vector is set to 100 meters. In Table II and Table III, the 1st row shows the injection direction; the 2nd and the 3rd row show the injection sub-components in the North and the East directions; the 4th and the 5th row show the errors in the North and the East directions; the 6th and 7th row show the error rates in the North and the East directions. We can see the attack errors for these cases are still very small (under 0.9% in a sub-component), which shows the DPM attack can accurately redirect the drone to different directions.

TABLE II
DPM Attack error under different attack directions (1).

Injection direction	E	W	N	S
Injection: North (m/GPS-cycle)	0	0	5	-5
Injection: East (m/GPS-cycle)	5	-5	0	0
Error: North (m)	/	/	-0.23	-0.29
Error: East (m)	0.06	-0.13	/	/
Error Rate: North	/	/	**-0.19%**	**-0.24%**
Error Rate: East	**0.053%**	**-0.11%**	/	/

TABLE III
DPM Attack error under different attack directions (2).

Injection direction	NE	NW	SE	SW
Injection: North (m/GPS-cycle)	5	5	-5	-5
Injection: East (m/GPS-cycle)	5	-5	5	-5
Error: North (m)	0.16	0.02	-1.1	-0.59
Error: East (m)	0.62	0.14	0.23	-0.05
Error Rate: North	**0.13%**	**0.016%**	**-0.90%**	**-0.48%**
Error Rate: East	**0.55%**	**0.12%**	**0.20%**	**-0.044%**

(a) Before attack. (b) Under attack. (c) Reached Destination.

Fig. 6. DPM Demo in ArduPilot SITL.

3) Injection limitation and Feasible range: Although we have shown that the DPM can redirect a drone to any direction with high accuracy, the maximum size of the redirection is limited by the maximum injection allowed in each cycle that is limited by the bad data detector of the drone. To find the maximum injection rate allowed in a direction, we gradually increased the injection rate until the system detects the large error term and raises GPS-fail alarms. We repeated these simulations to confirm the maximum injection rate for DPM in each direction, which will then give us the largest redirection size **DR** for a given attack duration in the direction. Then we can combine the maximum redirection size in all directions to outline the feasible range of the DPM attack, i.e., we can redirect the drone to any point within this range under this attack duration. In Table IV and Table V, the attack duration was 50 seconds, and we tested in 8 directions to outline the feasible redirection range for attacking 50 seconds. The 1st row shows the redirection directions; the 2nd row shows the maximum injection rate in a direction; the 3rd row shows the maximum size of redirection. The maximum injection rate for each direction varies from 7.09 to 7.65 meter/GPS cycle; the maximum redirection size in each direction varies from 169.28 meters to 177.03 meters for the attack duration of 50 seconds.

Furthermore, to show the feasible ranges of redirected destinations under different attack duration, we varied the attack duration from 20 to 100 seconds, and determined the corresponding maximum redirection sizes in 8 directions. We then outlined the feasible ranges under different attack durations in Figure 7. In this 2D plane, the center location (0, 0) is the original destination; the smallest circle-like range is the feasible range under an attack duration of 20 seconds; the largest circle-like range is the feasible range under an attack duration of 100 seconds. It is easy to see that the attack feasible range expands as the attack duration increases.

V. Related Work

As the flow chart of an autopilot system shown in Figure 2, attacking a drone may happen at three levels:

TABLE IV
DPM Maximum Redirection Size (1).

Injection Direction	E	W	N	S
Max Injection (m/GPS-cycle)	7.56	7.6.	7.09	7.14
Max **DR** size (m)	171.58	173.32	171.32	173.00

TABLE V
DPM Maximum Redirection Size (2).

Injection Direction	NE	NW	SE	SW
Max Injection (m/GPS-cycle)	7.24	7.48	7.50	7.37
Max **DR** size (m)	169.28	176.77	177.03	173.28

(1) **Sensor-level attacks.** Different from mission-critical systems, consumer drones are usually equipped with low-end sensors with limited protection to reduce costs, which leaves many opportunities for hardware or software attacks.

Hardware attacks include selectively jamming GPS and radio control channels [21]–[23], or compromising drone hardware components (such as MEMS sensors) via acoustic approaches to disturb its normal operation [1], [2], [24], [25]. In particular, a high-accuracy covert spoofer was built by manipulating the signal delays in the physical layer to spoof GPS signals arriving at a drone GPS receiver [1]. The spoofer measures relevant delays to the receiver within a few nanoseconds, and compensates for these delays by generating a slightly advanced version of the official GPS signals that the spoofer receives. Then, it gradually increases power to win the signal acquisition on the receiver over the official signals. This covert GPS spoofer is a pioneer work that can also help us implement our attack to manipulate GPS signals at the physical layer.

(2) **Attacks on State Estimation.** Compromising system states is a common method to cause serious errors in control systems. The proposed drone position manipulation utilizes a type of *False Data Injection (FDI) attack* to exploit the small tolerance ranges of common *bad-data detection schemes* in order to compromise drone state estimation to achieve accurate position manipulation. Common FDI attacks aim to manipulate state estimations via modifying corresponding measurements

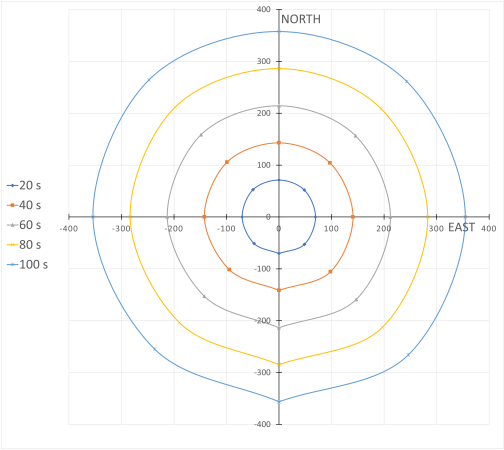

Fig. 7. Feasible ranges of redirected destinations under different attack durations in DPM.

without being detected by bad data detectors [26]. Based on existing research on state estimation algorithms [3], in this paper, we are able to determine the GPS spoofing signals that can pass the bad data detection and also manipulate the system states based on our needs to make the navigation to adjust drone positions.

(3) **Attack Drone Navigation Controls.** By analyzing the practical navigation code of ArduPilot, we identified weaknesses in the most popular path-following algorithms [19] as introduced in Section II-B. In this paper, we utilize the vulnerability in the state estimation and exploit the weakness of navigation to accurately manipulate a drone's position in order to guide it to a redirected destination. Another project [5] focused their attack on the navigation control on an earlier version ArduPilot 3.3. They provided a taxonomy of GPS fail-safe mechanisms on consumer drones, and investigated the attack strategy for each mechanism. In contrast, we develop complete algorithms on how to guide an invading consumer drone to a redirected destination, and we also determined the feasible range of the redirected destination relative to an original destination.

VI. Conclusions

In this paper, we have examined the entire control stack in the control loop of consumer drones, and identified the vulnerabilities at each level of the stack; we then developed the DPM attack. Our analysis and evaluation have shown that DPM can accurately guide an invading drone to a redirected location. We have also analyzed the maximum feasible range of the redirected destination for a given original destination. We believe this is the first work that is able to guide a consumer drone accurately to a feasible destination.

This work is supported by NSF-1662487 and ONR No. N000142012049 and No. N000142112168. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF or ONR.

References

[1] 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.

[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," in *2017 IEEE European symposium on security and privacy (EuroS&P)*. IEEE, 2017, pp. 3–18.

[3] W. Chen, Y. Dong, and Z. Duan, "Manipulating drone dynamic state estimation to compromise navigation," in *2018 IEEE Conference on Communications and Network Security (CNS)*, May 2018, pp. 1–9.

[4] P. Dash, M. Karimibiuki, and K. Pattabiraman, "Out of control: stealthy attacks against robotic vehicles protected by control-based techniques," in *ACSAC '19: Proceedings of the 35th Annual Computer Security Applications Conference*, Dec. 2019.

[5] J. Noh, Y. Kwon, Y. Son, H. Shin, D. Kim, J. Choi, and Y. Kim, "Tractor beam: Safe-hijacking of consumer drones with adaptive gps spoofing," *ACM Transactions on Privacy and Security*, vol. 22, no. 2, p. 12, 2019.

[6] E. Deligne, "Ardrone corruption," *Journal of Computer Virology, vol.8, pp.15-27*, 2012.

[7] A. M. Shull, "Analysis of cyberattacks on unmmaned aerial systems," Master's thesis, Purdue University, 2013.

[8] M. Monnik, "Hacking the parrot ar.drone 2.0," https://dronesec.com/blogs/articles/hacking-the-parrot-ar-drone-2-0, 2019.

[9] 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, Sep. 2014.

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

[11] ArduPilot, "Ardupilot autopilot suite," http://ardupilot.org/ardupilot/.

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

[13] G. Welch and G. Bishop, "An introduction to the kalman filter," *University of North Carolina at Chapel Hill, Chapel Hill, NC*, 1995.

[14] P. Gasior *et al.*, "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.

[15] ArduPilot, "Extended kalman filter navigation overview and tuning," http://ardupilot.org/dev/docs/extended-kalman-filter.html, 2020.

[16] W. Chen, Y. Dong, and Z. Duan, "Manipulating drone position control," in *in Proc. of IEEE Conference on Communications and Network Security(CNS)*, June 2019, pp. 1–9.

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

[18] E. Ghahremani and I. Kamwa, "Dynamic state estimation in power system by applying the extended kalman filter with unknown inputs to phasor measurements," *IEEE Transactions on Power Systems*, vol. 26, no. 4, pp. 2556–2566, 2011.

[19] P. Sujit *et al.*, "Unmanned aerial vehicle path following: A survey and analysis of algorithms for fixed-wing unmanned aerial vehicless," *IEEE Control Systems*, vol. 34, no. 1, pp. 42–59, Feb. 2014.

[20] J. Cao, "Practical gps spoofing attacks on consumer drones," Master's thesis, University of Hawaii, https://drive.google.com/file/d/1QEpPT8JzEBVmptSi_Y92Sx7MUVRRTvy0/view?usp=sharing, 12 2020.

[21] I. SRC, "Silent archer counter-uas system," http://www.srcinc.com/what-we-do/ew/silent-archer-counter-uas.html, 2016.

[22] B. S. Systems, "Auds anti-uav defence system," http://www.blighter.com/products/auds-anti-uav-defence-system.html, 2016.

[23] Battelle, "Drone defender," https://www.battelle.org/government-offerings/national-security/tactical-systems-vehicles/tactical-equipment/counter-UAS-technologies, 2016.

[24] N. O. Tippenhauer *et al.*, "On the requirements for successful gps spoofing attacks," in *Proc. of 18th ACM conf. on Comp. and comm. security*, 2011, pp. 75–86.

[25] Y. Son *et al.*, "Rocking drones with intentional sound noise on gyroscopic sensors," in *24th USENIX Security Symposium (USENIX Security 15)*, Washington, D.C., Aug. 2015, pp. 881–896.

[26] 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, 2009, pp. 21–32.