DPM: Towards Accurate Drone Position Manipulation

Wenxin Chen, Yingfei Dong, *Senior Member, IEEE,* and Zhenhai Duan, *Senior Member, IEEE*

Abstract—While existing methods can disrupt drone invasions in a protected airspace, none of them is able to accurately guide a drone to a desired location for safe handling. To address this issue, the *Drone Position Manipulation (DPM)* attack is proposed in this paper to utilize weaknesses in common navigation algorithms of consumer drones. The main advantage of DPM is that it can accurately redirect an invading drone to a desired location, by carefully crafting the spoofed guidance inputs of the drone to exploit the adjustment in the path-following navigation algorithms. Compared with existing methods, this is the first work to achieve such accurate quantitative control. Another main contribution of this work is that it explores three fundamental components (i.e., guidance sensing, state estimation, and navigation control) together to enable the quantitative manipulation of flight paths, different from all existing methods. Furthermore, a formal analysis of the attack range is presented for investigating where a drone can be redirected from its target under given constraints. The evaluation on the Software-in-the-loop (SITL) module of *ArduPilot* system shows that the proposed attack is able to not only accurately lead a drone to a redirected destination but also make it fairly far away from its target. Lastly, the proposed attack can be applied to many autopiloted systems, because it exploits common weaknesses in these systems.

Index Terms—autonomous vehicle, drone security, navigation algorithms, UAS

1 Introduction

As many emerging autonomous robotic vehicles, consumer drones have enabled numerous new applications [1], such as search-and-rescue, aerial imaging, and package delivery. However, they have also been abused in many incidents [2]: a drone crashed into the White House ground [3]; malicious drones disturbed 1000+ flights with 140,000+ passengers around Gatwick airport for three days; drones were deployed in an assassination attempt on Iraq prime minister on Nov. 7, 2021. Therefore, it is urgent to develop effective drone countermeasures. This paper focuses on "consumer drones" because of their low costs and broad availability, and this paper does not consider expensive high-end mission-critical drones because they have more on-board resources and different requirements. In the following, "drones" mean consumer drones.

Existing drone countermeasures are mostly developed by industry using direct physical methods, such as jamming a drone's radio control channels to trigger it into a fail-safe mode to land, shooting it down with a projectile, or capturing it with a net. While such brute-force physical methods work well in many cases, they have serious limitations, e.g., they usually do not handle collateral damages well. If a drone carries a malicious payload, it is certainly undesirable to land it in a protected area. The best solution for this case is to guide the drone away to a designated area for safe handling. While several projects [4], [5], [6], [7], [8], [9], [10], [11] have demonstrated the feasibility of such attacks, none of them is able to provide quantitative control on practical systems. Different from all these existing methods, this paper focuses on accurate quantitative control by systematically investigating three fundamental components of the control stack together for a specific goal, i.e., redirecting a drone to a given destination.

An ideal attack is to gain the complete control of an invading drone, e.g., if attackers can take over its control channel (by cracking its encryption) or hack into its control software. While several methods have been developed to exploit specific drone settings, they require directly compromising drone software, sensors, or communication channels [8], [12], [13], [14], [15], [16], which are often difficult to achieve in practice. While these methods may work well on weak systems with known vulnerabilities, it is impractical to solely rely on such methods because the vulnerabilities can be easily patched.

Therefore, different from these methods, this paper focuses on a new challenge: *how to accurately manipulate a drone without depending on directly compromising its software or hardware.* To achieve this goal, the paper proposes to exploit the three fundamental components of autopiloting (i.e., guidance sensing, state estimation, and navigation control) together, as presented in the following. First, almost all consumer drones depend on guidance inputs such as civil GPS that can be easily spoofed. We have utilized existing software-defined radio (SDR) tools to achieve this on real drones [17]. Furthermore, assume existing methods [18], [19], [20] can help identify the model of an invading drone, we can further identify its state estimation and navigation algorithms based on previous analysis. As these common algorithms are designed mainly for better control without considering security concerns, they are vulnerable to attacks as demonstrated in this paper. After carefully analyzing them, the guidance inputs of a drone are identified as the attack surface in the proposed solution. This paper then focuses on carefully constructing spoofed GPS signals to exploit both the limitations of state estimation and navigation algorithms for manipulating a drone's position. Such a holistic solution integrates the vulnerabilities at three levels together and achieves accurate quantitative position control.

In summary, the main contributions of this paper are as

- *W. Chen and Y. Dong are with the Department of Electrical and Computer Engineering, University of Hawaii, Honolulu, HI 96822 USA. e-mail: wenxinc@hawaii.edu, and yingfei@hawaii.edu.*
- *Z. Duan is with Department of Computer Science, Florida State University, Tallahassee, FL 32306. e-mail: duan@cs.fsu.edu.*

follows: (1) The DPM attack with a theoretical model is developed to achieve accurate manipulation of a drone's position, while existing methods are only able to disrupt a drone's mission but did not provide a clear model and cannot achieve quantitative control. (2) Three fundamental components in the control loop are explored together, while existing methods mostly focus on one or two. (3) The proposed attack is validated on ArduPilot, arguably the most popular open-source flight control system (compared to Paparazzi [21] and OpenPilot/LibrePilot [22]), to show its effectiveness in practical settings; while existing methods are mostly evaluated numerically, not on real systems. (4) The proposed attack does not require to directly compromise the software or hardware of a drone as many existing methods requires [8], [15], [16].

While the initial idea was discussed in [11], now a complete framework with in-depth analysis and detailed evaluation on practical settings are developed in this paper. To validate the proposed ideas, the proposed attacks are evaluated on the Software-in-the-loop (SITL) module of a stable release of ArduCopter [23], which runs the same code as a firmware on a real drone. The results show that the proposed attack is able to accurately lead a drone to a redirected destination. Furthermore, the feasible range of such redirected destination is analyzed to show where the proposed attack can redirect a drone away from its target.

The remainder of this paper is organized as follows. Section 2 introduces the problem statement, common drone control algorithms, and the attack methods. Then the proposed *basic DPM (bDPM)* attack and the practical *measurement-based DPM (mDPM)* attack are presented in Section 3 and Section 4, respectively. The system instrumentation and performance evaluation are discussed in Section 5. Related work is presented in Section 6, and Section 7 concludes this paper and also points out future research directions.

2 Problem Statement and Background

2.1 Problem Statement

As shown in Figure 1, a restricted area is set up around a critical asset to protect it from drone invasions. When an invading drone flies towards the critical asset, a preferred defense scheme is to lead it away from the asset to a desired location for safe handling, e.g., a blast containment chamber. Assume existing approaches (via radar, radio or traffic profiling, or image processing [18], [19], [20]) help the defense recognize the invading drone; as a result, its sensors (e.g., GPS receivers) and firmware (including its state estimation and navigation algorithms) can also be identified. Because the defense can obtain the same model of drone and analyze it ahead of time, it is able to utilize the weaknesses of the drone's GPS receiver, state estimation, and navigation algorithms to manipulate the drone's position states and flight paths towards a desired location, by carefully spoofing the GPS signals. As a result, the defense does not require to compromise the drone's software or hardware, as some existing methods requires.

To address this challenge, in this paper we carefully explore the fundamental components of the control loop (i.e., guidance sensing, state estimation, and navigation control) together to achieve quantitative control. In particular, the following problems need to be solved: (1) First, the defense needs to make a drone locking on the spoofed GPS signals. We rely on existing methods to achieve this, e.g., using the covert attack proposed in [4]. Although this is not the focus of this paper, we have successfully made a drone's GPS receiver locking on the spoofed signals transmitted with a BladeRF SDR card [17]. In addition, on the ArduPilot SITL platform, we have also discovered different methods to spoof GPS signals in simulation, in order to understand how to spoof GPS signals without triggering detection schemes. (2) Second, we need to determine how to construct the spoofed GPS position inputs, e.g., based on the drone's original flight path and a redirected destination in a given attack duration (or dynamically adjusting the injections based on measurements). *This is the focus of this paper* as presented in the following.

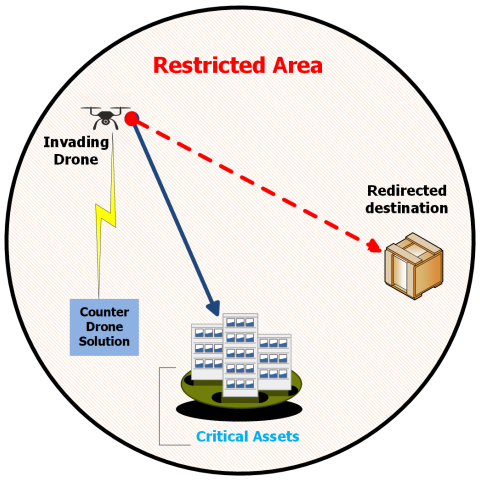

Fig. 1. Restricted Area around a critical asset.

The process of constructing spoofed GPS signals can be further divided into two steps: First, the shifted distance of the drone's position in each GPS cycle is determined in order to make the navigation algorithm to adjust its flight path towards the redirected destination. Second, the spoofed GPS inputs are carefully constructed based on the shifted distance in each GPS cycle (normally 0.1 second). The challenge is that, because the maximum spoofing distance in a cycle is limited by a bad data detection threshold (see Section 2.2) and the physical limitation of a GPS receiver, the spoofing inputs have to be carefully constructed within proper ranges in order to shift the drone position inputs as much as we can, while not triggering GPS-failure alarms. Furthermore, a drone cannot be arbitrarily redirected to any destination due to various constraints, such as the maximum redirection distance per cycle and the attack duration. In Sections 3 and 4, the detailed attack procedures and their feasible ranges are presented.

2.2 Drone Control Background

The related drone control background and the basic ideas of addressing the above research problems are introduced in this subsection.

Without loss of generality, assume that an invading drone is on autopilot to simplify the proposed model. The autopilot is often achieved in four steps as in many feedback-control systems. Starting with *sensor measurements*, a drone then *estimates related system states* and then passes the states to its *navigation algorithms* to determine how 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, triggered by the data from Inertial-Measurement-Unit (IMU) sensors. In the

following, the most popular state estimation algorithm and the common adjustment in navigation control are introduced.

Common State Estimation Algorithms. A drone usually uses state estimation methods to process sensor readings from accelerometers, gyroscopes, magnetometers, GPS signals, and barometers. Extended Kalman Filters (EKF) and its variants are the most popular state estimation methods [24], [25], [26], [27]. ArduPilot EKF system estimates 24 states for drone control. This paper focuses on the position and velocity estimation because the goal is to manipulate drone positions to affect flight paths. Readers interested in the details of these EKF algorithms can refer to [9].

Several common assumptions are adopted in this paper. First, assume that the defense is 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, this paper considers that the proposed attack is performed when the system is in *a steady state*, such that it is feasible to 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 [25], [28], [29].

Bad Data Detection. The ArduPilot EKF uses a common anomaly detection algorithm to determine if a sensor reading is acceptable, by checking if the following condition is true:

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

where τ is a pre-set threshold, inn_N and inn_E are the innovations (i.e., differences) between position predictions and position measurements 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 obtained based on the covariance matrix of the EKF and the GPS position accuracy information; they are regarded as constants in a steady-state system, based on our observations and many papers such as [4], [25], [28], [29], [30], [31], [32].

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 [33]. 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 2. 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.

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 $P^{EKF}(t)$, drifting away from the track, and it then performs the following adjustment. First, the distance between $P^{EKF}(t)$ and its projection on the track $P_{proj}^{EKF}(t)$ 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. 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 $track\_desired\_max$. The proposed attack exploits this adjustment to achieve drone position manipulation.

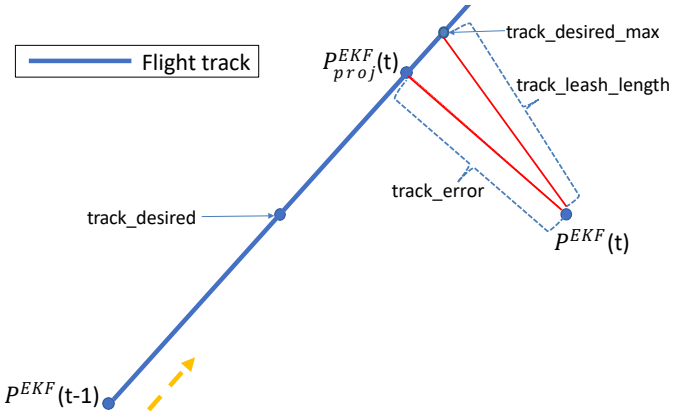

Fig. 2. Adjustment in path-following.

2.3 Attack Methods: Smart GPS Spoofing

After an invading drone enters the restricted zone, GPS spoofing is used to compromise its GPS position and velocity readings, to feed carefully crafted inputs to the drone state estimation algorithms in order to mislead its navigation control. This paper focuses on manipulating the drone position in a 2D plane of longitude and latitude[1]. In ArduPilot simulations, we pass the GPS inputs with MAVLink *GPS_input* messages [34] to emulate a covert spoofing, such that the proposed attacks on the navigation control can be evaluated, which are the focus of this paper.

In addition, we have also verified the feasibility of practical GPS spoofing on real drones. Using existing SDR tools, we have conducted GPS spoofing on a SkyViper GPS drone for testing, whose firmware is a variant of ArduPilot. In the experiment, the civil GPS signals are overpowered by the shifted GPS signals transmitted via a BladeRF A9 card from a moderate distance [17]. On a warm boot, the drone usually locks on the spoofed GPS signals after a short delay. This is verified by reading the GPS raw inputs from the drone via its MAVLink interface [34].

3 Basic DPM (bDPM) Attack

In this section, we introduce the *basic Drone Position Manipulation (bDPM)* attack that directly uses the EKF estimated position states of a drone to craft spoofed GPS positions. Although obtaining the EKF states is impractical on a real system, this

[1]Because the altitude control is more critical than longitude and latitude, a drone usually uses both GPS and barometer for altitude control. To the best of our knowledge, we did not find a practical method to remotely attack barometers accurately for spoofing attacks. So, the attack on altitude is delayed for future investigation.

method helps us better understand the proposed attack on a complicated control system in order to build a baseline analysis model of the attack. To further develop a practical solution, in the next section, we will introduce the *measurement-based Drone Position Manipulation (mDPM)* attack that uses measured drone positions to craft spoofed GPS position inputs. In the following, we present the bDPM attack and then show its capability by formally analyzing where a drone can be redirected from its target.

3.1 Theoretical Foundation of bDPM

Notations. First, the related notations are defined as shown in Table 1. Because the proposed attack focuses on the drone position in a horizontal 2D plane, the drone's *real velocity* is considered 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; e.g., when a drone moves at 4 m/s to the Northeast, a velocity vector V^r as (2.81, 2.81) m/s is observed. In bDPM, for each GPS cycle, the spoofed GPS position inputs are built by first obtaining the estimated EKF 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. For example, when applying $I = (0, 10)$ m/s to the drone flying to the Northeast with V^r, i.e., injecting 0 m/s to the GPS position input in the North direction and 10 m/s to the GPS position input in the East, it is observed that the drone's real velocity V^r is quickly stabilized at (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 theoretical foundation of bDPM, consisting of three important propositions.

Proposition 1. In a bDPM attack, the drift velocity V^r_{drift} due to injections is proportional to injection rate I with an attack coefficient C^a defined as follows:

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

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 injection size in any direction that can pass the bad data detection scheme.

Justifications. Proposition 1 and Proposition 2 are corollaries of the results in our previous work [6], [7], [9]. In the bDPM, the injection rate I is chosen as a fixed value in each GPS injection, 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 converges to a constant in a steady state, the deviation of position estimation $V^{EKF}_{drift} = K_k \cdot \Delta$ also converges to a constant. Because the deviation will be corrected by the navigation algorithm in each interval, $V^r_{drift} = -V^{EKF}_{drift}$ also converges to a constant. In addition, $C^a \approx -K_k$, which can be regarded as the same constant for attacks with the injection sizes that can pass the bad data detection on the same drone in the same environment. C^a is an observed system property that helps us formulate the attack model for identifying the relationship between the deviation of a drone's physical position and the injection on the GPS inputs.

TABLE 1
Notations used in the bDPM Attack.
A notation may have a subscript N or E representing its North or East component.

K_k	Kalman gain
C^a	Attack coefficient
V^{EKF}	Drone velocity estimation
V^r	Real drone velocity when under attack
V^r_{drift}	Drone drift velocity
$I(t)$	Position injection rate at interval t
$P^{EKF}(t)$	Drone position estimation at interval t
$P^{GPS}(t)$	GPS position input at interval t

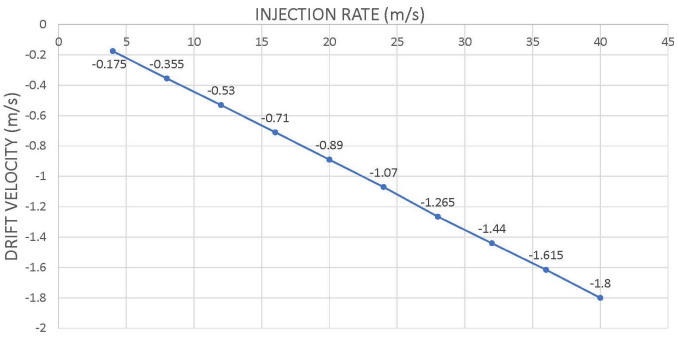

Fig. 3. Relationship between the drift velocity and the injection rate. As the injection rate increases, the drift velocity increases proportionally, and the attack coefficient C^a stays roughly constant.

These propositions have been validated with ArduPilot SITL simulations. Figure 3 shows the relationship between the injection rate (x-axis) and the observed drift velocity during bDPM attacks (y-axis). With the injection rate increasing from 4 m/s to 40 m/s, the drift velocity increases proportionally to the injection rate with coefficient C^a. Note that the sign of drift velocity only indicates the direction of the drift. Furthermore, Propositions 1 and 2 have been validated for injections in any direction in the 2D plane, when a proper constant injection rate that can pass the bad data detection is applied in the attack: North-only, East-only, West-only, South-only, and other combinations. In addition, C^a_E and C^a_N are observed as about -0.0455 and -0.0491 in all of these simulations.

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

Proposition 3 holds because the drone state estimation algorithms usually decompose the 3D positions and velocities into North, East, and Down sub-components [31]. (This is a common practice such that critical state estimations such as altitude are obtained first for fast responses.) Based on Propositions 2 and 3, the analysis of the attack result under an injection rate I in any direction can be divided 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 the measured attack coefficients C^a_E and C^a_N, the drift velocities of the drone in the North and East directions can be determined as: $V^r_{drift,N} = I_N \cdot C^a_N$ and $V^r_{drift,E} = I_E \cdot C^a_E$; then the attack result can be obtained by combining the drift velocities in the two directions.

3.2 bDPM Attack

The main steps of bDPM attack are presented in Algorithm 1. For ease of illustration, a single track is considered here. Given the flight track of a drone from origin (O_N, O_E) to destination (D_N, D_E), a redirected destination $R = (R_N, R_E)$, and drone position states, the algorithm builds a position injection for each GPS cycle in order to lead the drone to the redirected destination R. In particular, given R, in Line 3, a redirection vector $\mathbf{DR} = (DR_N, DR_E)$ is defined as the vector difference between the redirected destination (R_N, R_E) and the original destination (D_N, D_E). In Line 4, the total number of n cycles is determined to achieve the attack goal $\mathbf{DR}$. The "while" loop from Line 5 to Line 12 applies the injections. In Line 6, if more injections are still needed, an injection is added by Line 7 and Line 8: distribute the required injection into each cycle, i.e., in cycle t, an injection $I(t) = (DR_N/(n \cdot C_N^a), DR_E/(n \cdot C_E^a))$ is applied on the current position state $P^{EKF}(t)$ to build its position input $P^{GPS}(t), 0 \le t < n$. As a result, the navigation algorithm observes that the drone had drifted away from the track, and it will make an adjustment to "move" the drone back to the track, leading to its real position actually moves away from the track. 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 as illustrated in [9] in theory; it also can be measured in practice. The attack coefficient $C^a = (C_N^a, C_E^a)$ is measured in advance. In Line 9 and Line 10, as sufficient injections have been added, no extra injections are added and the state estimates are passed as the GPS inputs. (So, the drone will keep flying in the current heading without further adjustment.) In Line 11, the GPS inputs are sent to the drone; in Line 12, the procedure moves on to the next cycle. A video demonstration of a bDPM attack on SITL is at Youtube [35].

An example of bDPM attack is illustrated in Figure 4 to show the redirection of a drone to a specific destination in a 2D plane. In this example, 5 injection cycles are needed to achieve the required redirection; after 5 cycles, no injections are added and the drone flies towards the redirected destination in a path parallel to the original track.

3.3 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, a natural question is where a drone can be redirected by the bDPM attack. In the following, an analysis of a feasible range of redirected destination for a given target is presented to show

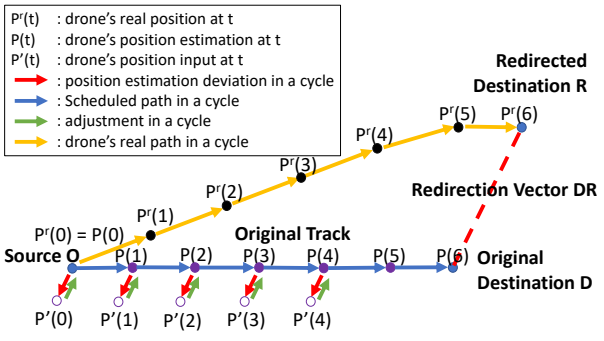

Fig. 4. Illustration of the bDPM attack.

Algorithm 1: bDPM 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 $P^{EKF}(t)$.

1 Initialization: $\{I(t)\} \longleftarrow \emptyset$, $t \longleftarrow 0$; n_0 = the remaining number of cycles on the original track;
2 $\mathbf{DR} = (DR_N, DR_E) \longleftarrow (R_N - D_N, R_E - D_E)$;
3 $n \longleftarrow max(\lceil DR_N/(I_N^{max} \cdot C_N^a)\rceil, \lceil DR_E/(I_E^{max} \cdot C_E^a)\rceil)$; find the total no. of injection cycles;
4 **while** $t \le n_0$ **do**
5 **if** $t < n$ **then**
6 $I(t) \longleftarrow (DR_N/(n \cdot C_N^a), DR_E/(n \cdot C_E^a))$; injecting until $t \ge n$;
7 $P^{GPS}(t) = I(t) + P^{EKF}(t)$; build fake position inputs;
8 **else**
9 $P^{GPS}(t) = P^{EKF}(t)$; fly towards R, not add injection;
10 send $P^{GPS}(t)$ as GPS position inputs;
11 $t \longleftarrow t + 1$;

the overall capability of bDPM, e.g., the range can be used to test if a redirected destination is reachable.

Because $(var_N^{inn} + var_E^{inn}) \cdot \tau$ 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, they can be plugged into (1), which results in (3):

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

Then, based on (2), the following result is obtained:

$$(V_{drift,N}^r/C_N^a)^2 + (V_{drift,E}^r/C_E^a)^2 = \lambda, \tag{4}$$

or

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

Equation 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 $\sqrt{1-(C_E^a/C_N^a)^2}$ (close to 0 in practice). After n attack cycles, the feasible range of the redirected destination will be

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

The accuracy and capability of bDPM are evaluated in Section 5.

4 Practical Measurement-based DPM (mDPM)

Although the bDPM attack is useful to develop the previous analysis model, it is impractical because it is very hard to obtain the EKF position estimation from a drone not under the defense's control. Therefore, in this section, the mDPM attack is developed, which builds the attack position inputs based on measured drone positions, instead of EKF states. In practice, many methods are available to measure the position and velocity of a drone from a

distance [4], e.g., using radar, radio signal triangulation, or other localization methods. In a restricted area, it is feasible to deploy measurement tools to obtain drone positions and velocities, and use such measurements to build spoofed GPS inputs. In this paper, the measured drone positions and velocities are obtained from the SITL simulator for tests.

4.1 Identifying a Practical Injection Method

First, let us show why the previous injection method does not work well in the new practical setting. As shown in Fig. 5, the thick blue line in the middle is the drone's original track. When the drone's real position is at $P^r(t)$, the drone is fed with a compromised position input by applying a position injection to the East (indicated by the blue curly brace), such that the autopilot system believes that the drone deviates from the track to the East at $P^{GPS}(t)$. Consequently, the control algorithm will compensate the difference by moving the drone to the West (shown as the red arrow), which makes the drone's real position move further to the West at $P^r(t+1)$ in the next cycle. However, if the same injection size (indicating by the blue curly brace) is used to build the next position input: $P^{GPS}(t+1)$ is equal to the real position $P^r(t+1)$ plus the constant injection as in the bDPM, $P^{GPS}(t+1)$ will also move to the West, which makes it closer to the drone's position estimation $P^{EKF}(t+1)$ in this cycle. In the next cycle, the smaller difference between $P^{GPS}(t+1)$ and $P^{EKF}(t+1)$ leads to a shorter West drift of the drone's real position; and the following GPS position inputs crafted with the same method will continue to move towards the drone EKF position estimation, as the drone's real position moves to the West. After some n cycles, the difference between the crafted GPS position input $P^{GPS}(t+n)$ and the drone position estimation $P^{EKF}(t+n)$ will be close to 0. After this, the drone's physical position will stop drifting to the West, and the drone will move on a track *parallel to the original track at a fixed distance (equal to the injection size)*. The above process has been also validated in the simulation.

The analysis of this process is presented in the following. Let us denote the difference between the crafted GPS position input and the drone position estimation (i.e., the innovation in drone state estimation) at cycle t as $\Delta(t) = (\Delta_N(t), \Delta_E(t))$; let us denote the constant injection to the East on the position inputs as $I = (0, I_E)$. In time cycle 0, $\Delta(0) = (0, I_E)$, since the real position and the drone position estimation are the same at the beginning. In cycle 1, because the drone's real position moves by $(0, -K_{k,E}\Delta_E(0))$ due to the innovation (where $K_{k,E}$ is the steady-state Kalman gain in the East), the position input will also move by $(0, -K_{k,E}\Delta_E(0))$, then $\Delta_E(1) = \Delta_E(0) - K_{k,E} \cdot \Delta_E(0) = (1 - K_{k,E}) \cdot \Delta_E(0)$. Similarly, $\Delta_E(2) = (1 - K_{k,E}) \cdot \Delta_E(1), \ldots$; then, we have $\Delta_E(t) = (1 - K_{k,E})^t \cdot I_E$. Furthermore, because $0 < K_{k,E} < 1$, the following result is obtained:

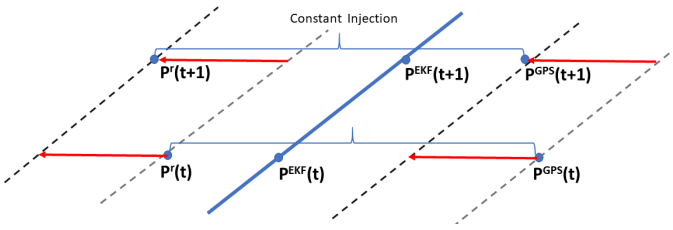

Fig. 5. Crafting GPS position inputs based on measured drone positions with constant injection sizes.

$$\lim_{t\to\infty} \Delta_E(t) = \lim_{t\to\infty} (1 - K_{k,E})^t \cdot I_E \to 0. \tag{7}$$

This analysis shows that the crafted GPS position inputs and the drone EKF position estimation will eventually converge. Since the difference between the GPS position input and the drone real position is the constant injection, the difference between the drone's real position and the drone EKF position estimation will also stabilize.

New Attack Strategy. The above analysis shows that the previous injection method will have limited effects, because the maximum drift distance is limited by the injection size. However, as the injection size must be smaller than a threshold determined by the bad data detector, the drift distance is also limited by the threshold. Therefore, a different attack strategy must be identified in mDPM to achieve larger redirection distances.

With careful analysis and testing, we propose to increase the injection rate linearly over time. In particular, assume the injection rate applied to the GPS position input is: $I = I^0 + X^I \cdot t$, i.e.,

$$I = (I_N, I_E) = (I_N^0 + X_N^I \cdot t, I_E^0 + X_E^I \cdot t) \tag{8}$$

where the base injection rate I^0 is a constant determined based on the drift velocity to be induced on the drone, X^I is the increase of injection in each cycle, and t indicates the t-th attack cycle. How to determine I^0 is presented in Section 4.2. Under this attack strategy, the following propositions are obtained:

Proposition 4. In an mDPM attack, the drift velocity V^r_{drift} due to injections that can pass the bad data detection is proportional to the injection increase rate X^I with an attack coefficient $\hat{C}^a$ defined as follows:

$$\hat{C}^a = (\hat{C}^a_N, \hat{C}^a_E) = (V^r_{drift,N}/X^I_N, V^r_{drift,E}/X^I_E) \tag{9}$$

for $X^I_N \neq 0$ and $X^I_E \neq 0$; if $X^I_N = 0$, $\hat{C}^a_N = 0$; if $X^I_E = 0$, $\hat{C}^a_E = 0$.

Proposition 5. For attacks on the same drone in the same environment, attack coefficient $\hat{C}^a$ keeps unchanged for any injection size that can pass the bad data detection in any direction.

Proposition 6. When applying an 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 in the East direction with $I_2 = (0, I_E)$, respectively.

The justifications of Propositions 4, 5, and 6 are similar to Propositions 1, 2, and 3 because they both exploit the same adjustment of navigation algorithm. In the following, the unique requirement of mDPM is further explored and more interesting results are obtained on how the injection size should be increased.

Although there are many potential methods to build the compromised position inputs, most of them will not be effective due to the specific setting of drone control and bad-data detection. For example, simply increasing the injection rate may not work in mDPM. (For simplicity, the subscripts indicating the directions are omitted in the following paragraph. The following analysis works for any direction.) In particular, when $I(t+1) - I(t)$ monotonically increases, $\lim_{t\to\infty}(I(t+1) - I(t)) \to \infty$; V_r^{drift} will also increase over time. When $I(t)$ increases but $I(t+1) - I(t)$ monotonically decreases, and $\lim_{t\to\infty}(I(t+1) - I(t)) \to 0$, the innovation will converge to 0, which results in the similar results as the constant injection case. Therefore, the following result is discovered.

Proposition 7. To successfully perform the mDPM attack, the injection rate should increase at least linearly, and the increase amount per cycle should have a finite upper bound.

The proof of Proposition 7 is as follows. Let us denote the innovation, i.e., the difference between the GPS input and the EKF position estimation at cycle t as $\Delta(t)$, and the position injection on the GPS input at cycle t as $I(t)$. Then, in cycle $t+1$, with Kalman gain K_k, the following relation is obtained:

$$\Delta(t+1) = \Delta(t) - \Delta(t) \cdot K_k + I(t+1) - I(t), \tag{10}$$

Equation (10) is based on the following facts. First, the change of Δ in cycle $(t+1)$ is determined as $\Delta(t+1) - \Delta(t)$. Although such a change cannot be directly obtained, it can be decomposed into two components: (a) the change of the difference between the real position of a drone and its estimated position; (b) the change of the difference between the GPS position input and the drone's real position. The first component is equal to $\Delta(t) \cdot K_k$ because the real position moves away from the position estimation by $\Delta(t) \cdot K_k$, due to the innovation $\Delta(t)$ at the last cycle. The second component is $I(t+1) - I(t)$ due to the change of the injection. Adding them up, we have the difference between $\Delta(t+1)$ and $\Delta(t)$, which leads to (10).

Based on (10), the following three types of injection methods can be analzyed:

1) If $I(t)$ increases linearly, i.e., $I(t) = I^0 + X^I \cdot t$, and $I(t+1) - I(t) = X^I$. Then, $\Delta(t+1) = \Delta(t) - \Delta(t) \cdot K_k + X^I$. If $-\Delta(t) \cdot K_k + X^I > 0$, $\Delta(t+1)$ will continue increasing until $-\Delta(t) \cdot K_k + X^I = 0$. Otherwise, if $-\Delta(t) \cdot K_k + X^I < 0$, $\Delta(t+1)$ will continue decreasing until $-\Delta(t) \cdot K_k + X^I = 0$. In either case, $\Delta(t+1)$ will be finally equal to $\Delta(t)$. Then, $\Delta(t+1)$ will keep unchanged afterwards. Because $\Delta(t+1)$ keeps unchanged but the injection increases linearly, the real drone position moves away from the GPS position input with a constant V^r_{drift}.
2) If $(I(t+1) - I(t))$ monotonically decreases, and $\lim_{t\to\infty}(I(t+1) - I(t)) \to 0$, let us denote $I(t+1) - I(t)$ as $A(t+1)$. Therefore,

$$\Delta(t+1) = \Delta(t) \cdot (1 - K_k) + A(t+1).$$

Now given a time T ($T \gg 0$), another series is defined as:

$$B(t) = \begin{cases} \Delta(t) & \text{t < T} \\ \Delta(T-1) \cdot (1 - \bar{K_k}) + A(T) & \text{t = T} \\ B(t-1) \cdot (1 - \bar{K_k}) + A(T) & \text{t > T} \end{cases}$$

where the $\bar{K_k}$ is the upper bound of K_k for $t \geq T$, $A(T) = I(T+1) - I(T)$, then for time interval $(t-1)$ to t, then

$$B(t) - A(T)/\bar{K_k} = (1 - \bar{K_k}) \cdot (B(t-1) - A(T)/\bar{K_k})$$

and after $(t - T + 1)$ time intervals,

$$B(t) - A(T)/\bar{K_k} = (1 - \bar{K_k})^{t-T+1} \cdot (B(T-1) - A(T)/\bar{K_k}).$$

Since $0 < \bar{K_k} < 1$, then

$$\lim_{t\to\infty} B(t) - A(T)/\bar{K_k} \to 0,$$

and

$$\lim_{t\to\infty} B(t) \to A(T)/\bar{K_k}.$$

Because

$$\lim_{T\to\infty} A(T) \to 0,$$

then

$$\lim_{t\to\infty} B(t) \to 0.$$

As $\Delta(t) \leq B(t)$, and $\Delta(t) > 0$,

$$\lim_{t\to\infty} \Delta(t) \to 0.$$

3) If $(I(t+1) - I(t))$ monotonically increases, and $\lim_{t\to\infty}(I(t+1) - I(t)) \to +\infty$, from (10), $\Delta(t+1) > I(t+1) - I(t)$. Therefore $\lim_{t\to\infty} \Delta(t) \to +\infty$. In practice, the system will be alerted by the bad data detector after the innovation is over a threshold, which leads to the attack failure.

Based on the above analysis, we conclude that, to successfully perform the attack, the injection rate should increase at least linearly, and the increase step per cycle should have a finite upper bound.

In summary, we need to carefully determine a proper increasing injection to move the drone away from its original track and not being detected. Although more complicated injection methods could be developed, we must carefully monitor the innovations caused by the injections, which is another problem to be explored in future investigation.

4.2 mDPM Attack

An mDPM attack has the similar procedure as the bDPM attack, but with the spoofed GPS position inputs constructed based on measured drone positions. The main steps of mDPM are shown in Algorithm 2. Based on the above analysis, the spoofed GPS

Algorithm 2: mDPM Attack Algorithm.

input: Original destination (D_N, D_E);
Redirected destination (R_N, R_E);
Drone position measurements $P^m(t)$.

Initialization: $\{I(t)\} \longleftarrow \emptyset$, $t \longleftarrow 0$;
n_0 = the remaining number of cycles on the original track;
$\mathbf{DR} = (DR_N, DR_E) \longleftarrow (R_N - D_N, R_E - D_E)$;
$n \longleftarrow max(\lceil DR_N/(I^{max}_N \cdot C^a_N)\rceil, \lceil DR_E/(I^{max}_E \cdot C^a_E)\rceil)$;
while $t \leq n_0$ **do**
  if $t < n$ **then**
    $I(t) \longleftarrow (DR_N/(n \cdot \hat{C}^a_N) \cdot t + I^0_N, DR_E/(n \cdot \hat{C}^a_E) \cdot t + I^0_E)$; injecting until $t \geq n$;
    $P^{GPS}(t) = I(t) + P^m(t)$; build fake position inputs;
  else
    $P^{GPS}(t) = I(n-1) + P^m(t)$; fly towards to R, injection size not increasing any more;
  send $P^{GPS}(t)$ as position input;
  $t \longleftarrow t + 1$;

position inputs are constructed with a linearly-increased injection rate for mDPM attacks in Line 7 and Line 8. Assume that the flight track of a drone is known in advance, the shifted GPS positions are built based on the drone's original destination (D_N, D_E), a redirected destination (R_N, R_E), and the maximum injection rate (which is determined by the bad-data detector and specific parameters of a drone [9] and can also be measured). The vector difference between the redirected destination and the original destination is defined as **DR** in Line 3. In Line 4, the total number of attack cycles n is determined for achieving this redirection vector. $I_0 = (I_N^0, I_E^0)$ is determined as $I_N^0 = V_{drift,N}^r / C_N^a$ and $I_E^0 = V_{drift,E}^r / C_E^a$, where $(V_{drift,N}^r, V_{drift,E}^r) = (DR_N/n, DR_E/n)$ is the drift velocity to be achieved; and the attack coefficient C^a is defined in Proposition 1. Under this setting of I_0, the innovation between the GPS input and EKF position estimation in the initial attack cycle will stabilize at this value and result in the drift velocity $(V_{drift,N}^r, V_{drift,E}^r)$ immediately. In simulations, we find that even if I_0 is set to different values (e.g., 0), the innovation will still converge to the previous setting values $I_N^0 = V_{drift,N}^r / C_N^a$ and $I_E^0 = V_{drift,E}^r / C_E^a$ very quickly. Line 5 to Line 12 are used to implement the mDPM attack. When injections are applied by Line 6 to Line 8 (in each attack cycle t, $0 \le t < n$,) an injection $I(t) = (DR_N/(n \cdot \hat{C}_N^a) \cdot t + I_N^0, DR_E/(n \cdot \hat{C}_E^a) \cdot t + I_E^0)$ is added to the measured position. Attack coefficient $\hat{C}^a = (\hat{C}_N^a, \hat{C}_E^a)$ is defined in Proposition 4 and can be measured in advance. Line 9 to Line 10 is the case that sufficient injections have been added; so, after that, the injection size is not changed. The GPS position input is sent to the drone in Line 11, and the algorithm goes to the next cycle in Line 12.

4.3 Range of Compromised Destination under mDPM attacks

To show the capability of mDPM attack, the feasible range of redirected destinations under the mDPM attack is further analyzed for determining if the mDPM attack can achieve a given redirected destination. According to (3), the feasible range of redirected destination under one cycle of mDPM attack is as follows:

$$(V_{drift,N}^r / \hat{C}_N^a)^2 + (V_{drift,E}^r / \hat{C}_E^a)^2 = \lambda, \tag{11}$$

After n injection cycles, the feasible range of the redirected destination will be

$$((R_x - D_x)/\hat{C}_N^a)^2 + ((R_y - D_y)/\hat{C}_E^a)^2 = n^2 \cdot \lambda, \tag{12}$$

The accuracy and capability of mDPM are evaluated in Section 5.

5 System Instrumentation and Performance Evaluation

As the goal is to develop a practical solution, different from existing methods, this paper focused on a broadly-deployed open-source system ArduPilot. In the following, the system instrumentation is introduced, which not only helps us understand the critical issues in the system but also allows us to evaluate different practical attacks.

5.1 System Instrumentation

We have conducted extensive analysis and testing of ArduPilot Copter code to understand the state estimation and navigation algorithms [6], [7], [9], [11], [36]. The SITL module runs the same code as a real firmware to simulate a flight with a large set of common parameters. Although ArduPilot has a logging scheme that provides many states for debugging, this investigation need to look into some specific states, which are not supported by the existing logging facility. In order to better understand system dynamics and capture real-time states, we have enhanced the system log facility by instrumenting the source code of drone control algorithms (including the key control loops, EKF state estimation algorithms, and navigation algorithms), such that related system states are captured into its Dataflash logs for analysis. As a result, the testing system can observe and capture all key variables for control algorithms in each cycle, including all sensor readings, state estimation variables, control parameters, and system states. Furthermore, at a higher level, the MAVLink interface of a drone is used to obtain high-level states such as the simulated position of a drone (i.e., its "physical position" in the simulated world) and the corresponding position estimation for us to build spoofed GPS inputs.

Similar to a real drone, a SITL drone 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. We performed a covert GPS spoofing by making MAVLink messages as its GPS input, in the same way as it receives GPS messages from a GPS-capable device. Then, an attack testing program is built with the DroneKit Developer tools (*https://dronekit-python.readthedocs.io/en/latest/*) to obtain real-time drone states, craft GPS_INPUT messages, and send them to the drone via MAVProxy, same as a common Ground Control Station (e.g., *QGroundControl*) communicates with a real drone. A callback listener is installed to receive position state updates in the attack program.

As shown in the demonstration video of a bDPM attack [35], with the source location as (0, 0), the drone's destination is set to the waypoint of (500, 500) meters in the Northeast with a velocity of 4 m/s. After 20 seconds into the mission when the system entered a steady state, the GPS spoofing is started with an injection of 4.07 meter/per GPS cycle to the North. (The simulation usually takes less than 20 seconds for the drone to enter a steady state.) As shown in Figure 6.(b), the drone state (shown as the top drone icon) is still on its original track to the Northeast; but its real position (shown as the bottom 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 shifted 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 built this in-depth instrumentation platform for examining the control algorithms and the proposed attacks in great detail, which also facilitated the evaluation presented in the following.

5.2 Evaluation of basic DPM (bDPM)

5.2.1 Simulation Settings

In each simulation, to show the pure attack effect, the attack is not launched until the system enters a steady state, after the drone reaches its takeoff altitude and begin to fly to a preset waypoint. The common settings of drone parameters are used 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

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

Fig. 6. Snapshots in the bDPM Demo on ArduPilot SITL.

velocity accuracy of GPS input is 0.1 meter/second; a default drone starting velocity is 4 m/s. The simulations are repeated many times to measure the coefficients for the basic model presented in Section 3: for example, linear regression is used to find the attack coefficients $C_a^N = -0.0491$, and $C_a^E = -0.0455$ for this setting.

5.2.2 Accuracy of bDPM

As the goal is to mislead a drone to a redirected destination, the accuracy of bDPM is first evaluated, i.e., identifying the difference between the expected redirected destination and the actual final destination. For easy illustration, the injection direction is set to the East, and the total attack duration is set to 50 seconds. Consider the home position as the original point (0, 0), the original destination was set to (500, 500) meters in the Northeast in a local frame. *Note that various destinations in different directions or distances have been tested, and the observed attack effects are similar. So, we use the above example to show the basic effects here.* To evaluate the accuracy under different attack sizes, the size of **DR** (the vector difference between the redirected destination (R_x, R_y) and the original destination (D_x, D_y)) is varied from 20 to 100 meters. We consider a redirection size of 100 meters is sufficiently large to show the attack effect.

Table 2 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 simulations. The 4th row shows that the bDPM attack achieved very small errors (under 1.5%) for different redirection sizes, i.e., it can accurately redirect a drone to the intended destination.

TABLE 2
bDPM 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, the bDPM's accuracy is evaluated 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 3 and Table 4, the 1st row shows the injection direction; the 2nd and the 3rd rows show the injection sub-components in the North and the East directions; the 4th and the 5th rows show the errors in the North and the East directions; the 6th and 7th rows show the error rates in the North and the East directions. Clearly, the attack errors for these cases are still very small (under 0.9% in a sub-component), which shows the bDPM attack can accurately redirect the drone to different directions.

TABLE 3
bDPM 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 4
bDPM 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%**

5.2.3 Injection limitation and Feasible range

Although the above evaluation has shown that the bDPM can redirect a drone to any direction with high accuracy, the maximum redirection size 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, the injection rate is gradually increased until the system detects the large error term and raises GPS-fail alarms. These simulations are repeated many times to confirm the maximum injection rate for bDPM in each direction, which will then give us the largest redirection size **DR** for a given attack duration in the direction. Then the maximum redirection sizes in all directions are combined to outline the feasible range of the bDPM attack, i.e.,

TABLE 5
bDPM Maximum Redirection Size (1)

Injection Direction	E	W	N	S
Max Injection (m/GPS-cycle)	7.56	7.60	7.09	7.14
Max **DR** Size (m)	171.58	173.32	171.32	173.00

TABLE 6
bDPM 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

the drone can be redirected to any point within this range under the attack duration. In Table 5 and Table 6, the attack duration was 50 seconds, and attacks in 8 directions are tested to outline the feasible redirection range. 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 durations, the attack duration is varied from 20 to 100 seconds, and the corresponding maximum redirection sizes in 8 directions are determined. Then the feasible ranges under different attack durations are outlined 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 feasible range of a bDPM attack is correspondingly enlarged as the attack duration grows, which shows that the attack can redirect a drone further away when giving a longer attack duration.

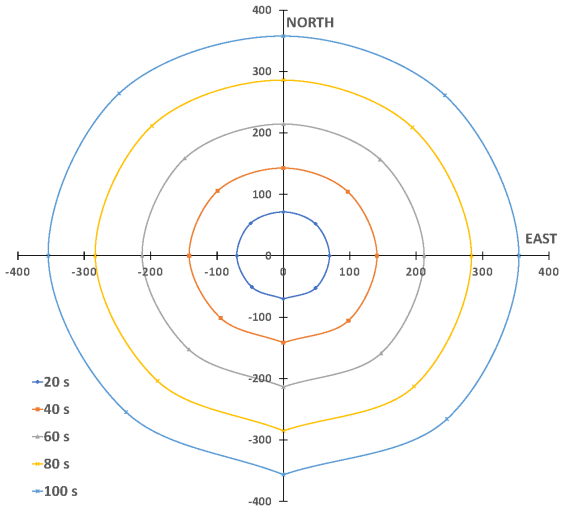

Fig. 7. Feasible ranges under different attack durations in bDPM.

5.3 Evaluation of Measurement-based DPM (mDPM)

5.3.1 Settings

The basic simulation settings for mDPM is the same as bDPM. Extensive simulations have been conducted to measure the attack

TABLE 7
mDPM Attack Error Rate Under Different X^I

X^I (m/s^2)	0.5	1	1.5	2	2.5
Exp. **DR** Size (m)	46.65	93.3	139.95	186.6	233.25
Act. **DR** Size (m)	46.76	93.34	139.84	186.54	232.96
Error Rate	**0.24%**	**0.043%**	**-0.079%**	**-0.032%**	**-0.12%**

TABLE 8
mDPM Attack Error Rate Under Different Attack Directions (1)

Injection Direction	E	W	N	S
X^I_N (m/s^2)	0	0	1	-1
X^I_E (m/s^2)	1	-1	0	0
Act. **DR** Size - North (m)	/	/	100.13	99.72
Act. **DR** Size - East (m)	93.34	92.88	/	/
Error - North (m)	/	/	0.13	-0.28
Error - East (m)	0.04	-0.42	/	/
Error Rate - North	/	/	**0.13%**	**-0.28%**
Error Rate - East	**0.043%**	**-0.45%**	/	/

coefficients for the mDPM model presented in Section 4: a regression method is used to identify the measurement-based attack coefficients $\hat{C}^a_N = -1.00$, and $\hat{C}^a_E = -0.933$; we further determine the parameters of this model as $I^0_N = \hat{C}^a_N / C^a_N \cdot X^I_N = 20.37 \cdot X^I_N$, $I^0_E = \hat{C}^a_E / C^a_E \cdot X^I_E = 20.51 \cdot X^I_E$.

5.3.2 Accuracy of mDPM

To evaluate the accuracy of the mDPM attack, as shown in the 1st row of Table 7, the injection increment velocity X^I is altered from 0.5 to 2.5 meter/$second^2$ to examine the difference between the expected redirected destination and the actual destination. The 2nd row shows the expected redirection size; the 3rd row shows the actual redirection size; and the 4th row shows the attack error rates under different X^I. Here the injection direction is set to the East and the total attack duration is set to 100 seconds. Consider the home position as location (0, 0), the original destination was set to (500, 500) meters in the Northeast in a local frame. Clearly, for these X^I's, the mDPM attack shows very small errors (under 0.45%), i.e., it can precisely mislead a drone to the expected redirection destinations.

Furthermore, the attack accuracy in 8 directions are tested as shown in Table 8 and Table 9. The 1st row represents the redirection direction. The 2nd and 3rd rows show the sub-component of X^I's in the North and East directions. The 4th and 5th rows show the actual redirection sizes in the North and East. The 6th and 7th rows show the errors between the expected **DR** and the actual **DR**. The 8th and 9th rows show the error rates in the North and East. For all cases, the mDPM showed a very small error rates (under 0.45% in a sub-component).

5.3.3 Injection limitation and Feasible Range

Because the maximum redirection size is determined by the maximum X^I, X^I is gradually increased to find its maximum value allowed until the bad data detector is triggered. These simulations are repeated many time to confirm the maximum X^I for mDPM in each direction, which will give us the maximum redirection size in the direction for a given attack duration. Then these maximum redirection sizes in different directions are combined to outline the feasible range of redirected destinations under the mDPM attack, i.e., the attack can redirect the drone to any point within this range. The attack duration was set to 50 seconds, and attacks in 8 directions are conducted to outline the range. As shown in Table 10 and Table 11, the maximum X^I for each direction varies

TABLE 9
mDPM Attack Error Rate Under Different Attack Directions (2)

Injection Direction	NE	NW	SE	SW
X_N^I (m/s^2)	1	1	-1	-1
X_E^I (m/s^2)	1	-1	1	-1
Act. **DR** Size - North (m)	100.10	100.18	99.65	99.75
Act. **DR** Size - East (m)	93.29	92.91	93.40	92.91
Error - North (m)	0.10	0.18	-0.35	-0.25
Error - East (m)	-0.01	-0.39	0.10	-0.39
Error Rate - North	**0.10%**	**0.18%**	**-0.35%**	**-0.25%**
Error Rate - East	**-0.011%**	**-0.42%**	**-0.11%**	**-0.42%**

TABLE 10
mDPM Max Redirection Sizes for Different Directions (1)

Injection Direction	E	W	N	S
Max X^I (m/s^2)	3.6	3.4	3.4	3.2
Max **DR** Size (m)	167.77	158.09	170.10	159.77

from 3.2 to 3.6 $meter/second^2$; the maximum redirection size in each direction varies from 156.80 to 170.10 meters.

Furthermore, the attack duration is varied from 20 to 100 seconds, and the corresponding maximum redirection sizes in 8 directions are determined. Based on the maximum redirection sizes under different attack durations, the feasible ranges of redirected destinations are outlined as shown in Figure 8. So, the feasible range of an mDPM attack is correspondingly enlarged as the attack duration grows, i.e., the attack can redirect a drone further away for a longer attack duration.

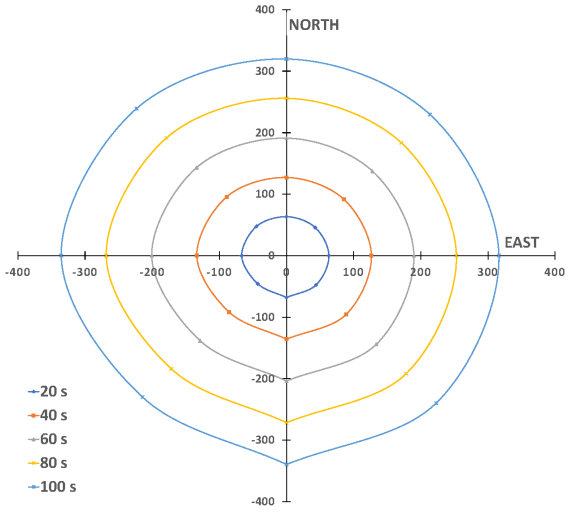

Fig. 8. Feasible ranges of redirected destinations under different attack durations in mDPM.

6 Related Work

As some existing drone countermeasures have been discussed in the introduction, let us further discuss other related work in this section. As discussed in Section II.B, a drone may be attacked at three steps of its control loop, i.e., attacking its onboard sensors [4], [5], [37], [38], its state estimation scheme [4], [6], [7], [9], or its navigation algorithms ([10], [11] and this paper).

(1) **Sensor-level attacks.** A consumer drone often uses an IMU with MEMS sensors (e.g., rate-gyroscopes, accelerometers, or magnetometers) to measure three-dimension angular velocities, accelerations, and magnetic readings, respectively. Based on these measurements, a drone can estimate its system states including position, velocity, and attitude [24], [39], [40]. Different from mission-critical systems (such as military drones), 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 [41], [42], [43], or compromising drone hardware components (e.g., MEMS sensors) via acoustic attacks to disturb its normal operation [4], [5], [37], [38]. Furthermore, although anecdotes on military GPS spoofing attacks have been reported (such as the capture of the US Sentinel drone by Iran [44]), the details have never been revealed. So, these attacks on military drones are considered beyond the scope of this paper, and this work focuses the civil GPS system on consumer drones.

TABLE 11
mDPM Max Redirection Sizes for Different Directions (2)

Injection Direction	NE	NW	SE	SW
Max X^I (m/s^2)	3.25	3.39	3.39	3.25
Max **DR** Size (m)	157.36	163.89	163.88	156.80

In a closely related project [4], a high-accuracy covert spoofer is built by manipulating the signal delays in the physical layer to spoof GPS signals arriving at a drone's GPS receiver. This 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, which can also help us implement our attack to manipulate GPS signals. Although the idea of spoofing GPS is similar to our approach, their work mostly focused on the manipulation in the physical layer, while our work considers all three levels and focuses on navigation algorithms. We can use their physical layer solution for covert spoofing; our work is complementary to theirs because we further developed a more complete model including both state estimation and navigation control. Two key differences at the higher level are: First, they built their own simple state estimator for analysis, while we exploited the mature state estimation schemes on an open-source system broadly used on many commercial drones, which makes our method more closer to practical systems. Second, they did not consider the navigation control such that they do not have accurate control over where the victim drone will fly to, while we exploited the navigation control and are able to accurately control the position of a drone.

(2) **Attacks on State Estimation.** Compromising system states is a common method to cause serious errors in control systems. The proposed attack 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 without being detected by bad data detectors [45], [46]. These FDI attacks are designed to exploit data transmission delays and uncertainty in large-scale distributed systems, different from the local setting on a drone with strict timing constraints.

Because sensors may generate wrong readings, most systems apply state estimation methods to handle such errors, as presented

in Section 2.2. Extended Kalman Filter (EKF) [24], [25], [26] and its variants are the most popular estimation algorithms on drones and many other control systems. Based on our previous research on state estimation algorithms [6], [7], [9], 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 the attack requirements in order to make the navigation algorithm properly adjust drone positions.

(3) **Attack Drone Navigation Controls.** We have pointed out the weaknesses of the most popular path-following algorithms [33] in Section 2.2. In this paper, the proposed attack utilizes the vulnerabilities in both state estimation and path-following to accurately mislead a drone to a redirected destination. Another project [10] focuses their attack on the GPS fail-safe mechanisms of navigation control. They propose an attack strategy for fail-safe mechanisms. However, they consider the state estimation together as a semi-blackbox without combining it with the navigation algorithm. As a result, although they can show some successful testing cases, they do not specify the conditions and the method for constructing successful spoofing signals that can pass the bad data detection and also affect the navigation algorithm to achieve quantitative control. While they can achieve some control over a target drone, they cannot achieve quantitative control as the proposed DPM. In contrast, we develop complete algorithms on how to guide an invading consumer drone to a redirected destination, and we can also determine where we can mislead a drone. To our best knowledge, this paper is the first to explore three fundamental components (guidance sensing, state estimation, and navigation control) together to achieve accurate attack effects.

7 Conclusions and Future Work

In this paper, we have examined three fundamental control components of a consumer drone together, identified the vulnerabilities at guidance sensing, state estimation, and navigation adjustment, and developed the DPM attack to exploit these vulnerabilities. The analysis and evaluation have shown that the DPM attack can accurately guide an invading drone to a redirected location for safe handling. We have further analyzed the feasible attack range to test if a drone can be redirected to a given destination. We believe this is the first work that is able to guide a consumer drone accurately to a desired destination.

Although the DPM aimed at common consumer drones, the idea of exploring the vulnerabilities of sensing, state estimation, and navigation control in a holistic method is applicable to many other existing and emerging autonomous systems. Exploring such a sequence of vulnerabilities could be a generic attack to many control systems. To stop such an attack, the vulnerability at each step need to be addressed. First, there have been various proposals to detect GPS spoofing [47], [48], [49]. However, to implement these enhancements on consumer drones may still have a long way to go [4]. Second, the EKF's anomaly detection algorithm can be improved to detect attacks such as DPM. We have briefly discussed the general idea of an effective detector in our previous work [9], and we will further investigate countermeasures based on physical properties of drones. Third, secure navigation algorithms to detect position/velocity manipulations should be investigated. We are also looking into other interesting research problems, such as how to apply the DPM to compromise a mission with multiple waypoints, or how to compromise the mission of a swarm.

Acknowledgment

This material is based upon work supported by the National Science Foundation (NSF) under Grant No. 1662487 and Office of Naval Research (ONR) Contract 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] J. Vanian, "Drone Registrations Are Still Soaring," *Fortune, http://fortune.com/2017/01/06/drones-registrations-soaring-faa/*, Jan. 06, 2017.
[2] A. Michel and D. Gettinger, "Analysis of New Drone Incident Reports," http://dronecenter.bard.edu/analysis-3-25-faa-incidents/, May 8, 2017.
[3] 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.
[4] 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.
[5] 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)*, 2017.
[6] W. Chen, Z. Duan, and Y. Dong, "False Data Injection on EKF-based Navigation Control," in *International Conference on Unmanned Aircraft Systems (ICUAS)*, 2017.
[7] 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.
[8] 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.
[9] W. Chen, Y. Dong, and Z. Duan, "Manipulating Drone Position Control," in *Proc. of IEEE Conference on Communications and Network Security(CNS)*, June 2019.
[10] 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 (TOPS)*, vol. 22, no. 2, pp. 1–26, April 2019.
[11] W. Chen, Y. Dong, and Z. Duan, "Compromising Flight Paths of Autopiloted Drones," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2019.
[12] E. Deligne, "ArDrone Corruption," *Journal of Computer Virology, vol.8, pp.15-27*, 2012.
[13] A. M. Shull, "Analysis of Cyberattacks on Unmmaned Aerial Systems," Master's thesis, Purdue University, 2013.
[14] A. Kim, B. Wampler, J. Goppert, and I. Hwang, "Cyber Attack Vulnerabilities Analysis for Unmanned Aerial Vehicles," *Infotech at Aerospace*, 2012.
[15] M. Monnik, "Hacking the Parrot AR.Drone 2.0," https://dronesec.com/blogs/articles/hacking-the-parrot-ar-drone-2-0, 2019.
[16] F. Samland, J. Fruth, M. Hildebrandt, T. Hoppe, and J. Dittmann, "AR. Drone: Security Threat Analysis and Exemplary Attack to Track Persons," *Proceedings of the SPIE*, vol. 8301, 2012.
[17] J. Cao, "Practical GPS Spoofing Aattacks on Consumer Drones," Master's thesis, University of Hawaii, https://scholarspace.manoa.hawaii.edu/bitstream/10125/73336/Cao_hawii_0085O_10909.pdf, Dec. 2020.
[18] 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.
[19] Drone Labs, "Drone detector," http://www.dronedetector.com/how-drone-detection-works/, 2016.
[20] DeDrone, "Secure your airspace now," http://www.dedrone.com/en/dronetracker/drone-protection-software, 2016.
[21] Paparazzi, "Paparazzi: The Free Autopilot," http://wiki.paparazziuav.org/wiki/Main_Page, 2003.
[22] OpenPilot, "DIY Drones: The Leading Community for Personal UAVs," https://diydrones.com/page/openpilot-1, 2021.
[23] ArduPilot, "ArduPilot Autopilot Suite," http://ardupilot.org/ardupilot/, 2021.

[24] K. M. Smalling and K. W. Eure, "A Short Tutorial on Inertial Navigation System and Global Positioning System Integration," NASA, Tech. Rep. NASA/TM–2015-218803, Sep. 2015.
[25] G. Welch and G. Bishop, "An Introduction to the Kalman Filter," *University of North Carolina at Chapel Hill, Chapel Hill, NC*, 1995.
[26] 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.
[27] R. Van Der Merwe, E. Wan, and S. Julier, "Sigma-point Kalman Filters for Nonlinear Estimation and Sensor-fusion: Applications to Integrated Navigation," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2004.
[28] C. Hajiyev and S. Y. Vural, "LQR Controller with Kalman Estimator Applied to UAV Longitudinal Dynamics," *Positioning*, vol. 4, no. 1, 2013.
[29] 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.
[30] P. Riseborough, "Inertial Navigation Filter," https://github.com/priseborough/InertialNav, 2015.
[31] ArduPilot, "Extended Kalman Filter Navigation Overview and Tuning," http://ardupilot.org/dev/docs/extended-kalman-filter.html, 2020.
[32] E. A. Wan and R. Van Der Merwe, "The Unscented Kalman Filter for Nonlinear Estimation," in *IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium (AS-SPCC)*, 2000.
[33] P. Sujit, S. Saripalli, and J. Sousa, "Unmanned Aaerial Vehicle Path Following: A Survey and Analysis of Algorithms for Fixed-wing Unmanned Aerial Vehicles," *IEEE Control Systems*, vol. 34, no. 1, pp. 42–59, Feb. 2014.
[34] MAVLink, "MAVLink Developer Guide," https://mavlink.io/en/, Dec., 2019.
[35] W. Chen, Y. Dong, and Z. Duan, "A Video Demo of Drone Position Manipulation Attack," https://youtu.be/kE0T4sFJZ7o, Feb 16, 2021.
[36] W. Chen, Y. Dong, and Z. Duan, "Attacking Altitude Estimation in Drone Navigation," in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, April 2018.
[37] N. O. Tippenhauer, C. Popper, 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*, 2011.
[38] Y. Son, H. Shin, D. Kim, Y. Park, J. Noh, K. Choi, J. Choi, and Y. Kim, "Rocking Drones with Intentional Sound Noise on Gyroscopic Sensors," in *24th USENIX Security Symposium (USENIX Security 15)*. Washington, D.C.: USENIX Association, 2015.
[39] O. J. Woodman, "An Introduction to Inertial Navigation," University of Cambridge, Computer Laboratory, Tech. Rep. UCAM-CL-TR-696, Aug. 2007.
[40] K. Gade, "The Seven Ways to Find Heading," *The Journal of Navigation, vol.69, pp.955-970*, 2016.
[41] Silent Archer, INC, "Silent Archer Counter-UAS system," http://www.srcinc.com/what-we-do/ew/silent-archer-counter-uas.html, 2016.
[42] B. S. Systems, "AUDS Anti-UAV Defence System," http://www.blighter.com/products/auds-anti-uav-defence-system.html, 2016.
[43] Battelle, "DroneDefender Technology," https://www.battelle.org/insights/case-studies/case-study-details/dronedefender-technology, 2021.
[44] S. Peterson and P. Faramarzi, "Exclusive: Iran Hijacked US Drone, Says Iranian engineer," https://www.csmonitor.com/World/Middle-East/2011/1215/Exclusive-Iran-hijacked-US-drone-says-Iranian-engineer, 2011.
[45] 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.
[46] F. Pasqualetti, R. Carli, and F. Bullo, "A Distributed Method for State Estimation and False Data Detection in Power Networks," in *IEEE International Conference on Smart Grid Communications (SmartGridComm)*, 2011.
[47] B. M. Ledvina, W. J. Bencze, and I. Galusha, B.and Miller, "An In-line Anti-spoofing Module for Legacy Civil GPS Receivers," in *Proc. of the ION ITM, San Diego, CA. Institute of Navigation*, 2010.
[48] D. S. De Lorenzo, J. Gautier, J. Rife, P. Enge, and D. Akos, "Adaptive Array Processing for GPS Interference Rejection," in *Proc. of the ION GNSS Meeting, Long Beach, CA. Institute of Navigation*, 2005.
[49] K. D. Wesson, B. L. Evans, and T. Humphreys, "A Combined Symmetric Difference and Power Monitoring GNSS Anti-spoofing Technique," in *Proceedings of the IEEE Global Conference on Signal and Information Processing, Austin, TX.*, 2013.

Wenxin Chen Wenxin Chen received the B.E. from South China University of Technology, Guangzhou, China, in 2012, and the M.Sc. from the Chinese University of Hong Kong, Hong Kong, in 2013, both in Information Engineering. He recieved his Ph.D. degree in Electrical Engineering in Department of Electrical and Computer Engineering at the University of Hawaii at Manoa. His research interest is about security and privacy issues in Unmanned Aerial Vehicles (UAVs), Internet of Things (IoTs), and cloud computing.

Yingfei Dong Yingfei Dong received his B.S. and M.S. degrees in Computer Science at Harbin Institute of Technology, P.R. China, in 1989 and 1992, Doctor degree in Engineering at Tsinghua University in 1996, and Ph.D. in Computer and Information Science at the University of Minnesota in 2003. He is a Professor at the Department of Electrical and Computer Engineering at the University of Hawaii at Manoa. His current research focuses on computer and network security, privacy, and secure distributed systems, including consumer drones, CPS/IoT, machine learning, and real-time control networking. He has published over 100 refereed research papers in international journals and conferences, and has served as associated editors for several international journals. He has served as organizers and technical program committee members for many IEEE/ACM/IFIP conferences. His research is supported by National Science Foundation, Office of Navy Research, Air Force Research Lab, Navy Applied Research Lab, and industrial partners.

Zhenhai Duan (S '97–M '03–SM '10) received the B.S. degree from Shandong University, China, in 1994, the M.S. degree from Beijing University, China, in 1997, and the Ph.D. degree from the University of Minnesota, in 2003, all in Computer Science. He is a Professor in the Department of Computer Science at the Florida State University. His research interests include computer networks and network security. Dr. Duan is a co-recipient of Best Paper Award of IEEE ICNP 2002, IEEE ICCCN 2006, IEEE GLOBECOM 2008, and ASE International Conference on Cyber Security 2012. Dr. Duan has served as a TPC co-chair of CEAS 2011, IEEE GLOBECOM 2010 Next-Generation Networking Symposium, and IEEE ICCCN 2007 and 2008 Network Algorithms and Performance Evaluation Track.