Un-Rocking Drones: Foundations of Acoustic Injection Attacks and Recovery Thereof

Jinseob Jeong*†, Dongkwan Kim‡, Joonha Jang*, Juhwan Noh*, Changhun Song*, and Yongdae Kim*
*KAIST, †Agency for Defense Development, ‡Samsung SDS
{*jeongjin-sub, dkay, cyber040946, juhwan, songch, yongdaek*}@kaist.ac.kr

***Abstract*—Drones equipped with microelectromechanical system (MEMS) inertial measurement unit (IMU) sensors are exposed to acoustic injection attacks. These attacks resonate sensors, compromising their output and causing drones to crash. Several mitigation strategies have been proposed; however, they are limited in terms of practicality as they cannot make the drone fly to its planned destination in the event of an attack.**

To remedy this, we aim at recovering the compromised sensor values for the practical mitigation of acoustic injection attacks. To achieve this, we first constructed a realistic testbed and delved into the implications of resonant MEMS sensors on drones. We discovered that sampling jitter, which refers to the inconsistent timing delay in retrieving sensor values, has a significant impact on drone crashes during the attack. Note that while any real-time system needs to satisfy its real-time requirements, it does have sampling jitter owing to manufacturing errors or scheduling/operational overheads. The sampling jitter is negligible in terms of real-time requirements; however, we found that it became critical for drones being attacked. This is because the sampling jitter spreads the resonant sensor signals into the in-band range of the drones' control logic, thereby neutralizing the drones' safety mechanisms, such as a low-pass filter.

Considering the resonant signals affected by sampling jitter as noise, we developed a novel mitigation strategy that leverages a noise reduction technique, namely a denoising autoencoder. This approach recovers benign sensor signals from compromised ones for the resonant MEMS IMU sensors, without requiring other supplementary sensors. We implemented this prototype, termed UNROCKER, and demonstrated its capability through a series of experiments reflecting real-world scenarios. To facilitate future research, we released our source code and experimental data.

I. INTRODUCTION

Commercial drones have gathered considerable popularity for applications in delivery services, emergency rescues, and military operations [1]. Drones have high requirements in terms of their real-time responses for dealing with various conditions for stable flight. Thus, drones are equipped with a variety of sensors such as gyroscopes, accelerometers, barometers, global navigation satellite system (GNSS) sensors, and optical flow sensors. Although these sensors are indispensable for drones, they are exposed to distinct security threats unlike those in traditional software systems.

Network and Distributed System Security (NDSS) Symposium 2023
28 February - 4 March 2023, San Diego, CA, USA
ISBN 1-891562-83-5
https://dx.doi.org/10.14722/ndss.2023.24112
www.ndss-symposium.org

TABLE I: Past studies on the security of MEMS sensors

	Description	Gyroscopes	Accelerometers
Attack	Denial of Service (DoS)	[2]	[3]
	Spoofing	[3]	[4]
Mitigation	Mechanical Shielding & Damping†	[2], [3]	[3], [4]
	Circuit-Level Change†	[2], [3]	[3], [4]
	Sampling Rate Change†	[3]	[3], [4]
	Detection	[9], [10]	[9]
	Partial Sensor Value Recovery	[11], [12]‡	·
	Full Sensor Value Recovery	★	★

† These require hardware modification; thus, they are not directly applicable.
‡ This requires supplementary sensors that are unaffected by the attack.
★ These are our focus to make drones accomplish the assigned missions.

Sensors are used to measure physical properties as quantitative values. Therefore, by design, they can measure malicious signals transmitted by adversaries in addition to normal (benign) signals. Exploiting this, several studies have demonstrated the feasibility of attacks against various sensors in drones, that can cause the drones to crash. In particular, these attacks manipulate the values of microelectromechanical system (MEMS) inertial measurement unit (IMU) sensors, such as gyroscopes [2], [3] and accelerometers [3], [4], by resonating these sensors with acoustic noise.

Notably, *acoustic injection attacks* are becoming increasingly serious, practical security threats owing to the technical improvements in sonic weapons. For example, a long-range acoustic device (LRAD) [5], which is a type of directed energy weapon [6], is widely employed as an anti-drone solution [7]. Moreover, the significance of the acoustic noise effect is also well-known in the military. The military environmental test standard (Method 515.8 of MIL-STD-810H [8]) explicitly specifies the necessity of testing the system's robustness against acoustic noise in the development stage.

Several mitigation strategies have been proposed to address the attacks on MEMS sensors, as shown in Table I. However, these strategies entail limitations in terms of their practicality. For instance, mechanically shielding these sensors [2] and applying sampling randomization [4] may not fundamentally prevent these attacks, while requiring comprehensive analyses of their side effects on drones. In particular, heating problems need to be considered when applying mechanical shielding [2], [13]. Although sampling randomization may address spoofing attacks, it makes drones more vulnerable to a denial of service (DoS) attack [3]; this aspect is described in further detail in §V-F. Conversely, detection-based approaches have been proposed [9], [10]; however, these do not include contingency plans for the drone's flight. The recently proposed sensor recovery approach [11], [12] is promising; however, its current status remains immature as its recovery lasts only for a few seconds.

Consequently, these approaches cannot work against persistent attacks. Moreover, the current state-of-the-art recovery [11] approach requires other supplementary sensors; thus, it cannot be employed if all the sensors are compromised simultaneously. To summarize, thus far, a practical mitigation strategy that enables drones to continue their flight toward the target location has not been reported.

In this regard, it is necessary to determine whether compromised MEMS sensors can be recovered without requiring additional supplementary sensors. This would enable the drones to continue flying toward the target location and accomplish their assigned missions. To achieve this, we thoroughly investigated the implications of compromised sensor values on the drones' control logic. In particular, we focused on MEMS gyroscopes and accelerometers, because these are essential for controlling the attitude and position of drones.

We first developed an acoustic injection testbed that emulates the realistic behaviors of the drones under attack (§IV). Testing attacks with actual drones can be cost-expensive as these attacks can physically damage drones. However, the developed testbed enabled detailed investigations of the drones' behavior, without any risk of physical damage. We implemented this testbed on PX4, which is an industry-leading, open-source drone project. Notably, PX4 provides simulators that can test drone software for planned missions, under both the software- and hardware-in-the-loop (SITL and HITL) modes. Using PX4, we implemented software models of the resonant sensors to simulate attacks. Finally, we automated the simulation procedures for rapid and repeated analyses.

Using this testbed, we conducted a series of rigorous experiments, and thus, discovered that sampling jitter, which refers to the inconsistent timing delay in retrieving sensor values, has a significant impact on drone crashes during an attack (§V). Note that although a real-time operating system (RTOS) is guaranteed to satisfy the deadline for its designated operation, it does have sampling jitter owing to manufacturing errors or scheduling. *The sampling jitter does not significantly affect the drones in a benign situation [14]. However, under acoustic injection attack, it acts as a key factor in a drone crash, by spreading the resonated sensor signals across multiple frequencies, which include the in-band range of the drones' control logic.* Thus, it neutralizes the safety mechanisms of the drones, such as a low-pass filter (LPF). Consequently, even a small amount of resonant noise could cause the drone to crash in the presence of sampling jitter. Additionally, based on our investigations, we discovered several misbeliefs associated with previously proposed mitigation methods.

Our investigations revealed that, from the perspective of signal processing, the spread resonant signal caused by the sampling jitter can be considered as additive noise in the original signal. Based on this observation, we developed a novel mitigation strategy that leverages a noise reduction technique to recover benign MEMS sensor signals from the compromised ones, without requiring supplementary sensors (§VI). To this end, we used a denoising autoencoder (DAE), which is a popular neural network model used for noise reduction. We subsequently implemented a prototype of this proposed recovery model, UNROCKER. Thereafter, we demonstrated its capability through a series of experiments with real-world scenarios.

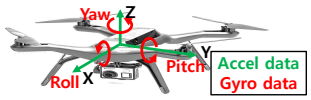


Fig. 1: Attitude and position of a drone.

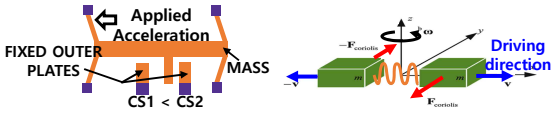


Fig. 2: Structure of MEMS accelerometer (left) and gyroscopes (right) [18], [19].

In summary, our contributions can be stated as follows:

- We developed an acoustic injection testbed for MEMS gyroscopes and accelerometers; this testbed enables automated testing of the influence of compromised sensor values on drones, without the risk of physical damage to the drones.
- Using this testbed, we conducted rigorous experiments and discovered that sampling jitter is the essential factor influencing drone crashes during attacks. Notably, sampling jitter has not been discussed in previous studies.
- During our investigations, we discovered that sampling jitter produces noise-like signals. Based on this finding, we developed a novel approach that adopts a noise reduction technique, particularly a DAE, to recover the benign values of compromised MEMS sensors.
- We implemented a prototype recovery system, UNROCKER, and demonstrated its capability through various experiments including real-world scenarios on physical sensors.
- We released our code, dataset, and demo videos [15].

II. BACKGROUND

A. Attitude and Position Control in Drones

The attitude and heading reference system (AHRS) is vital for drone flight as it computes the attitude (*i.e.*, roll, pitch, and yaw) and position (*i.e.*, forward/back, up/down, and left/right) of the drones, as depicted in Fig. 1. Specifically, the AHRS comprises IMUs with gyroscopes and accelerometers to measure the angular rate and acceleration, respectively. Based on the measured sensor values, it computes the attitude and position of the drones. To minimize errors, the values of various types of sensors can be combined in a complementary sense, which is referred to as sensor fusion. Notably, the extended Kalman filter (EKF) is a popular sensor fusion algorithm used in open-source drone projects [16], [17].

B. MEMS IMU Sensor

An IMU typically consists of a gyroscope and an accelerometer on a single integrated chip. Owing to their lightweight and low cost, MEMS IMUs are widely used in commercial drones for attitude and position control. The typical structure of the sensors in MEMS IMUs is illustrated in Fig. 2.

MEMS Accelerometer. An accelerometer is used to measure the acceleration of linear motion based on inertial force. It contains a proof mass on a spring that can travel back and forth along the sensing direction, *i.e.*, the axis. Upon applying a force, the mass tends to remain stationary owing to its inertia. Thus, the spring is stretched or compressed, and this creates variations in capacitance because of the fixed outer plates

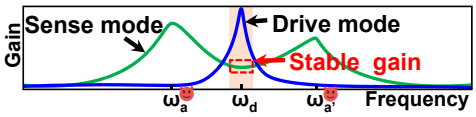


Fig. 3: Frequency response of an exemplary gyroscope in the drive and sense modes (ω_d: driving frequency; ω_a: resonance frequency maximizing the attack's implication).

(electrodes). Upon measuring this capacitance change (voltage change), the accelerometer induces the spring's displacement to compute acceleration a using the law of inertia and Hooke's law ($F_a = m \cdot a = -k \cdot x$, where m denotes the mass, k indicates the spring coefficient, and x denotes the induced displacement.)

MEMS Gyroscope. A MEMS gyroscope measures the angular rates based on the Coriolis effect. It operates in two orthogonal directions simultaneously, such as driving and sensing directions, as shown on the right of Fig. 2. Here, its operation for each direction is referred to as drive and sense modes, respectively [20], [21]. It contains a proof mass on a spring that continuously oscillates in the driving direction. In case the gyroscope rotates, the rotating force is applied to the oscillating mass. The vibration of this rotating mass creates an orthogonal force (*i.e.*, the Coriolis force) in the sensing direction, which is the cross-product of the rotational and driving directional axes. Eventually, this force causes changes in capacitance (voltage change). By measuring this capacitance change, the gyroscope computes the displacement of the mass in the sensing direction and uses it to compute the angular rate using the Coriolis force:

$$F_{coriolis} = -2m(\omega \times v) = -kx \tag{1}$$

where m, ω, v, k, and x denote the mass, angular rate, velocity of the proof mass, spring constant, and displacement of the mass, respectively.

The frequency response of each operation mode can be determined in the design stage of the gyroscope, particularly for its mechanical structures and coefficients. An exemplary gyroscope's response characteristics of the drive and sense modes over various frequencies are plotted in Fig. 3, where ω_d denotes the driving frequency. Although the driving frequency is designed as fixed, it can fluctuate in practice owing to external factors such as manufacturing errors, temperature, and aging. Therefore, a driving frequency is often set in the middle of the flat response region of the sense mode; thus, the gyroscope can produce stable output regardless of external factors [20], [22].

C. Acoustic Injection Attack

By design, MEMS gyroscopes and accelerometers are vulnerable to acoustic injection attacks, because they include a mechanically moving proof-mass. The resonance of this proof mass can cause fluctuations in the capacitance values, thereby compromising the final computed values. Moreover, in the case of MEMS gyroscopes, resonating the mass at a frequency where the sense mode responds intensively (the left or right peaks in Fig. 3) can amplify the implication of the attack. Notably, the resonance frequency of each sensor is determined by the coefficient of each element in the sensor, such as the springs or dampers. Thus, accurate estimation of this coefficient could be challenging. Instead, adversaries can identify the resonance frequency by injecting acoustic signals over a range of frequencies and monitoring the output

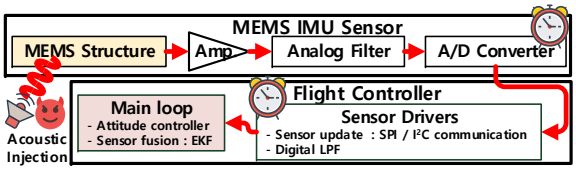


Fig. 4: Workflow of an acoustic injection attack, from the MEMS sensors' sensing structure to the drones' flight controller.

sensor values. Additionally, if the driving frequency of the target sensor is specified in its datasheet, adversaries can infer the resonance frequency with reference to this driving frequency [23], given that the resonance frequency resides near the driving frequency (Fig. 3).

The typical workflow of an acoustic injection attack is illustrated in Fig. 4. Initially, the adversary emits a malicious acoustic signal that resonates the MEMS structure of the sensor (*e.g.*, the proof mass). Subsequently, the sensed analog signal is converted into digital data through an analog-to-digital converter, and the digital data are transferred to the flight controller (FC) of the drones. Ultimately, the AHRS in the FC computes the attitude and heading based on these transferred sensor values to determine the appropriate actions for the rotors.

D. Sampling Sensor Values in Drones

Sampling is the process of converting a continuous signal into a series of discrete values [24]. Therefore, if the sampling frequency is inadequately low, the sampled values may represent the original signal in a distorted form, which is referred to as *aliasing*. To avoid this, the sampling frequency should be at least twice that of the original signal, as indicated by the Nyquist–Shannon sampling theorem [25].

By contrast, with respect to the acoustic injection attack, a compromised (resonated) sensor signal exists at a considerably higher frequency (several kHz). As this frequency is significantly higher than the sampling frequency of the drones (400 Hz for ArduCopter [17] and 250 Hz for PX4 [16]), the effect of aliasing markedly increases. This is because the resonated signal is under-sampled to a signal with a frequency ranging across only half of the sampling frequency. For instance, if the sampling frequency is 250 Hz and the resonated signal has a frequency of 5,040 Hz, the resonated signal would be under-sampled to a 40-Hz signal: $5{,}040 \mod 250 = 40$.

Based on the Nyquist-Shannon sampling theorem, an under-sampled resonant signal can be expressed as follows:

$$\hat{s}'(t) = A_i \cdot \cos(2\pi(F_i - N \cdot F_s)t + \phi) \tag{2}$$

where A_i denotes the induced gain (amplitude) of the injected acoustic signal, F_i indicates the induced frequency, and F_s denotes the sampling frequency. Here, N, t, and ϕ represent the number (for modular operation), timestamp, and phase of the under-sampled resonant signal, respectively. Thus, $A_i \cdot \cos(2\pi(F_i - NF_s)t + \phi)$ indicates the induced resonance effect considering the under-sampling effect of the induced signal at frequency F_i.

In real-world scenarios, the sampling frequency can vary because of various external factors, such as manufacturing errors, imprecise internal clocks, or scheduling issues. Thus, such factors can increase the implications of an acoustic injection attack to an even greater extent by distorting the original resonated signal.

III. Overview

Our goal is to *make drones accomplish their planned missions under acoustic injection attacks*. Accordingly, we aimed at recovering the benign sensor values from the compromised ones. An overview of our approach is illustrated in Fig. 5. First, to delve into the implications of compromised sensor values on drone behavior, we developed a testbed that can simulate an acoustic injection attack on a MEMS IMU. To simulate the attack, we conducted an empirical analysis of the resonated MEMS sensors to construct their software models (❶ and ❷ in Fig. 5). Thereafter, we conducted a series of experiments to investigate the implications of the attack on the drone system and, consequently, determined several new findings (❸). Notably, we discovered that sampling jitter is a critical factor that causes drone crashes. Considering a compromised signal affected by sampling jitter as noise, we developed a sensor recovery technique that attenuates the compromised sensor values and recovers the benign ones using a denoising technique (❹).

A. Threat Model

Here, we assume a strong adversary capable of remotely transmitting a strong acoustic and ultrasonic signal to the target drones to initiate a DoS or spoofing attack. This adversary possesses comprehensive knowledge of the structure and operating logic of the target drones. Therefore, they are completely aware of the model of the sensors employed on the drones, and their sampling frequencies. Consequently, they can generate and transmit a malicious signal at the resonance frequency of the sensors on the MEMS IMUs in the drones. Here, the adversary utilizes the resonance frequency that yields the peak (maximum) response of the sensor. Additionally, he or she can perform an attack on all MEMS sensors simultaneously, thereby incapacitating mitigation strategies that utilize other supplementary sensors [11], [12]. Lastly, the adversary can continue this attack for a prolonged period. Thus, unless the drone can mitigate the attack persistently, it will not be able to continue flying toward its target location and will fail to accomplish the assigned mission.

B. Analysis Target

We selected PX4 as the target drone software, which is an industry-leading, open-source drone project. PX4 enables both SITL and HITL simulations, which can be used to test drone software for planned missions; thus, we leveraged this feature to develop an acoustic injection testbed. Notably, SITL and HITL simulations are commonly employed in aerospace engineering to validate both flight control logic and hardware [26], [27]. SITL is an entirely software-based simulation used for testing the FC software of drones. In contrast, HITL is a hardware-assisted simulation that is widely used for evaluating the FC software using a physical board [26], [28]. As HITL utilizes physical hardware to run the flight control logic during simulation, it reflects hardware issues, such as operational or scheduling times. We used both simulation types to thoroughly investigate the implications of the compromised sensor values on a drone. Through a comparative analysis of SITL and HITL, we discovered sampling jitter (§V), a critical factor for acoustic injection attacks that has not been discussed in previous studies.

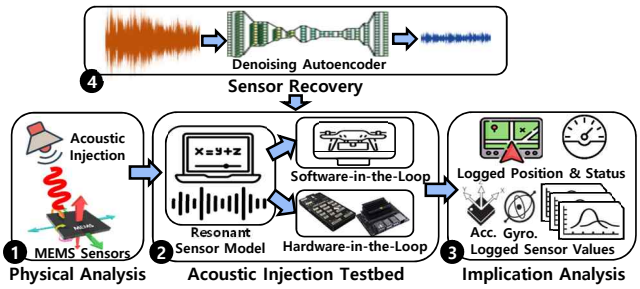


Fig. 5: Our approach overview (§IV: ❶&❷, §V: ❸, §VI: ❹).

Among the several commercial drones implemented based on PX4 [29], we selected 3DR Iris and Solo as our target drones, because they are widely used (as default) by PX4. As the HITL simulation requires actual hardware, we prepared an FC board named Pixhawk, which was used as default by the target drones. The Pixhawk board is equipped with two MEMS IMUs from InvenSense, namely, ICM-20689 (primary IMU) and ICM-20692 (secondary IMU), which include both gyroscopes and accelerometers. We also prepared a finished product of 3DR Solo to validate the simulation results.

C. Experimental Setup

All experiments were performed on a server equipped with Intel Core i9-10900K at 3.70 GHz, 128 GB DDR4 RAM, GeForce RTX 2070 GPU, and 2 TB SSD, operated on Ubuntu 20.04. To generate the acoustic signals, we prepared a function generator, RIGOL DG1000, which supports a wide range of frequencies. We then input the generated signals to a JBL Stage A120 speaker, through an Omnitronic MK-60DG amplifier.

IV. Acoustic Injection Testbed

As mentioned earlier, testing the acoustic injection attack in practice requires heavy experimental overheads. Specifically, it needs a speaker and a physical drone. The speaker must be able to aim the flying target drone and emit an attack signal precisely at the resonance frequency of the target sensors. Furthermore, such an approach can physically damage drones; which necessitates the use of several spare drones. By contrast, a software testbed enables rapid and reproducible analyses without these experimental overheads. Therefore, we developed a software-based acoustic injection testbed.

We implemented our testbed based on the PX4 simulator. Specifically, we developed software models of the resonant MEMS sensors and integrated these models into the simulator.

A. Modeling Resonant MEMS IMU Sensors

Our testbed uses emulated sensor values based on the software models of the resonant MEMS sensors on drones. As drones can be stationary or flying in real-world scenarios, these software models should account for both cases.

Researchers [4] successfully modeled the resonant signal of MEMS accelerometers under both stationary and dynamic cases. However, in the case of MEMS gyroscopes, researchers [30] only modeled stationary cases without considering dynamic cases. In this work, we simply adopted the previously reported model for accelerometers. For the gyroscopes, we developed a resonant signal model reflecting dynamic cases based on

the proposed stationary model [30]. Hereafter, we describe the related details.

Modeling Resonant MEMS Accelerometers. As described in §II, an acoustic attack signal can resonate with a MEMS accelerometer. When resonated, the accelerometer generates a resonated signal in addition to its original sensor signal. This resonated signal can be represented as the sum of the original signal and the resonant signal. In this context, a previous study [4] expressed the resonant signal $\hat{s}_{acc}(t)$ as follows:

$$\hat{s}_{acc}(t) = s_{acc}(t) + A_i \cdot \cos(2\pi F_a t + \phi) \tag{3}$$

where $s_{acc}(t)$ denotes the original sensor signal, A_i denotes the induced amplitude of the attack signal, F_a denotes the resonance frequency, and ϕ denotes the phase of the attack signal. We used this equation in our accelerometer model.

Modeling Resonant MEMS Gyroscopes. The operation mechanism of a MEMS gyroscope is distinct from that of the accelerometer, as it operates in two orthogonal directions, namely sensing and driving directions (§II). Therefore, resonance can appear in both directions, producing the resonant signal for each direction. Consequently, the output signal of a resonant gyroscope can be represented as a mixture of three signals: (1) a benign angular rate (2) the cross-product of a benign signal and a resonant signal in the driving direction, and (3) a resonant signal in the sensing direction. The first, second, and third terms in Equation 4 (and in Equation 5) denote the signal of (1), (2), and (3), respectively.

A prior study [30] proposed a resonant gyroscope model, particularly for stationary cases. Although this study is insightful, the authors only focused on stationary cases where the benign signal $\Omega(t)$ is zero. Thus, they only needed to consider the last term, *i.e.*, the resonant signal in the sensing direction.

To remedy this, we derived an equation model of resonant MEMS gyroscopes considering dynamic cases. We first modeled the displacement of the proof mass in resonated gyroscopes along the sensing direction, as follows:

$$\begin{aligned}\hat{x}(t) = S\Omega(t)cos(w_d t) + A_d\Omega(t)cos(\omega_{ac}t + \phi) \\ + A_s cos(\omega_{ac}t + \phi)\end{aligned} \tag{4}$$

where $\hat{x}$ denotes the displacement of the proof mass in the resonant gyroscope with moving positions; S, A_d, and A_s denote the scaling gains of the drive-mode-driven Coriolis force, acoustic-noise-driven Coriolis force, and direct impact on acoustic noise on the sensing direction, respectively.

To derive the angular rate from Equation 4, we divided both sides by the scale gain S. Then, we multiplied $cos(w_d t)$ to remove the ω_d component caused by modulation. Finally, Equation 5 is obtained by applying an LPF to remove the harmonic component generated by the modulation.

$$\begin{aligned}\hat{\Omega}(t) = \Omega(t) + \Omega(t)\Big(\frac{A_d}{S}cos\big((\omega_{ac} - \omega_d)t + \phi\big)\Big) \\ + \Big(\frac{A_s}{S}cos\big((\omega_{ac} - \omega_d)t + \phi\big)\Big)\end{aligned} \tag{5}$$

where $\Omega(t)\Big(\frac{A_d}{S}cos\big((\omega_{ac} - \omega_d)t + \phi\big)\Big)$ and $\Big(\frac{A_s}{S}cos\big((\omega_{ac} - \omega_d)t + \phi\big)\Big)$ denote the angular rate components of the $\Omega_{false}(t)$ signal induced by the acoustic signal that occurs in the driving and sensing directions, respectively.

The impact of an acoustic signal on MEMS gyroscopes can be analytically divided into two orthogonal directions: the sensing direction and the driving direction (§II-B). We calculated the relative magnitude G_{rel} of the two angular rates of each mode that makes up $\Omega_{false}(t)$.

The sensing directional excitation x_{acS} can be expressed as:

$$x_{acS} = x_s \cdot \cos(2\pi f_{ac} \cdot t + \phi) \tag{6}$$

where x_s denotes the sensing directional distance deviation caused by the acoustic signal, and $\cos(2\pi f_{ac} \cdot t + \phi)$ represents the sinusoidal form of the acoustic frequency ($\omega_{ac} = 2\pi f_{ac}$). Similarly, the driving directional excitation x_{acD} can be described as follows:

$$x_{acD} = x_d \cdot \cos(2\pi f_{ac} \cdot t + \phi) \tag{7}$$

where x_d denotes the driving directional distance deviation caused by the acoustic signal.

We assumed that x_d and x_s have similar orders of magnitude, that is, $x_d \approx x_s = x$. Recalling our threat model (refer to §III-A) and Fig. 3, this was a severe assumption for the driving directional impact. Although the sensing directional excitation is directly converted into the electric signal, the driving directional excitation should be transferred to the sensing directional deviation detected by the sensing mass. Upon applying the Coriolis force equation (Equation 1), the induced deviation in the distance to the sensing direction with respect to the excitation signal in the driving direction can be derived as follows:

$$x_{SacD} = \frac{2m\Omega}{k_s} \cdot \frac{\partial x_{acD}}{\partial t} = \frac{4\pi m f_{ac} x_d \Omega \sin(2\pi f_{ac} t + \phi)}{k_s} \tag{8}$$

Subsequently, the relative gain G_{rel} is derived through dividing the amplitude of x_{SacD} by that of x_{acS}, assuming that the magnitude of the distance deviation is similar to that of the acoustic excitation. Consequently, we can obtain a simplified equation, as follows:

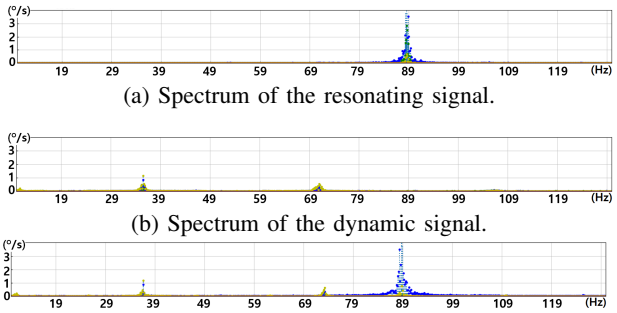
$$G_{rel} = \frac{|x_{SacD}|}{|x_{acS}|} = \frac{A_d \cdot \Omega}{A_s} = \frac{4\pi \cdot m \cdot f_{ac} \cdot \Omega}{k_s} \tag{9}$$

where m denotes the mass of the *proof mass*, k_s denotes the sense mode spring coefficient, and f_{ac} denotes the resonance frequency.

In particular, m is considerably smaller than the other coefficients (m : $2.5 \times 10^{-9} kg$, f_{ac} : 23 kHz, Ω : 10 rad/s, and k_s : 22.2 N/m). Here, we referenced the values from a previous study [30], and the application of these values yielded a relative gain G_{rel} of $3.3 \times 10^{-5} \approx 0$. This calculated result indicates that the angular rate of the sense mode mainly governs $\Omega_{false}(t)$ and that the angular rate induced by the drive mode is negligible. Thus, we can remove the term related to the drive mode, and the resonant gyroscope signal for dynamic cases can be deduced as follows:

$$\begin{aligned}\hat{\Omega}_{gyro}(t) \approx \Omega_{gyro}(t) + A_i \cdot cos(2\pi F_i t + \phi) \\ \Big(A_i = \frac{A_s}{S}, F_i = \frac{|\omega_{ac} - \omega_d|}{2\pi}\Big)\end{aligned} \tag{10}$$

where the induced acoustic frequency F_i corresponds to the difference between the acoustic signal frequency ($F_{ac} = \frac{\omega_{ac}}{2\pi}$) and the driving frequency ($F_d = \frac{\omega_d}{2\pi}$). For clarification, we refer to $\frac{A_s}{S}$ as induced amplitude A_i.



(a) Spectrum of the resonating signal.

(b) Spectrum of the dynamic signal.

(c) Spectrum of resonating signal with dynamics.

Fig. 6: Spectrum analysis of MPU-9250 gyroscope.

B. Empirical Analysis with Physical Sensors

To build our testbed more precisely, we conducted two empirical experiments in the real world. First, we validated the model's correctness for reflecting a gyroscope's response to the attack. Second, we examined the frequency and amplitude of resonant gyroscopes, to determine the practical range to be applied for the testbed. To this end, we prepared an Arduino board and additional gyroscopes, MPU-{6050,6500,9150,9250}, in addition to the IMU sensors in the Pixhawk board (*i.e.*, ICM-{20689, 20692}).

Model Validation. We theoretically demonstrated that the second term in Equation 5 (*i.e.*, the cross product of the benign signal and the driving directional resonant signal) is negligible. To validate this in the real world, we conducted experiments using IMUs, such as MPU-6500 and MPU-9250. We set up these IMUs on an Arduino board and injected an acoustic signal. To reflect the dynamic case, we used an additional mechanical vibration source that produces consistent vibrations on the gyroscope. In total, we performed three experiments, and the output spectra of the gyroscope for each experiment are illustrated in Fig. 6. First, we applied an attack signal at the resonance frequency (27 kHz) to the gyroscope under a stationary state (Fig. 6a), which exhibited a peak at 89 Hz. Subsequently, we applied only the vibration to the gyroscope (Fig. 6b), which exhibited two peaks at 35 and 70 Hz. Lastly, we applied both the attack signal and the vibration (Fig. 6c), which showed three peaks at 35, 70, and 89 Hz.

The output spectrum of the third experiment was approximately a mixed form of those from the first and second experiments. Notably, we could not observe any other noticeable peak caused by the driving directional excitation (*i.e.*, the second term in Equation 5). If the driving directional excitation is present, it should appear as a peak at the frequency of $F_m + F_i$, where F_m denotes the frequency of the mechanical vibrating source. As this peak does not appear, we concluded that the driving direction excitation is negligible.

Actual Values for Resonant Sensors. We also measured the practical values of the *induced frequency* F_i and the *induced amplitude* A_i, to use as a reference for generating the resonant gyroscope values in the testbed. Specifically, we analyzed the spectrum of the resonating gyroscope to determine the peak response in the acoustic injection tests; these results are presented in Table II.

TABLE II: Resonance analysis results of gyroscopes in IMUs

Model	Acoustic Freq. (Hz)	Max Amp. (deg/s)	Induced Freq. (Hz)
MPU-6050	27,230	39.0	7.2
MPU-6500	27,600	143.9	129.0
MPU-9150	27,500	22.6	193.1
MPU-9250	27,410	100.8	0.9
ICM-20689	27,100	108.8	205.9
ICM-20602	27,400	53.1	19.7

C. Testbed Implementation

To implement our testbed, we incorporated the developed software sensor models into the simulation framework (Gazebo v9.15) of PX4 [16]. Notably, the framework provides a high-fidelity six-degree-of-freedom simulation, which reflects the sensor values from a real flight [31], [32]. It also affords complete software-based (*i.e.*, SITL) and hardware-assisted simulation (*i.e.*, HITL). We customized the sensor drivers of PX4 that retrieve the sensor value to evaluate the effect of the resonance. Thereafter, we modified the messaging modules that manage commands to configure parameters for our software sensor models. For an in-depth analysis, we inserted simple code for logging the generated sensor values, status of the drones (*e.g.*, attitude and position), and timestamps of the injected signals and drones' operations. Finally, we built a script that could automate the experiments. In total, we implemented 1,347 lines of code to build our testbed in C++ and Python. For more details regarding our testbed, please refer to §B in Appendix and our website [15].

V. Implication Analysis of Resonant Sensor Data

In this section, we explain our findings through a series of SITL and HITL experiments using the developed testbed. In particular, although existing studies argue that sensor resonance itself induces drone crashes, we found that sampling jitter causes resonating sensors to crash drones.

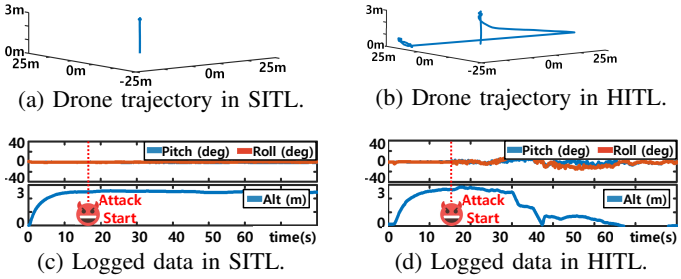
A. Resonant Sensors Alone Do Not Cause Crashes

It is widely accepted that resonant IMU sensors can directly cause drone crashes [2]. However, we discovered this is not always true.

Setup. For the investigation, we employed sine wave testing, which is commonly used to evaluate the robustness of the drones' controller during the design phase [33], [34]. Sine wave testing simply feeds various sine wave signals into the control logic of the target system by varying their amplitude and frequency. Similarly, we generated various resonant sensor signals and fed them to the target drone's control system.

Here, the resonating sensor signal can be represented as an additive sine wave, as shown in Equation 3 (for accelerometers) and Equation 10 (for gyroscopes). Although sine wave testing uses signals whose frequencies range over several tens of Hz, the test in this study involved signals with frequencies ranging from several hundred Hz to several kHz. Therefore, we carefully considered the under-sampling issues during our experiments.

For gyroscopes, we used an induced amplitude A_i between 0 and 4 rad/s ($\approx$230 deg/s) and an induced frequency F_i between 0 and 250 Hz. Here, for the induced amplitude, we chose the upper bound (4 rad/s) to cover the maximum values observed during the empirical analysis (Table II) and those reported by previous studies [2], [3]. We selected 250 Hz as



(a) Drone trajectory in SITL. (b) Drone trajectory in HITL.

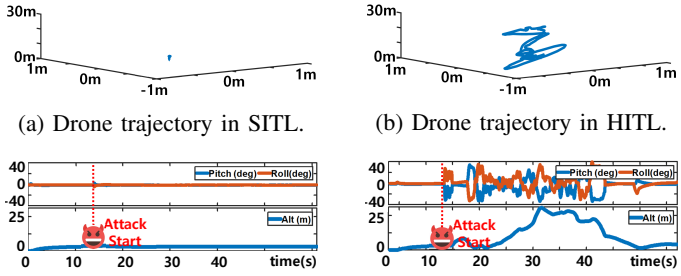
(c) Logged data in SITL. (d) Logged data in HITL.

Fig. 7: Results of resonance injection tests for gyroscopes.

the upper bound for the induced frequency, considering the sampling frequency (250 Hz) of the target drone.

For the accelerometers, we used a resonated amplitude A_i between 0 and 90 m/s^2 ($\approx$9 g) and an acoustic frequency F_{ac} between 5,000 and 5,250 Hz. Here, we selected a sufficient upper bound, as in the case of the gyroscopes; in other words, we referred to values measured in previous studies [4] for the amplitude. For the frequency, we selected the 250-Hz range, considering the sampling frequency (250 Hz) of the target.

Additionally, we set up a consistent environment and injected the acoustic signal into a drone under a stable state for precise experiments. More specifically, in each experiment, we first ran a simulation to fly the 3DR Solo drone and waited for it to hover at a stable altitude of 2.5 m ($\approx$15 s). We then injected noise signals into the drone and investigated its behavior.



(a) Drone trajectory in SITL. (b) Drone trajectory in HITL.

(c) Logged data in SITL. (d) Logged data in HITL.

Fig. 8: Results of resonance injection tests for accelerometers.

Results. Although it is widely accepted that IMU sensors cause drone crashes when resonated [2], we observed that the drones did not crash in most of the SITL experiment cases, for both the gyroscopes and accelerometers. By contrast, the drones crashed in all the cases of the HITL experiments.

Fig. 7 illustrates the drone's behavior and status for a resonant gyroscope, where the resonant signal has an F_i value of 100 Hz and an A_i value of 4 rad/s. Notably, the drone in the SITL experiment hovered in a stable manner, with subtle vibrations (Fig. 7a), whereas that in the HITL experiment crashed (Fig. 7b). In addition, the logged roll, pitch, and altitude of the drones in the SITL simulation remained consistent (Fig. 7c), whereas those in the HITL simulation significantly fluctuated immediately after the injection of the noise signal (Fig. 7d). The accelerometers exhibited similar behaviors as shown in Fig. 8; where the resonant signal has an F_i value of 5.125 kHz and an A_i value of 90 m/s^2.

Summary. From this investigation, we learned that resonant sensors do not always lead to drone crashes. Further, two questions were raised: 1) why did the drones not crash in most of the SITL experiments?, and 2) what caused the inconsistency between the SITL and HITL experiments? Hereinafter, we explore these questions.

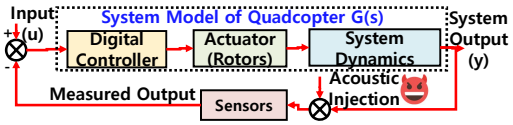
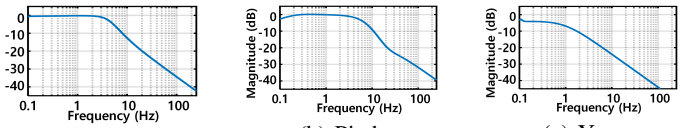


Fig. 9: Closed-loop system model of a quadcopter.



(a) Roll (b) Pitch (c) Yaw

Fig. 10: System response of the target drone. The bandwidths are 4.32, 5.37, and 0.005 Hz for (a), (b), and (c), respectively.

B. Robust Control Logic Can Prevent Drone Crashes in SITL Experiments

We further conducted causal analyses of the SITL experiments in which the drones did not crash. Subsequently, we discovered that such behavior mainly originated from two factors: 1) the sensor drivers contain digital LPFs that filter out unwanted signals and 2) the in-band frequency of the control logic is considerably smaller than that of the LPF boundary.

Digital LPFs. Digital LPFs are often employed in control systems such as those of drones to eliminate unwanted band signals (*i.e.*, out-band signals), including noise or harmonic components [30], [35]. In general, they are located in the software drivers of the drones that retrieve sensor values; thus, they are distinct from the analog LPFs inside the sensors.

In our experiments, the drone employed a digital LPF with a cutoff frequency of 30 Hz; this implies that the LPF filtered out all signals whose frequencies exceeded 30 Hz.

In-band Frequency Range of the Control Logic. Additionally, we conducted a simple test, where an acoustic signal was directly injected into the control logic under the SITL test by disabling the digital LPF. Here, we injected resonant signals by sweeping the induced frequency from 0 to 125 Hz, where 125 Hz denotes half of the sampling frequency of the drone.

As a result, we discovered that the drone crashed for signals with frequencies of just 0–5 Hz. In other words, for signals whose frequency ranged between 5 and 125 Hz, the drone successfully continued flying, without crashing. This demonstrated that the in-band frequency range of the drone's control logic was 0–5 Hz; thus, most of the resonant signals were ineffective.

To identify the in-band range of the target drone more precisely, we further conducted experiments by modeling the drone using a system identification method [36], [37]. System identification involves approximately building a model, namely a transfer function, for the target system using several input-output pairs. Prior research has introduced a similar approach for modeling a drone [9], [38]. Accordingly, we set up a closed-loop system model of a quadcopter, as illustrated in Fig. 9. Herein, we employed the angular rate as the input (u), along with its corresponding attitude control commands, such as

the roll, pitch, and yaw, as the output (y). Thereafter, we collected these values using our testbed, input them into MATLAB's system identification toolbox, and obtained a transfer function ($G(s)$).

The resulting transfer function is shown in Fig. 10. Notably, the -3-dB point is the de facto standard to measure the in-band range of a system, as it guarantees the response of the system in a stable manner [39]. The -3-dB points of the transfer functions were 4.32, 5.37, and 0.005 Hz, for roll, pitch, and yaw, respectively. That is, the in-band frequency range of the target drone was approximately 0–5 Hz.

Summary. In most of the SITL experiments, the drone did not crash owing to the digital LPF and the narrow in-band range of the control logic. This investigation revealed that simply excluding MEMS IMU sensors whose resonance frequency lies within the in-band range can prevent drones from crashing in the SITL experiments while drones crashed in all the HITL experiments. In this case what caused the resonant signals to bypass the digital LPF and affect the in-band range? Hereinafter, we explore this question.

C. Sampling Jitter as a Critical Factor for Drone Crashes

Resonant Sensor Signals are Dispersed in HITL Experiments. To identify the root causes of the drone crash in the HITL experiments, we first monitored the resonant sensor signals in both the SITL and HITL experiments before and after the low-pass filtering in the drones' control logic. After logging the sensor signals at each point, we investigated the spectrum of the sampled signals.

Fig. 11 illustrates the sampled resonant signals whose resonance frequency is 5,100 Hz. In an ideal case, this signal will be under-sampled to a signal whose induced frequency is 100 Hz; $5,100 \mod 250 = 100$, where 250 is the sampling frequency of the target drone. The blue line (❶) depicts the spectrum of benign sensor data, which are largely at low frequencies within the 'in-band' frequency. The green plots (❷) in the figure depict the spectrum of an ideal resonant signal from the SITL simulation, which does not include hardware issues. Note that the acoustic signal shows peak response at 100-Hz, which is still out-of-band. By contrast, the orange plots (❸) represent the resonant sensor signal before the low-pass filtering in the HITL simulation on a physical Pixhawk board. Notably, the peak at 100 Hz in the green plots was dispersed over multiple frequencies (0–125 Hz) in the orange plots owing to the sampling jitter. Lastly, the red plots (❹) indicate the dispersed orange plots after the signals were passed through the low-pass filter. Recall that the target drone employed a digital LPF with a cutoff frequency of 30 Hz. Because the dispersed resonant signal ($\leq$ 30) is within the range of the low-pass filter, it is not completely removed after filtering. As compared with the benign sensor data spectrum (❶), the residual resonant signal (❹) is substantially dominant. Consequently, it was passed to the control logic of the drone and affected its behavior. These unfiltered signals influence the sensor fusion filter (*i.e.*, EKF), preventing accurate estimation of the position or attitude. In compromised cases, we observed several consequences, such as EKF fail-over, position error, or sensor inconsistency error. For the details of these consequences, refer to Video-C on our website [15].

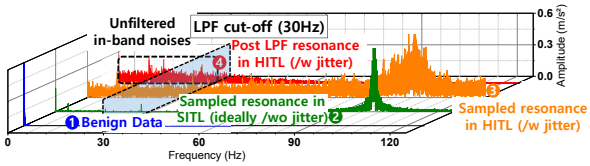


Fig. 11: Spectrum analysis of the sampled resonance signals.

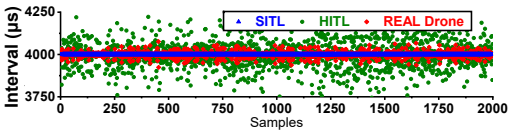


Fig. 12: Measured sampling intervals for each experiment.

Sampling jitter causes the dispersion of the resonant signals. If so, what caused the dispersion in the HITL experiments? Recall that a resonant signal is generally under-sampled as its frequency is considerably higher (several kHz) than the drone's sampling frequency (250 Hz). Notably, hardware developers are aware that such under-sampled signals are more susceptible to sampling jitter, which is essentially an inconsistent timing delay in retrieving the sensor values [40], [41]. Particularly, the under-sampled signals can be spread out by sampling jitter, as shown in Fig. 11. However, none of the previous studies on acoustic injection attacks have discussed this issue. Accordingly, we further investigated the implications of sampling jitter on the resonant signals. We simply measured the sampling interval after the retrieval of sensor values at the sensor drivers and before the sensor values were transferred to the control logic.

Fig. 12 illustrates the measured sampling intervals in each experiment. The average sampling interval (4 ms) corresponds with the sampling frequency (250 Hz) of the drone. However, several sensor values were sampled imprecisely, with large deviations. More specifically, although the standard deviation of the sampling intervals was nearly 0 in the SITL experiments, that in the HITL experiments was approximately 457 μs. Additionally, we conducted the same experiment with a commercial drone, 3DR Solo. As the drone utilized the same board as that employed in the HITL experiments, it also showed substantial sampling jitter, with a standard deviation of approximately 103 μs. This result demonstrates that sampling jitter is indeed present in real-world scenarios. However, is this sampling jitter sufficient to cause drone crashes?

In-depth Analysis of Sampling Jitter. To address the aforementioned question, we performed additional in-depth analyses for the sampling jitter. We conducted the same SITL experiments, but added an intentional random jitter during the retrieval of *resonant* sensor values in the sensor drivers. Thereafter, we investigated 1) drone crashes in the presence of jitter, and 2) the threshold of the jitter that caused the drone crashes for the sampled resonant sensor signals. For the gyroscopes, sampling jitter that was greater than 80 μs could cause the drone to crash. In the case of accelerometers, 6-μs sampling jitter was sufficient to crash the drone. For both sensors, the threshold sampling jitter that crashed the drone was considerably smaller than those in the HITL and actual drone experiments. Consequently, we concluded that the sampling jitter during hardware operation is a key factor influencing drone crashes during attacks.

We also measured the signal-to-noise ratio (SNR) to investigate the influence of sampling jitter on the original signals

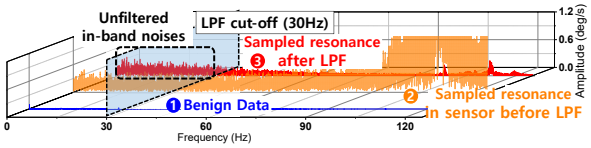


Fig. 13: Spectrum analysis of the real drone sensor.

in detail. Our SNR calculations for the resonant signal revealed that the SNR ranged from 15 to 33 dB in the absence of the sampling jitter, whereas it ranged from -23 to -5 dB in the presence of the jitter. Here, the negative SNR implies that the noise overpowers the original signal. Therefore, the resonance signal with sampling jitter negatively affects the safety of the drone control system.

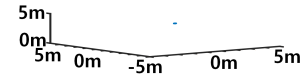
Impact of Sampling Jitter in Actual Sensors. Sampling jitter is inherent in an FC board, and it originates from multiple sources, primarily from hardware imperfections (refer to §V-E and §A in the Appendix). Thus, *the sampling jitter is not controllable*. To investigate the actual influence of the sampling jitter, we measured the resonant sensor signals using physical sensors under attack, and the results are presented in Fig. 13. Notably, the results are similar to those in the HITL test (Fig. 11). The blue line (❶) depicts the spectrum of benign sensor data. The orange plots (❷) represent the resonant sensor signal before low-pass filtering on a physical Pixhawk board while injecting actual sound with a speaker. Notably, the resonance signal with an induced frequency of 110 Hz is dispersed over multiple frequencies (0–125 Hz) as shown in the orange plots. As depicted in the red plots (❸), even though the *out-band* noises are diminished, the dispersed *in-band* signals are not filtered with an LPF. These *unfiltered in-band noises* are large enough to overpower the benign signals.

Summary. We investigated the inconsistent results between the ideal (SITL), and the hardware-related (HITL as well as actual sensors) experiments. Subsequently, we discovered that sampling jitter plays a critical role in drone crashes by spreading the resonant signals across various frequencies, including the in-band frequency of the drones.

D. Effect of Sampling Jitter in Actual Drones

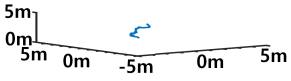
To confirm the effect of sampling jitter in the real world, we conducted an additional experiment using an actual drone, 3DR Solo. After inserting the resonant software sensor models into the drone's firmware, we set the drone to hover at an altitude of 2 m. Then, we initiated the attack to inject resonant sensor signals into the drone.

As actual drones naturally include sampling jitter, the generated resonant sensor signals are already affected by sampling jitter. To emulate the status without sampling jitter, we slightly modified the timestamp during the generation of the resonant signals, such that the resonant signals had an exact timestamp of 4 ms (250 Hz), which is the sampling frequency of the drone's sensor drivers.

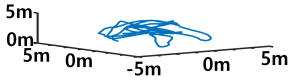


(a) Resonant gyro. w/o jitter.

(b) Resonant accel. w/o jitter.



(c) Resonant gyro. w/ jitter.



(d) Resonant accel. w/ jitter.

Fig. 14: Actual drone flights with and without sampling jitter.

As shown in Fig. 14, the presence of sampling jitter affected the output sensor signals and eventually crashed the drones for both the gyroscopes and accelerometers. This confirms that sampling jitter is a critical factor influencing drone crashes during an attack. For a demo, please refer to Video-B on our website [15].

E. Empirical Evidences of Sampling Jitter

Next, we investigated the potential causes of sampling jitter. As expected, multiple factors can cause sampling jitter. Here, we introduce three potential causes discovered during our investigation.

Scheduling Issues. First, control systems often employ an RTOS to satisfy the strict computational requirements. Although an RTOS ensures that its executable tasks are compliant with the deadlines, it does not ensure the deterministic execution of the tasks, *i.e.*, a task in an RTOS is scheduled to be executed within a specific deadline but not at a specific moment. Consequently, *scheduling jitter naturally occurs on the execution of a periodic task*. In practice, such scheduling jitter is prevalent in several control systems [42]–[44]. To confirm this, we measured the scheduling jitter of the sensor task for our target system, PX4 running an RTOS named NuttX. As a result, we observed a scheduling jitter of 124.6 μs with a standard deviation, which is comparable to the period of the acoustic frequency.

Operation Interval Mismatches. Second, each task in a drone is scheduled to be performed at designated intervals. However, the operation period of the drone's control logic may not match that of the sampling logic (in sensor drivers) retrieving the sensor values. Such mismatches may not cause an issue during benign operation, *i.e.*, in the absence of an attack, because the sensor loop ensures that the main loop receives new data in this setup. However, if the sensors are under attack, this misaligned time interval can cause jitters from the perspective of the main control logic. By investigating the source code of PX4 and ArduCopter, we discovered several cases in which the sensor drivers implemented in the drone firmware used time intervals that are different from those of the operation interval of the control logic. For instance, the sampling frequency (1000 Hz) of the InvenSense drivers in ArduCopter is 2.5 times higher than that (400 Hz) of the control logic. For detailed results, refer to Table VI in §A of the Appendix.

Imprecise Clocks. Each MEMS sensor includes its own clock source to independently process analog-to-digital (ADC) sampling. Nonetheless, this clock source was not synchronized with that of the FC. Such a mismatch can cause inaccurate clock timings when processing sensor values, thus resulting in jitters. During our experiments, we observed a 5 % precision error (Fig. 21 in §A of the Appendix).

F. Misbeliefs in Previous Mitigation Methods

During our investigations, we identified several misbeliefs associated with previous studies regarding mitigation strategies.

Low-Pass Filters Cannot Mitigate the Attacks. A previous study [4] proposed that employing LPFs can mitigate attacks. Notably, LPFs can be largely divided into two categories:

analog and digital. As analog circuits cannot completely suppress the electric signals at unwanted frequencies owing to hardware issues, a part of the resonant signals can remain and affect the drone's control logic. By contrast, digital LPFs can almost entirely suppress unwanted frequency signals. Nevertheless, when using either type of LPF, it is challenging to completely filter out the resonant signals affected by the sampling jitter. This is because the sampling jitter spreads the under-sampled resonant signals across the in-band frequency range, as discussed in §V-C. Thus, both analog and digital LPFs cannot mitigate acoustic injection attacks.

Sampling Randomization Increases Vulnerability to DoS Attacks. Prior studies [3], [4] have suggested that sampling randomization can prevent spoofing attacks in MEMS sensors. This approach is indeed promising to prevent spoofing attacks; however, during our experiments, we discovered that it increases vulnerability to DoS attacks (for a demo, refer to Video-D on our website [15]). As this mitigation randomizes the sampling phase, it essentially increases the possibility of under-sampling the resonant sensor signals. This under-sampling can affect the sampled signals in a similar manner as sampling jitter: by spreading the sampled signal across multiple frequencies, including the in-band range. In addition, ideal uniform sampling is fundamentally infeasible in actual hardware. Thus, such inherent nonuniform sampling makes actual drones more vulnerable to DoS attacks. Therefore, random sampling cannot be the ultimate solution to prevent such attacks.

VI. Recovering Resonant Sensor Values

Based on an in-depth analysis of the resonant sensor signals, we learned that they are spread across the in-band frequency range of the drone's control logic owing to the sampling jitter. These in-band resonant signals subsequently overlap the benign signals and eventually crash the drone. Then, how can we recover the benign signals from the compromised ones?

A. Design of Recovery System, UnRocker

Based on the findings (§V), we hypothesized that 1) the overlapping resonant signal can be considered noise from the perspective of benign signals, and 2) we could thus leverage noise reduction techniques to recover the benign signals. Our intuition is that the compromised signal is essentially a combination of the benign and resonant signals, as demonstrated in §IV-A. Thus, removing this noise component from the compromised signal would yield a benign signal.

First, we investigated if using conventional heuristic denoising filters could mitigate the impact of acoustic injection attacks. As popular heuristic filters, a Savitzky-Golay (Sav-Gol) smoothing filter [45] and a wavelet domain Wiener filter [46] were employed; the latter is noted as the *industry standard*. Using these filters, we attempted to thwart acoustic injection attacks, but both failed. For more details, refer to §VII.

The advent of deep neural networks (DNNs) contributes to overcoming the limitations of conventional heuristic approaches [47]. Among existing approaches, we chose a DAE [48], [49] as our basis. Notably, a DAE is widely applied to safety-critical systems, including medical imaging and various industrial processes [50]–[54]. As drones are also safety-critical systems, we considered that the DAE can be applied to them as well.

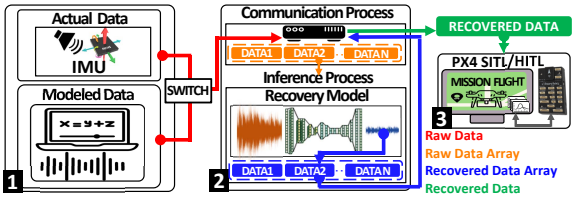


Fig. 15: Overview of our recovery system UnRocker.

Note that we do not claim that our approach is optimal. However, our unique contribution is that we applied a denoising approach to the compromised sensor signal under acoustic injection attacks, demonstrating that the attack can be effectively mitigated. Meanwhile, we also conducted a brief experiment that compares the DAE with recent advanced approaches; this also shows the robustness of DAEs (refer to §VIII).

A DAE is fundamentally an autoencoder; thus, it constitutes a cascaded form of an encoder and a decoder (Fig. 16). It converts a noisy input signal into a signal with much less noise through the encoding and decoding processes. Based on the DAE model, we designed a recovery system that we call UnRocker. An overview of UnRocker is illustrated in Fig. 15. UnRocker is designed to operate in the middle of the sensor drivers and an FC; thus, it directly recovers the digitized values from the sensors. First, UnRocker receives the sensor data from the sensor drivers. As these sensor data are transferred in a digitized form, a sufficient number of samples are required to accurately indicate the original analog signal. Thus, UnRocker stores the received data in a queue (by means of a sliding window) and transfers the stored values to its pre-trained DAE model. Then, it delivers the output recovered sensor values to the FC.

B. DAE Model Construction

For the DAE model of UnRocker, we simply borrowed a DAE model from previous work [55] as it has been widely used in various studies for signal denoising [51], [56], [57]. Note that this model was implemented with TensorFlow and Keras. We adjusted its hyper-parameters, such as the input size, step size, and epoch, by empirically iterating the training phase.

First, to set the input size, we considered the trade-off between the performance and recovery. It is obvious that a small number of input values would reduce the complexity of the network and accelerate recovery. However, few samples may not sufficiently express their tendency effectively; thus, the recovery performance would decrease. Considering the computational overhead and latency in our experimental setup, we empirically set the input size to 256, which comprises a signal of approximately 1 s.

For the batch size, we chose 32. Note that a large batch size reduces the training time, but it consumes large amounts of memory resources and sacrifices the training loss. By contrast, if the batch size is small, both the training accuracy and time increase. After iteratively testing different batch sizes, we observed that the training accuracy becomes saturated if the batch size is less than 32; thus, we used it.

For the number of layers, we also tested various combinations. For the accelerometer, we found that 18 is the best

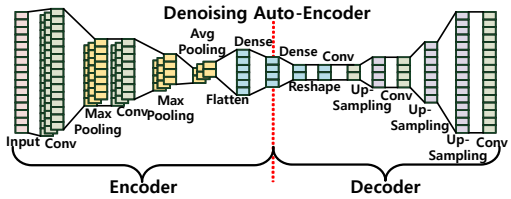


Fig. 16: Architecture of a denoising autoencoder (DAE).

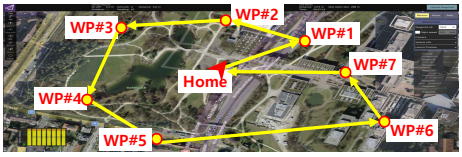


Fig. 17: Trajectory of the planned mission.

configuration, as depicted in Fig. 16. For the gyroscope, we applied 14 as the number of layers; from the layers of the model for the accelerometer, four Conv and Pooling pairs were reduced from the Encoder and Decoder, respectively.

We trained the DAE model for 500 epochs with a step size of 4 and, limited the loss to 0.001. For the learning curve, refer to Fig. 22 in the Appendix. We describe the details of the training dataset in the following section.

C. Resonant Sensor Dataset Construction

In general, training a model requires massive amounts of data to ensure sufficient performance. However, physically acquiring such a large number of sensor signals using actual sensors and a speaker is laborious and time-consuming. In addition, it imposes large experimental overheads because an anechoic chamber and a high-performance speaker that can emit precise acoustic signals are necessary.

To address this, we utilized the resonant sensor models and our testbed, described in §IV, for the construction of the training dataset. For this, we modified the source code of PX4's drone firmware and implemented additional scripts to automatically run simulations and generate the signal pairs. We utilized HITL simulations during the training dataset construction to take into account the effect of sampling jitter. For a detailed analysis of the effect of sampling jitter during training between the SITL and HITL simulations, refer to §C in the Appendix.

For the HITL simulations, we used two target drones, namely 3DR Iris and Solo; these are based on the Pixhawk board. We randomly set seven waypoints at an altitude of 25–100 m as shown in Fig. 17, such that the sum of the distance between the waypoints was approximately 1.33 km. Thus, once a drone took off from the starting point, it was assigned to visit all the waypoints and return to the starting point. The time required for this operation was approximately 6 min, which entailed approximately 90k samples of the sensor values.

During the simulations, we generated and collected both resonant sensor signals and benign ones for each sensor (*i.e.*, a gyroscope and an accelerometer). As each sensor has three axes (*i.e.*, x, y, and z), we collected the signals for each of these sensor axes. We also ensured that the resonant signals were not transferred to the FC to make the drone fly without disturbances; thus, we collected both resonant signals and the corresponding benign signals simultaneously. Then, we used these signal pairs for the evaluation, where the benign signals were used as the ground truth.

TABLE III: Recovery performance summary of UNROCKER with four Amplitudes and three Drone Models for an Accelerometer and a Gyroscope (Total of 72 cases)

Amplitude of Resonance Signals		Accelerometer ($\sigma, m/s^2$)				Gyroscope ($\sigma, rad/s$)			
		20	40	60	80	1	2	3	4
Compromised Signals		14.13	28.28	42.38	56.64	0.707	1.414	2.12	2.83
				↓				↓	
Recovery of 3DR Iris	X-axis	0.270	0.281	0.282	0.296	0.011	0.011	0.013	0.012
	Y-axis	0.086	0.081	0.097	0.083	0.021	0.021	0.020	0.023
	Z-axis	0.871	0.923	0.903	0.930	0.013	0.014	0.015	0.016
Recovery of 3DR Solo	X-axis	0.311	0.308	0.342	0.322	0.254	0.254	0.276	0.289
	Y-axis	0.125	0.118	0.131	0.143	0.046	0.047	0.047	0.054
	Z-axis	0.973	0.971	0.970	1.037	0.031	0.029	0.034	0.038
Recovery of Flight Data	X-axis	3.461	3.473	3.476	4.078	0.079	0.089	0.093	0.100
	Y-axis	2.212	2.213	2.213	2.228	0.050	0.051	0.053	0.056
	Z-axis	2.678	2.712	2.738	2.827	0.043	0.043	0.043	0.044

[1] The red cells indicate that the standard deviation of the errors was destructive to drones. Note that the minimum value to crash both Iris and Solo drones was 35.4 and 1.41 for accelerometers and gyroscopes, respectively.
[2] The amplitudes of the benign sensor signals ranged from -18 to +3 m/s^2 for accelerometers, and from -5 to +5 rad/s for gyroscopes

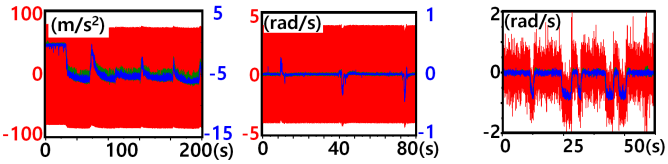
During this step, we set the resonance frequency to 27.1 kHz and 1.83 kHz for the gyroscopes and accelerometers, respectively, as these are the resonance frequencies of the primary IMU in our target board. Additionally, we collected signal pairs by varying their amplitudes. Specifically, we used 0, 1, 2, 3, and 4 (rad/s) for the gyroscopes, and 0, 20, 40, 60, and 80 (m/s^2) for the accelerometers. We also selected the upper bound of the amplitude as described in §V-A.

To construct a sufficiently large dataset, we repeated the aforementioned collection procedure six times. Thus, we collected six sets of 6-min signals for each axis, sensor, and drone. Then, we randomly split the six sets in the ratio of 4:1:1 for training, validation, and testing, respectively. In total, we collected 360 6-min signal pairs, which involve approximately 32.4M pairs of sensor values: 2 drones $\times$ 2 sensors $\times$ 3 axes $\times$ 5 amplitudes $\times$ 6 times $=$ 360.

D. Evaluation of UNROCKER

Next, we evaluated the recovery capability of UNROCKER. To evaluate the success of recovery, we used the standard deviation of the errors between the recovered sensor values and benign ones. This method of utilizing the standard deviation has been widely employed in previous studies [2], [4] for analyzing the degree of errors between benign and compromised sensor values. Similarly, we adopted it to measure the errors between the benign and recovered sensor values, where a smaller standard deviation in the errors represents better performance.

We first trained UNROCKER using the training and validation sets for each axis, sensor, and drone (§VI-C), which resulted in a total of 12 pre-trained models. Subsequently, we used each model to recover the compromised sensor signals for the four different amplitudes in the test set: 1, 2, 3, and 4 (rad/s) for the gyroscopes, and 20, 40, 60, and 80 (m/s^2) for the accelerometers.



(a) Accel. (HITL) (b) Gyro. (HITL) (c) Gyro. (Flight)

Fig. 18: Example of recovery results (red : compromised signal; green : recovered signal; blue : reference signal).

These results are illustrated in Table III. As shown in the table, UNROCKER successfully recovered the compromised sensor signals for all the amplitudes by significantly reducing the errors. Fig. 18a and Fig. 18b depict examples of the recovered signals for the 3DR Iris drone (X-axis), where the amplitude of the compromised signal is the maximum value, such as 80 m/s^2 for the accelerometers and 4 rad/s for the gyroscopes. The standard deviation of the errors was significantly reduced from 56.64 to 0.296 for accelerometers and from 2.83 to 0.013 for gyroscopes. Accordingly, the recovered signal (blue) was almost identical to the benign one (green).

We also measured the SNR to demonstrate UNROCKER's performance in terms of signal quality. The results are shown in Table VIII of Appendix §C. Notably, the SNR indicates an improvement from large negative (from -12.8 to -27.6 dB) values to positive values (at least 0.8 dB and upto 25.6 dB); this means that the recovered original signal overwhelms the noise, demonstrating the recovery capability of UNROCKER.

Evaluation with Actual Sensors (Software Injection). We also evaluated whether UNROCKER can be applied to physical sensors. To emulate the dynamic (moving) state as if the board was installed on a flying drone, we randomly shook the board and injected acoustic signals for 2 min; this involved 30k samples. For the injection, we first obtained the physical sensor signals from the board and then added the resonant sensor signals from the software model in our testbed. This is because if the sensors are actually resonated, it would not be possible to obtain the original values from the resonant sensors for constructing the ground truth. Therefore, we utilized the software sensor models to inject resonant signals and used the benign sensor signals as the ground truth. For the resonant signals, we employed amplitudes of 70 m/s^2 at a frequency of 1.83 kHz for the accelerometers, and 4 rad/s at 27.1 kHz for the gyroscopes.

Then, we measured the standard deviation of the errors between the compromised sensor signals and the recovered ones. Here, we present the results for the X-axis of each sensor on the primary IMU, because it is the most vulnerable to the acoustic injection attack. For the accelerometers, UNROCKER successfully reduced the standard deviation of the errors from 49.61 to 2.00, whereas it decreased that for the gyroscopes from 2.83 to 0.27. Notably, although we reused the model trained for the HITL simulations throughout the recovery experiment, UNROCKER successfully recovered the signals for both sensors.

Evaluation with Actual Sensors (Actual Injection). Although we may not build the ground truth for the dynamic cases, we can infer the ground truth for the stationary cases, as the position of the sensors was fixed and their values would be close to 0 with a subtle bias. We placed the Pixhawk board on a flat table and injected acoustic noise into them using a speaker 10 cm away from the board (110 dB SPL). Then, we recovered the resonant sensor signals. UNROCKER successfully reduced the standard deviation of the errors from 2.361 to 0.935 for the accelerometers, and from 1.469 to 0.025 for the gyroscopes.

Evaluation with Actual Flight Data. We also evaluated whether UNROCKER can recover the sensor data reflecting the dynamics of real flights. For this, we collected sensor logs from a flying drone (3DR Solo) for 90 s (≈22.5k samples). Then, we generated the resonant sensor signals using the software model. The results are summarized in the last row of Table III. UNROCKER successfully reduced the standard deviation of errors for all cases, demonstrating its recovery performance. Notably, this shows that the models trained in the HITL dataset can be applied for real-world data recovery.

Evaluation with Actual Drone with Actual Injection. To evaluate UNROCKER under real-world scenarios, we recovered compromised sensor data for a flying drone under attack. We prepared a drone equipped with a Pixhawk FC board described in §III-B. Recall that the Pixhawk board has two MEMS IMUs (primary: ICM-20689, secondary: ICM-20692), each of which has a different resonance frequency. We configured the drone to use the secondary IMU for its flight to make it continue flying while being attacked. Then, we injected actual acoustic noise using a small Bluetooth speaker attached to the target drone, as in a previous study [2]; we placed the speaker 10 cm from the drone's FC board (110 dB SPL). Then, we collected compromised sensor data from the primary IMU and benign data from the secondary one. Lastly, we recovered the compromised data and evaluated the degree of recovery by considering the data from the secondary one as ground truth.

UNROCKER successfully reduced the standard deviation of errors from 0.71 to 0.044 rad/s for the gyroscopes, as shown in Fig. 18c. Meanwhile, we could not apply the same approach to the accelerometers because the accelerometers in the primary and secondary IMUs have similar resonance frequencies; thus, they resonated simultaneously when being attacked. One may replace one of them with another one with a different resonance frequency; however, it is not trivial to simply replace an IMU with a finished drone product as it requires expert knowledge of configuration and calibration.

E. Towards Real-Time Drone Recovery

Recall that our ultimate goal is to make drones continue flying to planned destinations by recovering compromised sensor values in real-time. However, real-time recovery necessitates addressing several practical challenges. Particularly, as a real-time system, a drone has a tight requirement of computational overhead and latency.For example, the sensor values in PX4 must be updated every 4 ms, as its sampling frequency is 250 Hz. However, the machine-learning (ML)-based approach typically requires more time, *e.g.*, several seconds [58], [59], which is significantly large for PX4. We believe that fully addressing such challenges is beyond the scope of this paper. Instead, we present a proof-of-concept and share our experience towards real-time recovery to encourage future studies.

First, we investigated the timing requirements for real-time recovery in our target system PX4. For this, we forcibly caused a delay when retrieving sensor values during the HITL simulations, and then measured the maximal delay that

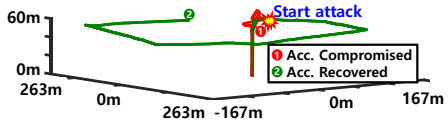
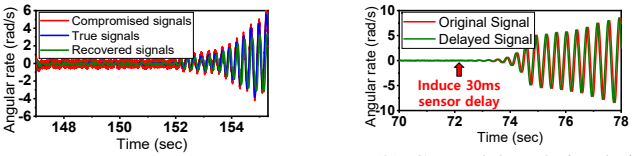


Fig. 19: Trajectory of drones in real-time recovery.



(a) Gyro. recovery closed-loop. (b) Gyro. delayed simulation.

Fig. 20: Gyroscope recovery limitation from latency issue.

our target drone could afford. As a result, we discovered that the recovery for gyroscopes and accelerometers should be completed within 24 ms and 2.8 s, respectively. The accelerometers were more robust than the gyroscopes because gyroscopes are more sensitive to dynamic changes [60]–[62].

Second, we extended UNROCKER to recover compromised sensor values in real-time during the HITL simulations. To minimize the computational overhead and latency for data transmission, we converted the base of UNROCKER from TensorFlow to TensorRT [63]. TensorRT supports a deep-learning inference optimizer for low latency and high throughput while sacrificing performance. In addition, we made UNROCKER return previously recovered sensor values if the current recovery was not finished.

For accelerometers, we confirmed that the drones under attack continued flying and visited all waypoints successfully when the recovered sensor signals were used. Fig. 19 illustrates the trajectory of the drones. Whereas the drones without recovery crashed down to the ground, those with recovery continued flying successfully. For a demo, refer to Video-G on our website [15].

However, we discovered that the recovery for the gyroscopes was limited in our experimental setup. As shown in Fig. 20a, after a short recovery period, gyroscope signals fluctuated and eventually crashed the drones.

We further conducted a causal analysis for the fluctuation of gyroscope signals. As a result, we discovered that a time delay in retrieving sensor values caused the fluctuation. Particularly, the data transmission between the Pixhawk board and server raised a 30-ms delay. To analyze the effect of this latency, we deliberately added a 30-ms delay to benign gyroscope signals during the HITL simulations without injecting resonant signals. As shown in Fig. 20b, the delayed benign gyroscope signals presented similar aspects to the recovered signals (Fig. 20a). While our current experimental setup could not address this latency issue, we believe that drones equipped with edge artificial intelligence (AI) computing boards may achieve real-time recovery. We further discuss this issue in §VII.

VII. DISCUSSION

In this study, we investigated the implications of compromised MEMS IMUs on drones and developed a novel sensor recovery strategy based on a noise reduction technique. This section describes the limitations of our study, as well as the remaining insights to further improve the performance of the proposed drone recovery.

Difficulties in Intentionally Targeting In-band Range. We discovered sampling jitter to be a critical factor in an acoustic injection attack by spreading out-band resonance signals into the in-band range of the drone's control logic (§V-C). Meanwhile, one may wonder if an adversary can directly target the in-band range without considering the sampling jitter. To address this concern, we discuss that 1) such a case is extremely challenging for a flying drone, and 2) even if it is feasible, it can be easily mitigated.

First, an adversary may intentionally manipulate the sensor's response signal to be within the in-band range as discussed in a previous study [4]. The study presented that an adversary can manipulate the sensor's response signal at the desired frequency. However, this is extremely challenging for flying drones owing to the environmental inconsistencies between stationary and flying drones because the driving frequency of the flying drones' gyroscope significantly fluctuates. Note that the frequency of the gyroscope's response signal is determined by its driving frequency; the difference between the frequency of the injected acoustic wave and the driving frequency becomes the frequency of the response signal. This driving frequency widely fluctuates depending on manufacturing errors, temperature, and aging [64]. Therefore, although the adversary conducts an attack using in-band signals identified for a stationary drone, the response signal would not likely appear within the in-band range, considering the narrow in-band range (0–5 Hz). Notably, this issue is also widely known by developers, and thus, they make MEMS gyroscopes have a wide flat sensing region (several hundred to thousands of Hz) to cover the fluctuating driving frequency for normal operation [64].

Meanwhile, MEMS accelerometers are much more sensitive to sampling jitter than gyroscopes as shown in §V-C. This means that if an adversary targets the in-band signals for the accelerometers, the actual signals sampled on flying drones will be spread more by the sampling jitter. Consequently, for accelerometers, an adversary cannot fit the resonance signals into the narrow 5-Hz in-band range. Similarly, Tu *et al.* [3] also demonstrated that only a DoS was feasible while failing on spoofing (without clear reasoning).

Based on these observations, we concluded that directly targeting the in-band range without considering the sampling jitter is extremely challenging.

Evasion of UnRocker. As discussed above, an adversary is most likely not to be able to carefully control the resonant signal to reside within the in-band range of the target drone. Nevertheless, we evaluated the resilience of UNROCKER against potential in-band resonant signals. More specifically, we conducted experiments by shifting the attack frequency to produce compromised sensor signals at different frequencies. Additionally, we modified the amplitude by continuously increasing it. For the frequency, we shifted the resonant signal by -100 and +100 Hz because the resonating frequency of sensors is often spread out over a $\pm$100-Hz range [65]. We generated a separate dataset containing 54 additional test cases in the same manner described in §VI-C; note that these are different from the 72 test cases shown in Table III. Consequently, UNROCKER exhibited reliable performance for all experiments listed in Table IV, showing that the standard deviation of errors resides within the range of benign signals.

TABLE IV: Recovery performance summary of UNROCKER with three attack variations and three Drone Models for Accelerometer and Gyroscope (Total of 54 cases)

Attack variations for Evasion of UNROCKER		Accelerometer $(\sigma, m/s^2)$			Gyroscope $(\sigma, rad/s)$		
		+100Hz	-100Hz	Amp.†	+100Hz	-100Hz	Amp.†
Compromised Signals		56.64	56.64	41.59	2.83	2.83	2.08
			↓			↓	
Recovery of 3DR Iris	X-axis	1.023	1.570	0.357	0.031	0.057	0.013
	Y-axis	0.276	0.289	0.199	0.118	0.097	0.028
	Z-axis	1.097	1.112	0.940	0.038	0.154	0.021
Recovery of 3DR Solo	X-axis	0.790	1.550	1.076	0.333	0.372	0.390
	Y-axis	0.377	0.441	0.322	0.096	0.105	0.059
	Z-axis	1.252	1.241	1.082	0.075	0.106	0.037
Recovery of Flight Data	X-axis	3.555	5.599	2.951	0.167	0.164	0.089
	Y-axis	2.223	2.225	2.263	0.094	0.104	0.051
	Z-axis	3.118	3.004	2.734	0.070	0.083	0.043

† Continuously varying amplitudes of attack with slow sinusoidal pattern (50 seconds period).

TABLE V: Comparison to the heuristic filtering approaches

Filter Types		Accelerometer $(\sigma, m/s^2)$				Gyroscope $(\sigma, rad/s)$			
		20	40	60	80	1	2	3	4
LPF	X	1.67 (x6.2)	2.91 (x10)	4.30 (x15)	5.94 (x20)	0.09 (x7.8)	0.17 (x15)	0.25 (x19)	0.34 (x28)
	Y	1.64 (x19)	2.90 (x36)	4.28 (x44)	5.93 (x72)	0.10 (x4.6)	0.17 (x8.2)	0.25 (x12)	0.33 (x15)
	Z	2.06 (x2.4)	3.16 (x3.4)	4.46 (x4.9)	6.09 (x6.5)	0.09 (x6.7)	0.17 (x12)	0.25 (x16)	0.34 (x21)
Sav-Gol Filter	X	1.06 (x3.9)	1.89 (x6.7)	2.76 (x9.8)	3.85 (x13)	0.08 (x7.5)	0.17 (x15)	0.25 (x19)	0.33 (x28)
	Y	1.04 (x12)	1.87 (x23)	2.76 (x28)	3.85 (x46)	0.08 (x4.0)	0.17 (x7.8)	0.25 (x12)	0.34 (x15)
	Z	1.24 (x1.4)	2.00 (x2.2)	2.84 (x3.1)	3.91 (x4.2)	0.08 (x6.4)	0.17 (x12)	0.25 (x16)	0.33 (x21)
Wie-ner Filter	X	1.57 (x5.8)	1.77 (x6.3)	2.03 (x7.2)	2.39 (x8.1)	0.06 (x5.1)	0.07 (x6.3)	0.09 (x6.6)	0.11 (x8.8)
	Y	0.54 (x6.3)	0.96 (x12)	1.41 (x15)	1.88 (x23)	0.07 (x3.3)	0.08 (x3.8)	0.09 (x4.7)	0.11 (x4.9)
	Z	1.17 (x1.3)	1.43 (x1.5)	1.74 (x1.9)	2.16 (x2.3)	0.13 (x9.6)	0.13 (x9.5)	0.14 (x9.5)	0.15 (x9.6)

Comparison with Heuristic Filters. To observe the validity of the data-driven ML approach, we evaluated the performance of UNROCKER with three conventional heuristic filters: a simple LPF, a Sav-Gol smoothing filter [45], and an *industry standard* Wiener filter [46]. Notably, the denoising performance of UNROCKER was higher than that of the LPF by up to 72 times, that of the Sav-Gol filter by up to 46 times, and that of the Wiener filter by up to 23 times (refer to Table V). Additionally, we implemented the heuristic filters in our acoustic injection testbed (§IV) and checked if they can indeed mitigate the attack. However, all filters failed to prevent the drones from crashing (refer to Video-F in our website [15]).

Toward More Practical Real-time Recovery. As discussed in §VI-E, real-time recovery necessitates addressing several practical challenges, particularly the computational overhead and latency issues. However, addressing such issues entails inevitable degradation in the recovery performance, thus provoking mundane discussions regarding the trade-off between recovery performance and speed. We, therefore, encourage further ground-breaking research that enables sensor recovery with negligible overheads.

Furthermore, communication latency might be addressed using an on-board inference chip, as it significantly reduces communication delays. Recently, many edge computing devices have been introduced. For example, NVIDIA's Jetson has been adopted as a lightweight single-board computer for IoT devices including drones [66]. Implementing real-time recovery on such systems would be a promising avenue for future studies, and we encourage further research on this topic. With such a trend, we believe that real-time recovery can be accomplished in the not-too-distant future.

VIII. RELATED WORKS

Security of Drones. With the growth of the drone industry, security threats pertaining to drones have been studied extensively. Such studies can largely be categorized into three types [67]: those based on communication channels, GNSS, and sensors. Among these, the initial studies on drone vulnerabilities focused on communication channel issues [68]–[70]. Subsequently, advanced analyses and mitigation strategies have been proposed against these communication channels along with detailed software analyses [71]–[76]. As drones are equipped with a variety of sensors, the security of sensors constitutes an additional attack surface for drones [77]. Despite these concerns, the future of unmanned vehicles is expected to exhibit a growing dependence on sensor technology. Across a range of aspects, the vulnerability of sensors used in drones has been extensively studied, such as for GNSS receivers [78], IMU sensors [2]–[4], and light detection and ranging (LiDAR) sensors [79].

Comparison to State-of-the-art Denoising. Among various applications of signal denoising, *speech enhancement* is one of the most active applications [80]. Past approaches in speech denoising adopted conventional spectral domain filtering (spectral subtraction and Wiener filter). However, owing to the recent development of DNNs, several ML-based approaches have been proposed [47]. As a case study, we utilized a recent DNN model called the deep-feature-loss (DFL) network [81] instead of DAE. It yielded better results than ours on the regular dataset (0.171 rad/s for a Solo X-axis gyroscope, which is superior to 0.289 rad/s from the original UNROCKER as presented in Table III). However, the performance of DFL significantly deteriorated for the unseen dataset (when the frequency was changed, and real sensor data were evaluated); it produced a 23–193% larger deviation than UNROCKER.

Recently, the state-of-the-art approaches in speech denoising have presented promising results by adopting the generative adversarial network (GAN). However, directly applying these approaches to mitigate acoustic injection attacks is not trivial as they are optimized for speech-centric data. As a case study, we evaluated the conformer-based metric-GAN (CMGAN) [82], which is the leading model in "Speech Enhancement on DEMAND" [80]. When we trained the CMGAN with our IMU sensor data, the training loss remained at 0.8, whereas UNROCKER mostly converged to a training loss of less than 0.0001. This result shows that the model cannot be directly applied to sensor signal recovery, and it requires additional fine-tuning or other engineering efforts. We encourage further studies on applying promising state-of-the-art denoising techniques to mitigate acoustic injection attacks.

IX. Conclusion

While MEMS IMU sensors are essential in drones, they are, by design, vulnerable to acoustic injection attacks, which can cause the drones to crash. Several mitigation techniques have been proposed thus far; however, they have not been able to ensure that the compromised drones continue flying to their planned destinations. In this study, we proposed a novel, practical mitigation that recovers the benign sensor values from compromised ones for MEMS gyroscopes and accelerometers. We first constructed an acoustic injection testbed and delved into the implications of compromised sensor values on drones using the testbed. Consequently, we discovered a critical factor, sampling jitter, which crashes drones during attacks by significantly increasing the implications of the attacks. Considering the resonant sensor signals affected by the sampling jitter as noise, we developed a sensor recovery technique that leverages a noise reduction technique, namely a DAE. We implemented a prototype recovery system, UnRocker, and demonstrated its capability through rigorous experiments. The findings of this study demonstrate that empirical implication analysis of the compromised sensor signals on drones is essential for root cause analysis of the attack. Thus, our released source code and experimental data can be employed in future studies to perform detailed empirical investigations.

Acknowledgments

This work was supported by Air Force Defense Research Sciences Program funded by Air Force Research Laboratory (AFRL) (Title: Cyber Physical Analysis of System Software Survivability by Stimulating Sensors on Drones), and Institute for Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2022-0-01202, Regional strategic industry convergence security core talent training business).

References

[1] L. SCHROTH, "THE DRONE MARKET SIZE 2020." [Online]. Available: https://droneii.com

[2] 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)*, 2015, pp. 881–896.

[3] Y. Tu, Z. Lin, I. Lee, and X. Hei, "Injected and delivered: Fabricating implicit control over actuation systems by spoofing inertial sensors," in *27th USENIX Security Symposium*, 2018, pp. 1545–1562.

[4] 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.

[5] Genasys, "LRAD - Long Range Acoustic Devices," 2022. [Online]. Available: https://genasys.com/lrad-products/

[6] B. Zohuri, "Directed energy weapons," in *Directed Energy Weapons*. Springer, 2016, pp. 1–26.

[7] Prime Consulting and Technologies, "Anti drone solutions," 2022. [Online]. Available: https://anti-drone.eu/products/long-range-acoustic-devices.html

[8] U.S Department of Defense, "MIL-STD-810H, DEPARTMENT OF DEFENSE TEST METHOD STANDARD: Environmental engineering considerations and laboratory tests," 2019.

[9] H. Choi, W.-C. Lee, Y. Aafer, F. Fei, Z. Tu, X. Zhang, D. Xu, and X. Deng, "Detecting attacks against robotic vehicles: A control invariant approach," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 10 2018, pp. 801–816.

[10] R. Quinonez, J. Giraldo, L. Salazar, E. Bauman, A. Cardenas, and Z. Lin, "SAVIOR: Securing autonomous vehicles with robust physical invariants," in *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Aug. 2020, pp. 895–912.

[11] H. Choi, S. Kate, Y. Aafer, X. Zhang, and D. Xu, "Software-based realtime recovery from sensor attacks on robotic vehicles," in *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, 2020, pp. 349–364.

[12] Z. Tu, F. Fei, M. Eagon, D. Xu, and X. Deng, "Flight recovery of MAVs with compromised IMU," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 3638–3644.

[13] C. Patel, A. Jones, J. Davis, P. McCluskey, and D. Lemus, "Temperature effects on the performance and reliability of MEMS gyroscope sensors," in *International Electronic Packaging Technical Conference and Exhibition*, vol. 43598, 2009, pp. 507–512.

[14] D. Gis, N. Büscher, and C. Haubelt, "Investigation of timing behavior and jitter in a smart inertial sensor debugging architecture," *Sensors*, vol. 21, no. 14, p. 4675, 2021.

[15] "UnRocker," 2022. [Online]. Available: https://sites.google.com/view/unrocker/

[16] "PX4 Open-source," 2021. [Online]. Available: https://dev.px4.io/v1.9.0/en/

[17] "ArduCopter," 2021. [Online]. Available: https://ardupilot.org/copter/

[18] Q. Quan, *Sensor Calibration and Measurement Model*. Springer, 06 2017, pp. 147–172.

[19] R. O'Reilly, A. Khenkin, and K. Harney, "Sonic nirvana: Using MEMS accelerometers as acoustic pickups in musical instruments," *Analog Dialogue*, vol. 43, no. 02, pp. 1–4, 2009.

[20] C. Acar and A. Shkel, "Inherently robust micromachined gyroscopes with 2-DoF sense-mode oscillator," *Journal of Microelectromechanical Systems*, vol. 15, no. 2, pp. 380–387, 2006.

[21] A. R. Schofield, A. A. Trusov, and A. M. Shkel, "Effects of operational frequency scaling in multi-degree of freedom mems gyroscopes," *IEEE Sensors Journal*, vol. 8, no. 10, pp. 1672–1680, 2008.

[22] C. Acar, A. R. Schofield, A. A. Trusov, L. E. Costlow, and A. M. Shkel, "Environmentally robust MEMS vibratory gyroscopes for automotive applications," *IEEE Sensors Journal*, vol. 9, no. 12, pp. 1895–1906, 2009.

[23] B. Gallacher, J. Burdess, and K. Harish, "A control scheme for a MEMS electrostatic resonant gyroscope excited using combined parametric excitation and harmonic forcing," *Journal of Micromechanics and Microengineering*, vol. 16, no. 2, p. 320, 2006.

[24] A. D. Wyner and S. Shamai, "Introduction to 'Communication in the presence of noise' by CE Shannon," *Proceedings pf IEEE*, vol. 86, no. 2, pp. 442–446, 1998.

[25] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[26] M. Bacic, "On hardware-in-the-loop simulation," in *Proceedings of the 44th IEEE Conference on Decision and Control*. IEEE, 2005, pp. 3194–3198.

[27] D. Jung and P. Tsiotras, "Modeling and hardware-in-the-loop simulation for a small unmanned aerial vehicle," in *AIAA Infotech@ Aerospace 2007 Conference and Exhibit*, 2007, p. 2768.

[28] H. K. Fathy, Z. S. Filipi, J. Hagena, and J. L. Stein, "Review of hardware-in-the-loop simulation and its prospects in the automotive area," in *Modeling and simulation for military applications*, vol. 6228. International Society for Optics and Photonics, 2006, p. 62280E.

[29] D. Benowitz, "Drone Analysist." [Online]. Available: https://droneanalyst.com/2021/05/30/rise-of-open-source-drones

[30] S. Khazaaleh, G. Korres, M. Eid, M. Rasras, and M. F. Daqaq, "Vulnerability of MEMS gyroscopes to targeted acoustic attacks," *IEEE Access*, pp. 89 534–89 543, 2019.

[31] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, 2004, pp. 2149–2154.

[32] M. Zhang, H. Qin, M. Lan, J. Lin, S. Wang, K. Liu, F. Lin, and B. M. Chen, "A high fidelity simulator for a quadrotor UAV using ROS and

gazebo," in *IECON 2015-41st Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2015, pp. 002 846–002 851.

[33] D. S. Kaputa and K. J. Owens, "Quadrotor drone system identification via model-based design and in-flight sine wave injections," in *AIAA Scitech 2020 Forum*, 2020, p. 1238.

[34] S. H. Cho, S. Bhandari, F. C. Sanders, M. B. Tischler, and K. Cheung, "System identification and controller optimization of coaxial quadrotor UAV in hover," in *AIAA Scitech 2019 Forum*, 2019, p. 1075.

[35] H. Gu, W. Su, B. Zhao, H. Zhou, and X. Liu, "A design methodology of digital control system for MEMS gyroscope based on multi-objective parameter optimization," *Micromachines*, vol. 11, no. 1, p. 75, 2020.

[36] P. C. Schulze, J. Miller, D. H. Klyde, C. D. Regan, and N. Alexandrov, "System identification of a small UAS in support of handling qualities evaluations," in *AIAA Scitech 2019 Forum*, 2019, p. 0826.

[37] W. Wei, M. B. Tischler, and K. Cohen, "System identification and controller optimization of a quadrotor unmanned aerial vehicle in hover," *Journal of the American Helicopter Society*, vol. 62, no. 4, pp. 1–9, 2017.

[38] A. Chovancová, T. Fico, L. Chovanec, and P. Hubinsk, "Mathematical modelling and parameter identification of quadrotor (a survey)," *Procedia Engineering*, vol. 96, pp. 172–181, 2014.

[39] L. R. Rabiner and B. Gold, "Theory and application of digital signal processing," *Englewood Cliffs: Prentice-Hall*, 1975.

[40] A. Kiyono, M. Kim, K. Ichige, and H. Arai, "Jitter effect on digital downconversion receiver with undersampling scheme," in *The 2004 47th Midwest Symposium on Circuits and Systems, 2004. MWSCAS'04.*, vol. 2. IEEE, 2004, pp. II–677.

[41] M. Kim, A. Kiyono, K. Ichige, and H. Arai, "Experimental study of jitter effect on digital downconversion receiver with undersampling scheme," *IEICE Transactions on Information and Systems*, vol. 88, no. 7, pp. 1430–1436, 2005.

[42] F. Proctor and W. Shackleford, "Real-time operating system timing jitter and its impact on motor control," in *Sensors and Controls for Intelligent Manufacturing II*, vol. 4563. SPIE, 12 2001.

[43] B. Ip, "Performance Analysis of VxWorks and RTLinux," in *Languages of Embedded Systems Department of Computer Science*, 2001.

[44] A. Barbalace, A. Luchetta, G. Manduchi, M. Moro, A. Soppelsa, and C. Taliercio, "Performance comparison of VxWorks, Linux, RTAI and Xenomai in a hard real-time application," *Nuclear Science, IEEE Transactions on*, vol. 55, pp. 435–439, 03 2008.

[45] R. W. Schafer, "What is a savitzky-golay filter?[lecture notes]," *IEEE Signal Processing Magazine*, vol. 28, no. 4, pp. 111–117, 2011.

[46] J. Le Roux and E. Vincent, "Consistent wiener filtering for audio source separation," *IEEE Signal Processing Letters*, vol. 20, no. 3, pp. 217–220, 2012.

[47] R. Xu, R. Wu, Y. Ishiwaka, C. Vondrick, and C. Zheng, "Listening to sounds of silence for speech denoising," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9633–9648, 2020.

[48] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th International Conference on Machine Learning*, 2008, pp. 1096–1103.

[49] X. Lu, Y. Tsao, S. Matsuda, and C. Hori, "Speech enhancement based on deep denoising autoencoder." in *Interspeech*, vol. 2013, 2013, pp. 436–440.

[50] S. S. Roy, S. I. Hossain, M. Akhand, and K. Murase, "A robust system for noisy image classification combining denoising autoencoder and convolutional neural network," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 1, pp. 224–235, 2018.

[51] F.-F. Xue, J. Peng, R. Wang, Q. Zhang, and W.-S. Zheng, "Improving robustness of medical image diagnosis with denoising convolutional neural networks," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2019, pp. 846–854.

[52] J. Liu, L. Xu, Y. Xie, T. Ma, J. Wang, Z. Tang, W. Gui, H. Yin, and H. Jahanshahi, "Toward robust fault identification of complex industrial processes using stacked sparse-denoising autoencoder with softmax classifier," *IEEE Transactions on Cybernetics*, 2021.

[53] Y. Xiong and R. Zuo, "Robust feature extraction for geochemical anomaly recognition using a stacked convolutional denoising autoencoder," *Mathematical Geosciences*, vol. 54, no. 3, pp. 623–644, 2022.

[54] Z. Qu, W. Wang, C. Hou, and C. Hou, "Radar signal intra-pulse modulation recognition based on convolutional denoising autoencoder and deep convolutional neural network," *IEEE Access*, vol. 7, pp. 112 339–112 347, 2019.

[55] A. Rosebrock, "Denoising autoencoders with Keras, TensorFlow, and Deep Learning," Feb, 2020. [Online]. Available: https://pyimagesearch.com

[56] P. Xiong, H. Wang, M. Liu, and X. Liu, "Denoising autoencoder for eletrocardiogram signal enhancement," *Journal of Medical Imaging and Health Informatics*, vol. 5, no. 8, pp. 1804–1810, 2015.

[57] P. G. Shivakumar and P. G. Georgiou, "Perception optimized deep denoising autoencoders for speech enhancement." in *Interspeech*, 2016, pp. 3743–3747.

[58] V. S. Marco, B. Taylor, Z. Wang, and Y. Elkhatib, "Optimizing deep learning inference on embedded systems through adaptive model selection," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 19, no. 1, pp. 1–28, 2020.

[59] S. Shi, Q. Wang, P. Xu, and X. Chu, "Benchmarking state-of-the-art deep learning software tools," in *2016 7th International Conference on Cloud Computing and Big Data (CCBD)*. IEEE, 2016, pp. 99–104.

[60] M. Ghanbari and M. J. Yazdanpanah, "Delay compensation of tilt sensors based on MEMS accelerometer using data fusion technique," *IEEE Sensors Journal*, vol. 15, no. 3, pp. 1959–1966, 2014.

[61] J. Li, S. Xu, Y. Liu, X. Liu, Z. Li, and F. Zhang, "Real-time indoor navigation of uav based on visual delay compensation," in *2019 IEEE International Conference on Mechatronics and Automation (ICMA)*. IEEE, 2019, pp. 2451–2456.

[62] X. Xinjilefu, S. Feng, and C. G. Atkeson, "A distributed MEMS gyro network for joint velocity estimation," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1879–1884.

[63] "NVIDIA TensorRT," 2021. [Online]. Available: https://developer.nvidia.com/tensorrt

[64] C. Acar and A. M. Shkel, "Nonresonant micromachined gyroscopes with structural mode-decoupling," *IEEE Sensors Journal*, vol. 3, no. 4, pp. 497–506, 2003.

[65] ——, "An approach for increasing drive-mode bandwidth of MEMS vibratory gyroscopes," *Journal of Microelectromechanical Systems*, vol. 14, no. 3, pp. 520–528, 2005.

[66] "Jetson platforms," 2022. [Online]. Available: https://developer.nvidia.com/embedded/community/quick-start-platforms

[67] B. Nassi, R. Bitton, R. Masuoka, A. Shabtai, and Y. Elovici, "SoK: Security and privacy in the age of commercial drones," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 1434–1451.

[68] J. Gabrielsson, J. Bugeja, and B. Vogel, "Hacking a commercial drone with open-source software: Exploring data privacy violations," in *2021 10th Mediterranean Conference on Embedded Computing (MECO)*. IEEE, 2021, pp. 1–5.

[69] J. Valente and A. A. Cardenas, "Understanding security threats in consumer drones through the lens of the discovery quadcopter family," in *Proceedings of the 2017 Workshop on Internet of Things Security and Privacy*, 11 2017, pp. 31–36.

[70] V. Dey, V. Pudi, A. Chattopadhyay, and Y. Elovici, "Security vulnerabilities of unmanned aerial vehicles and countermeasures: An experimental study," in *2018 31st International Conference on VLSI Design and 2018 17th International Conference on Embedded Systems (VLSID)*. IEEE, 2018, pp. 398–403.

[71] R. Ivanov, M. Pajic, and I. Lee, "Attack-resilient sensor fusion," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 02 2014, pp. 1–6.

[72] I. Radoslav, P. Miroslav, and L. Insup, "Attack-resilient sensor fusion for safety-critical cyber-physical systems," *ACM Transactions on Embedded Computing Systems*, vol. 15, pp. 1–24, 02 2016.

[73] J. Park, R. Ivanov, J. Weimer, M. Pajic, and I. Lee, "Sensor attack detection in the presence of transient faults," in *Proceedings of the ACM/IEEE Sixth International Conference on Cyber-Physical Systems*, 2015, pp. 1–10.

[74] L. R. Garcia Carrillo and K. G. Vamvoudakis, "Deep-learning tracking for autonomous flying systems under adversarial inputs," *IEEE Transactions on Aerospace and Electronic Systems*, vol. PP, pp. 1–1, 07 2019.

[75] T. Kim, C. Kim, J. Rhee, F. Fei, Z. Tu, G. Walkup, X. Zhang, X. Deng, and D. Xu, "RVFuzzer: Finding input validation bugs in robotic vehicles through control-guided testing," in *USENIX Security Symposium*, 2019, pp. 425–442.

[76] T. Kim, C. H. Kim, A. Ozen, F. Fei, Z. Tu, X. Zhang, X. Deng, D. J. Tian, and D. Xu, "From control model to program: Investigating robotic aerial vehicle accidents with MAYDAY," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 913–930.

[77] C. Yan, H. Shin, C. Bolton, W. Xu, Y. Kim, and K. Fu, "SoK: A minimalist approach to formalizing analog sensor security," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 05 2020, pp. 233–248.

[78] 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, pp. 1–26, 04 2019.

[79] H. Shin, D. Kim, Y. Kwon, and Y. Kim, "Illusion and dazzle: Adversarial optical channel exploits against LiDARs for automotive applications," in *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 08 2017, pp. 445–467.

[80] "Speech Enhancement on DEMAND," 2022. [Online]. Available: https://paperswithcode.com/task/speech-enhancement

[81] F. G. Germain, Q. Chen, and V. Koltun, "Speech Denoising with Deep Feature Losses," in *Proc. Interspeech 2019*, 2019, pp. 2723–2727.

[82] S. Abdulatif, R. Cao, and B. Yang, "CMGAN: Conformer-based metric-GAN for monaural speech enhancement," *arXiv preprint arXiv:2209.11112*, 2022.

APPENDIX

A. Causal Analysis Results of Sampling Jitter

In §V-E, we described three potential causes of sampling jitter that were discovered during our investigations: scheduling issues, operation interval mismatches, and imprecise clocks. Here, we present the detailed results.

First, we measured the scheduling jitter in PX4's *NuttX* RTOS. The standard deviation of the scheduling jitter was measured at 124.6 μs. In the case of a resonance signal with a frequency of several kHz, this timing jitter of 124 μs is comparable to the period corresponding with the acoustic frequency.

Furthermore, we measured the time interval to retrieve the IMU sensor data at the FC by directly monitoring the communication channel between the IMU and FC using an oscilloscope. In general, the IMU sensor data (*e.g.*, gyroscope data) are transmitted based on two interrupt signals. When the IMU samples sensor data, it notifies the FC using an interrupt signal (a voltage change from "low" to "high"). When the FC reads the sensor data, it also notifies the IMU through an interrupt signal (a voltage change from "high" to "low"), Therefore, these interrupt signals fluctuate, indicating inaccurate operation timing. The interval fluctuated with a distribution of approximately 100 μs. For details of this measurement, please refer to Video-E on our website [15].

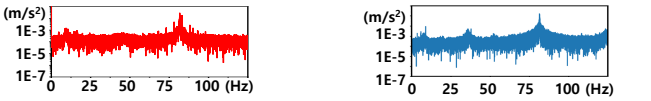
In addition, the mismatches between the sampling and operation intervals can affect the resonant signal in terms of sampling jitter. Notably, we discovered several mismatches caused by the specifications of the sensors, as listed in Table VI.

Lastly, to further investigate the sampling jitter sources, we attempted to recreate the measured resonance spectrum by introducing additional factors into our computed spectrum. We

TABLE VI: Sampling frequency of sensor drivers

Sensor Type		Gyro.	Acc.
ArduCopter (400Hz)	BMI055	2,000	2,000
	BMI160	1,600	1,600
	InvenSense	1,000	1,000
	L3G4200D	800	800
	LSM9DS0	760	1,000
	LSM9DS1	952	952
	Revo	1,000	1,000
PX4 Quadcopter (250Hz)	ADIS16477	250	250
	ADIS16497	1,000	1,000
	BMA180	250	250
	BMI055	1,000	1,000
	BMU160	800	800
	FXAS21002C	800	·
	FXO8701CQ	·	800
	ICM20948	1,000	1,000
	L3GD20	760	·
	LSM303D	·	800
	MPU6000	1,000	1,000
	MPU9250	1,000	1,000

measured the resonance signal of the ICM-20689 accelerometer during the injection of an acoustic signal at 1.83 kHz. The resonance frequency F_a of the ICM-20689 accelerometer was characterized by the emitted acoustic frequency (*i.e.*, 1.83 kHz). Thereafter, we followed Nyquist's sampling theorem to analyze the logged frequency.



(a) PX4 accel. (measured) (b) PX4 accel. (computed)

Fig. 21: Analysis of measured resonance and reproduction of measured signal considering sources of sampling jitters.

The accelerometer was analyzed considering under-sampling issues. As the ICM-20689 MEMS employs the MPU6000 series device driver, it operates at 1 kHz. The resonance signal at 1.83 kHz that passed through the device driver was under-sampled to a signal within 500 Hz, which was half of the 1 kHz sampling frequency. As calculated, the frequency was 170 Hz: $|1830Hz - 1000Hz \times 2| = 170Hz$. The signal output of 170 Hz from the driver was further under-sampled according to the sampling frequency of the microcontroller unit (MCU), which was 250 Hz. Given that half of 250 Hz is 125 Hz, the aliased frequency was 80 Hz: $|170Hz - 250Hz| = 80Hz$.

However, in addition to the 80-Hz peak, at least two peaks were observed in the accelerometer's resonance, as shown in Fig. 21. Individually, under-sampling cannot explain the two redundant peaks around 10 and 50 Hz. Accordingly, we considered two additional elements from the hardware operation to recalculate these components: scheduling jitter and clock-source precision error. We have already discussed scheduling jitter. The MCU and ADC of the MEMS gyroscope use an independent clock source (refer to Fig. 4). Generally, independent clocks are not synchronized, resulting in mismatched sampling because of precision errors. The spectrum shown in Fig. 21b was obtained by assuming a 5 % mismatch from the clock source precision error. Considering the unpredictability of the sampling factor, the results depicted in Fig. 21b are similar to the actual measurement results (Fig. 21a).

B. Implementation of UnRocker

The modified modules for overall system design, including the modified code lines and freshly developed auto-test code

TABLE VII: Overall implemented components and lines of code

Component	Module	Lines of Code
Drone Firmware (C++)	Sensor Drivers & ROM Filesystem	330
	Messaging Modules	202
	Drone State Logger	37
	Attitude and Position Control	103
	Simulator Module	183
Automation (Python)	Simulator	492
DAE (Python)	DAE Model	55
Dataset Construction (Python)	Dataset Generator	458
Model Training (Python)	Training Script & Data Loader	566
	Validation Script & Data Loader	450
	Test Script & Data Loader	633
Online Inference (Python)	Inference Program	208
	Communication Program	405
	Gyroscope Recovery Test	514
	Accelerometer Recovery Test	438
	Total	4,964

lines, are summarized in Table VII.

C. More Details on DAE of UNROCKER

Training Results. The training curve of the DAE in the recovery system of UNROCKER is plotted in Fig. 22. The red line represents the training loss, whereas the blue line indicates the validation loss. Both the training and validation losses converged after 300 epochs when training to 500 epochs, thereby implying a stable low loss. Therefore, the training procedure can be considered complete.

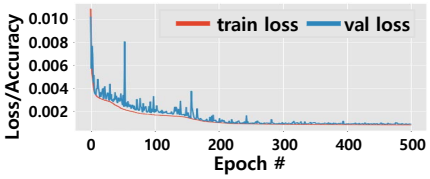


Fig. 22: Training curve.

The Implication of Sampling Jitter in Training. Sampling jitter is a vital factor in acoustic injection attacks, as described in §V. Thus, we conducted a simple experiment to investigate the implications of sampling jitter when training the model. Particularly, we reviewed the necessity of including a sampling jitter when generating the signals for the training dataset. To this end, we constructed two datasets for both the SITL and HITL simulations: one from SITL (no sampling jitter) and another from HITL (large sampling jitter).

Thereafter, we conducted two tests: 1) we trained the model using the training set of the SITL dataset, and then evaluated it with the validation set of the HITL dataset; and 2) we conducted the same experiment except with the SITL dataset, *i.e.*, using the training and validation set of the HITL dataset. To enhance the inference performance for harsh operations, a sampling jitter must be added during the dataset generation phase.

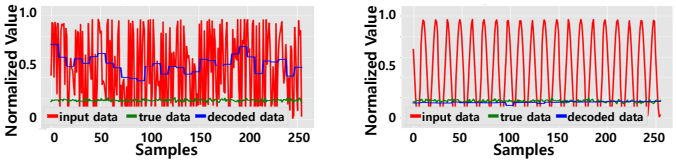
The outcome is depicted in Fig. 23. The model trained using the SITL dataset without sampling jitter failed to recover the sensor values (Fig. 23a), exhibiting significant fluctuations. By contrast, the model trained with the HITL dataset containing sampling jitter could successfully recover

TABLE VIII: Recovery performance summary of UNROCKER with four Amplitudes and three Drone Models for Accelerometer and Gyroscope (Total of 72 cases, SNR representation)

Amplitude of Resonance Signals		Accelerometer (m/s^2)				Gyroscope (rad/s)			
		20	40	60	80	1	2	3	4
Average SNR (dB) of Compromised Signals		-12.8	-18.8	-22.3	-24.8	-15.6	-21.6	-25.1	-27.6
				↓			↓		
Recovery of 3DR Iris (dB)	X-axis	25.6	25.3	25.2	24.8	13.7	13.7	12.2	12.9
	Y-axis	12.4	12.9	11.3	12.7	10.2	10.2	10.6	9.4
	Z-axis	19.9	19.4	19.6	19.4	20.1	19.4	18.8	18.3
Recovery of 3DR Solo (dB)	X-axis	24.5	24.5	23.6	24.2	8.7	8.7	8.0	7.6
	Y-axis	9.2	9.7	8.8	8.0	6.9	6.7	6.7	5.5
	Z-axis	19.0	19.0	19.0	18.4	13.0	13.6	12.2	11.3
Recovery of Flight Data (dB)	X-axis	2.2	2.2	2.2	0.8	5.3	4.3	3.9	3.3
	Y-axis	2.9	2.9	2.9	2.9	8.2	8.1	7.7	7.3
	Z-axis	12.5	12.4	12.3	12.0	1.3	1.3	1.3	1.1

[1] The red cells indicate that the level of noise was destructive to drones.

the sensor values (Fig. 23b). This result indicates that sampling jitter must be included in the signals for training.



(a) Trained w/o sampling jitter. (b) Trained w/ sampling jitter.

Fig. 23: Effect of sampling jitter in training.

Additionally, we injected an acoustic wave in the physical sensors and compared the recovery performance of two models: one trained with a dataset containing the sampling jitter, and the other with a dataset not containing it. Consequently, we discovered that the model trained with the sampling jitter performed 2.24 times better than the other one, which demonstrates the significance of sampling jitter in training.

Representation of Signal Quality. The performance of UNROCKER was evaluated on signal distributions, as shown in Table III. The signal distribution becomes small enough for threatening distributions. In addition to this, we evaluated UNROCKER using SNR measurements. Table VIII summarizes the results, which show significant improvements.

Online Inference. Online inference time results are shown in Fig. 24. The inference time is 4–8 ms on a laptop, with an average of 12 ms on a Jetson Nano, and 9 ms on a Jetson TX2. Although this result does not satisfy the 250Hz (4ms) requirement, future AI chips are expected to minimize the inference performance and I/O delay when *on-boarded*. We also measured the power consumption of a Jetson Nano; it consumed 1.6W for UNROCKER. Thus, the power consumption in the edge inference board is not crucial.

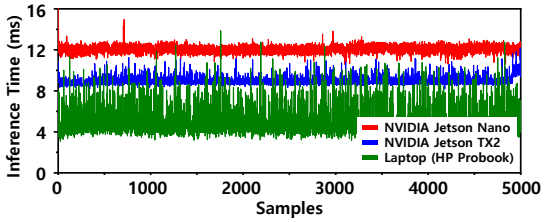


Fig. 24: Execution times of online inferences.