Ripple: Software-Only Detection of Signal Injection Attacks in Drone Temperature Sensors

Milad Rezaee
University of Oxford
Oxford, United Kingdom
milad.rezaee.barzani@cs.ox.ac.uk

Sebastian Köhler
University of Oxford
Oxford, United Kingdom
sebastian.kohler@cs.ox.ac.uk

Kasper Rasmussen
University of Oxford
Oxford, United Kingdom
kasper.rasmussen@cs.ox.ac.uk

Abstract

Signal Injection attacks pose a serious threat to systems that rely on sensor information to determine their behaviour. Using such an attack, an attacker can remotely manipulate the values of a sensor by transmitting appropriately formed RF signals that induce a current in the sensor wires. For example, to manipulate the temperature sensor in a battery management system, to trigger thermal protection and shut down the battery. While a number of defence mechanisms have been proposed, they all need additional hardware to work. In this paper, we present Ripple, a fully software-based detection mechanism that can reliably detect signal injection attacks against sensor systems in drones. A software-only solution is a practical way to add protection to an existing fleet of drones, and it is a cost effective alternative to the existing proposals for new drones.

Our detection mechanism exploits a physical layer property known as small-scale (fast) fading, which causes the wireless channel between the attacker and drone to change unpredictably. As a result, the power induced by the attacker's transmission will oscillate rapidly, whenever the drone is in motion. We show for the first time that this effect occurs even with extremely minimal motion, such as a drone hovering in place on a calm, windless day. This oscillation is used as the basis of our detection system. We conduct an in-depth evaluation of Ripple on drones in several different environments. Our results show that Ripple reliably detects signal injection attacks. Even for weak attacks, changing the temperature by as little as 2°C, and with a drone movement of only a few millimeters, we have a success rate of over 98%. The performance only improves with stronger attack signals or more movement.

CCS Concepts

• **Security and privacy → Embedded systems security**.

Keywords

Sensor System Security, Software Defense agains IEMI Attacks

ACM Reference Format:
Milad Rezaee, Sebastian Köhler, and Kasper Rasmussen. 2025. Ripple: Software-Only Detection of Signal Injection Attacks in Drone Temperature Sensors. In *Proceedings of the 18th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '25).* ACM, New York, NY, USA, 12 pages. https://doi.org/XXXXXXX.XXXXXXX

WiSec '25, June 30–July 3, 2025, Arlington, Virginia, USA
2025. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Sensor systems are vulnerable to signal injection attacks, also called intentional electromagnetic interference (IEMI) attacks, in which electromagnetic waves (radio transmissions) induce unwanted voltages in wires or circuit board traces, resulting in manipulated sensor readings. These attacks have been demonstrated on a wide range of sensor types, including temperature sensors [28]. Unfortunately, the susceptibility of sensors to electromagnetic signals is an inherent part of their design, governed by the laws of physics, making it difficult to completely eliminate. As a result, various hardware-based solutions have been proposed to enable microcontrollers to detect IEMI and injection attacks [16, 22, 28, 33]. Although these approaches can be integrated during the design and development of new sensor systems, they come with trade-offs: they tend to increase production costs and device size and are difficult to retrofit into existing sensor systems.

In this paper, we address the limitations of hardware-based detection systems by introducing Ripple, a lightweight and completely software-based IEMI detection mechanism. The key idea behind Ripple is the observation that during an IEMI attack, even the slightest movement of the target device alters the RF channel between the attacker and the target in an unpredictable way due to small-scale fading effects. As a result, any external signal that induces a voltage in the sensor system will fluctuate in amplitude and cause measurable oscillations in the sensor readings, regardless of the attack strategy or signal type. This phenomenon allows us to create a highly effective detection method that can be easily applied to both existing and future devices, such as drones, through a simple software update at minimal cost.

It might appear as if the movement would have to be at least half a wavelength of the attackers carrier signal for this to work, but that is not the case. We do not need the new channel environment to be entirely independent, we just need it to change. Any movement, however tiny, is enough to trigger a change in induced power.

Our focus is on temperature sensors, as these sensors are among the most widely used sensors today. They are used as a proxy for system health in battery management systems, motors, and processors, as well as for actual temperature sensing in environmental applications. Manipulation of these sensors can cause devices to shut down or, in the case of drones, force an emergency landing [1, 12].

We perform extensive experiments on 3 commercial off-the-shelf (COTS) temperature sensors to validate our approach and demonstrate that even very subtle vibrations – just a few millimeters – can induce unpredictable changes in the RF environment. These changes are significant enough to be exploited by Ripple to successfully detect attacks. Our experiments cover a variety of environments and include a range of motion and attacker power.

In summary, we make the following contributions:

- We demonstrate the impact of small-scale fading on the induced signal in sensor wires caused by IEMI. Specifically, we show that motion leads to significant detectable oscillations in the amplitude of measured sensor readings.
- We propose two fully software-based algorithms for realising Ripple: one based on variance analysis and another using Min-Max aggregation.
- We mathematically analyze these algorithms and provide a framework for setting detection thresholds tailored to the specific requirements of different applications.
- We perform extensive experiments to evaluate the performance of our algorithms using COTS sensors in different environments. Our tests demonstrate that Ripple is robust and environment independent.

2 Related Work

A large amount of academic research has highlighted the importance of sensor reading integrity, with attack channels based on electromagnetic (EM) waves [16], ultrasound [23], [11], infrared [24], and light [15]. In particular, signal injection using IEMI is effective against many classes of sensor types such as microphones [30], cameras [14], Inertial Measurement Units (IMUs) [10], and temperature sensors [28].

A number of countermeasures have also been proposed, summarized in [31] and [7]. These countermeasures can be categorized into two main approaches: hardware and software-based. Hardware-based methods employ particular materials or electronic elements to deal with IEMI attacks, while software-based methods do not require any additional hardware and the design of the defense mechanism relies only on data collected by the sensors and the anomaly detection algorithm running on a microcontroller. The most common hardware-based approaches are shielding and the use of robust hardware, which is less susceptible to intentional and unintentional electromagnetic interference. Shielding reduces the risk of IEMI attacks by providing a physical barrier that blocks or diverts electromagnetic energy away from the electronic devices that need protection [16–19, 21]. For example, researchers in [16] show that an imperfect shielding with openings attenuates injected signal up to 40 dB. Using circuits with less non-linearity or adding low-pass filters also makes it more difficult for the attacker to inject a signal into the circuit and affect the sensor output [8], [13]. However, these hardware defenses cannot detect attacks, nor can they completely prevent them; instead, they only slightly raise the bar for the attack.

There are some other proposed hardware defense mechanisms capable of attack detection [16], [28], [33], and [4], however, additional hardware components are still needed for implementation. Although these hardware-based approaches work in many applications, for some scenarios, especially when upgrading existing devices, a fully software-based solution is more desirable. This is because by applying software-based approaches, there is no need for any additional hardware and the defense mechanism can be added as a patch to address these vulnerabilities in the existing systems. This makes software solutions preferable, as they have no overhead costs and can be easily implemented on all existing systems through a software update. In addition, adding hardware components may reduce system performance in some situations. For example, adding hardware to a drone not only increases the overhead cost of producing a drone, but also makes it heavier, which decreases flight efficiency by reducing maximum flight time. To compensate, a more powerful battery is needed, which also increases the price and weight.

There are only a few software-based defense mechanisms in the literature, such as [26] and [3], where in [3] authors propose a machine learning approach to detect acoustic attacks on gyroscopes and magnetometers in smartphones, and authors in [26] propose a software-based sensor recovery system for robotic vehicles. Although the solution presented in [26] works well in its intended application, i.e., to recover sensor data for a short period of time, it requires relatively high computational power and has considerable runtime overhead, which adds to the complexity of the system. Similarly, the software approach proposed in [3] requires a large training set and has been only designed for smartphones. Furthermore, it has not been designed to optimize memory usage and computational power. To address these limitations, we present Ripple, which is a fully software-based detection mechanism against IEMI attacks that only requires a few bytes of memory and demands almost no computational power. To the best of our knowledge, Ripple is the first fully software-based detection mechanism against IEMI attacks in the literature that exploits channel properties, such as small-scale fading effect.

3 Background

3.1 Signal Injection into Sensor Systems

Electromagnetic waves can induce a voltage difference in nearby conductors. Since sensor system wires and PCB traces are not designed to act as antennas, this phenomenon is known as back-door coupling or IEMI. An attacker can exploit this phenomenon to induce voltage differences in the wires of a sensor system, resulting in manipulated sensor readings. The amplitude of the voltage difference, and therefore the discrepancy with the actual sensor reading, increases as the strength of the electromagnetic field increases. Each circuit component has its own specific operating frequency. To make the attack possible, attackers must find a suitable frequency to transmit their signal, often referred to as the resonant frequency. For example, an attacker aims to send a signal with a wavelength proportional to the length of the sensor wire into which the signal is supposed to couple into. Since the length of sensor wires and PCB traces is small, malicious signals are usually modulated onto high-frequency carriers, which are demodulated to in-band signals by non-linear components, such as amplifiers and analog-to-digital converters (ADCs). This technique has been used in many previous studies [5], [6], [20], [27], [28], [29], [32] and [25].

3.2 Small-Scale Fading

Small-scale fading is a phenomenon experienced by all RF channels where the amplitude of a wireless signal fluctuates significantly with small-scale changes in the environment. It is caused by several factors, such as non-uniform reflection, diffraction, and scattering of objects in the environment, as well as multi-path effects, i.e., the signal taking multiple paths to reach the receiver. These signal

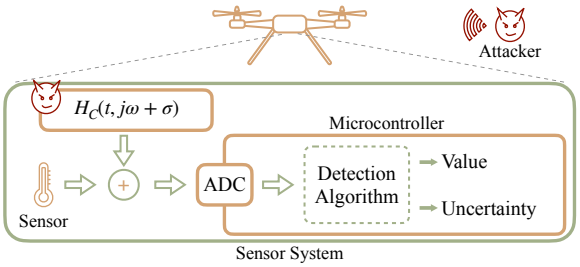


Figure 1: An illustration of system model. The detection algorithm is discussed in detail in Sections 6 and 7.

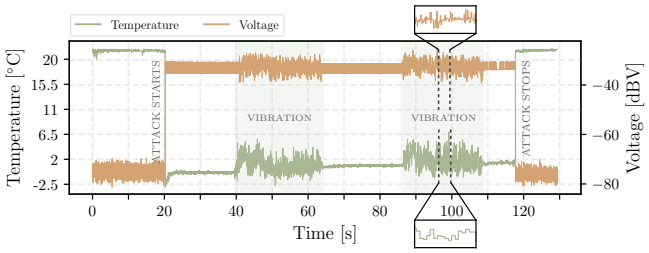


Figure 2: Example of how the vibration of the drone changes the wireless channel between the attacker and the target, causing the IEMI signal to induce a noisy signal that fluctuates in amplitude rather than a clean DC offset. The fluctuations are observable in both, the digital temperature output and the analog voltage measured in the wires of the sensor.

amplitude fluctuations are due to constructive and destructive interference of different parts of the transmitted signal that are received with different delays by the receiver. To quote [9]: "in small-scale fading, the instantaneous received signal power may vary as much as 30 to 40 dB when the receiver is moved by only a fraction of a wavelength".

4 System and Threat Model

4.1 System Model

Figure 1 shows our system and attacker model. The system consists of an analog sensor connected to a microcontroller through an analog-to-digital converter (ADC). The microcontroller periodically samples the analog signal to obtain a digital value. The value is sent to the detection algorithm which outputs two parameters: (1) A value representing the average of a number of raw samples, and (2) the uncertainty which indicates how (un)certain the value is.

Ripple detects attacks based on the raw digitized samples, so it can be used regardless of how the analog front-end works. We propose two different detection algorithms that differ in accuracy and resource consumption, described in Section 6.

4.2 Threat Model

The attackers transmission induces an IEMI signal in sensor wires on a victim device. This is an additive process where the IEMI signal is added to the existing sensor signal before being sampled by the ADC. We model the channel between the attacker and the sensor as a transfer function $H_C(t, j\omega + \sigma)$ which depends on time t, frequency ω and phase σ.

The attacker's goal is to modify the measured output of the sensor to achieve a change in behavior, without getting detected. The amount of change required is modeled by a threshold parameter.

The adversary has no physical access to the sensor system and is attacking the victim system remotely by transmitting electromagnetic (radio) waves. We do not place upper limits on signal power. The adversary is aware of the make and model of the sensor as well as the details of our detection mechanism; and can make a good guess of the approximate sensor output.

Finally, we make standard assumptions about the wireless channel, i.e., the adversary cannot know (or control) channel changes in response to movements and vibrations of the victim device as described in Section 3.2.

5 Ripple

5.1 Principle Concept of Ripple

The wireless channel between an attacker and the target sensor system in a drone will vary over time due to both the movement and vibration of the drone and changes in the environment. The key observation that enables our detection mechanism is that any tiny amount of movement of the drone is enough to significantly change the channel transfer function. This means that the amount of induced current caused by an IEMI attack will change rapidly and unpredictably as long as the drone is moving, even if that movement is only a few millimeters. Because this phenomenon is unpredictable, it causes the attacker's signal to induce rapid changes in the sensor output whenever the attacker transmits in a way that cannot be compensated for. We use this observation to design a fully software-based detection mechanism that detects these changes to reveal the presence of an adversarial (external) signal.

5.2 Proof of Concept Experiment

Before describing our algorithms we first demonstrate the existence and magnitude of the rapid changes with a simple experiment. We use an off-the-shelf temperature sensor (MAX31855, K-type thermocouple) and mount it on a X8+ drone from 3D Robotics. To cause a tiny but repeatable amount of movement we have unbalanced the propellers of the drone to cause vibration when the drone propellers are spinning. Later in Section 8 we perform extensive experiments on drones in different environments but at this stage we want to make sure that the effect is from the vibration and not some other source in the environment. For this reason we use vibration as a proxy for tiny movements so we can perform the experiments in a controlled environment, free from other external signals. The range of movement from the vibrations is on the order of a few millimeters which is smaller than the movement of a drone in flight, even when hovering in place.

We use an arbitrary waveform generator as a transmitter and an oscilloscope to record the resulting signal injected into the thermocouple wires. The attacking signal is sent through an antenna at distance of 30 cm from the drone. Figure 2 shows the analog voltage of the thermocouple wire, and the resulting digital sensor temperature reading. Before the attack starts, the voltage level is around

−75 dBV and the temperature is stable at around 22 °C. When the attack starts the voltage increases and the sensor output drops to around 0 °C. 10 seconds later when we start the vibration, the fast fading of the channel causes voltage fluctuation in the attackers signal, which in turn results in fluctuation of the digital temperature samples. We can clearly observe that the fluctuation of the signals are correlated perfectly with the periods of vibration, confirming our hypothesis that the vibration is causing the fluctuations. We go into more details on this in Section 8, but this is enough for us to start designing a detection mechanism around this phenomenon. We note that we used an oscilloscope to show the injected voltage fluctuation in the sensor wires, however, Ripple does not require a fast ADC to detect these attacks.

6 Attack Detection

In this section, we propose two fully software-based detection mechanisms, one based on variance, and one based on Min-Max.

We design our detectors with the awareness that they must run on flight controllers and other devices with limited computing resources. For this reason we have given high priority to solutions that use as few resources as possible, while still successfully performing the detection. There are many different statistical methods to measure rapid changes in the sensor output, however we found that simple variance provides the best trade off between detection power and resource consumption. We note that none of our approaches below requires a fast Analog-to-digital converter.

6.1 Variance-based Approach

One of the basic mathematical tools that quantifies variation in a series of data points is variance. In this variance-based approach, the detector measures the average value and the variance of the sensor output over a sample window of m. At the end of the sample window it outputs the average as the new value and the variance as the uncertainty of the measurement. This enables the consumer of these values, e.g., the flight controller, to make nuanced decisions about what values to accept in different situations.

The approach works as follows. The flight controller executes Algorithm 1 at regular intervals to sample the sensor. Each time the algorithm is executed it updates its internal registers and once m samples have been processed the algorithm outputs two values: an average value over the sample window, and an uncertainty value based on variance.

The variance of m samples would normally be calculated as follows:

$$V = \sum_{i=1}^{m} \frac{(x_i - X_{mean})^2}{m} \tag{1}$$

where x_i is the ith sample and X_{mean} is the average of all samples in the sampling window. However, there are two practical problems with this approach. First, we would have to store all the samples before they could be processed, and second, the computation of mean and variance would be unevenly distributed at the end of the sampling window.

Given the storage and computational limitations of flight controllers, we overcome these problems by splitting the computation of both the mean and the variance over all samples. Calculating the mean requires only that we know the size of the sampling window, so this must be provided as an input to the algorithm. For reasons we will get to shortly, we also require that this size is a power of 2, i.e., it can be expressed as 2^k.

Algorithm 1 Variance-based detection algorithm

Input: window size: $m = 2^k$
Output: Value and Uncertainty

```
1   i++
2   x = GetRawSample()
3   Xmean += (x >> k)
4   Var += ((Xlast - x)(Xlast - x)) >> k
5   if (i == m)
6   {
7       output Xmean        // Value
8       output Var          // Uncertainty
9       i = 0
10      Xlast = Xmean
11      Var = 0
12  }
```

To calculate the variance, we need to know the average of the samples in the sample window, which we cannot know if we are only halfway through. Our solution to this problem is to use the mean of the previous sample window as a proxy. This is an approximation, but since we design this algorithm for sensors that measure properties like temperature that does not normally change rapidly, we argue that it is sufficient. This method allows us to add each term of the sum in Equitation (1) to a running total each time a raw sample is taken.

Looking at Algorithm 1, on line 1 we increment the sample number i to indicate that we are starting a new sample. We then store the raw sample x on line 2 and perform the amortized part of calculating the mean and variance on lines 3 and 4. In these two lines, we take advantage of the fact that $m = 2^k$ is a power of two, so dividing by m is the same as right-shifting by k which is considerably faster and can be done by even the tiniest microcontroller. When we reach the end of a sampling window, i.e., if $i = m$, we need to reset the sample counter as well as the mean and variance computation, and output the computed values.

Algorithm 1 only needs space to store seven values: m, k, i, x, X_{mean}, X_{last}, and Var. Although this approach is already lightweight and does not require much memory, we wanted to further reduce the complexity of the computation. Motivated by this, we propose an alternative Min-Max-based approach that requires even less space to run the algorithm.

6.2 Min-Max-based Approach

In this approach, the flight controller executes Algorithm 2 at regular intervals to sample the sensor. Just like in the previous approach, each time the algorithm is executed it updates its internal registers and once m samples have been processed the algorithm outputs two values: a "sort of" average temperature value over the sample window, and an uncertainty value based on the difference between the minimum and maximum values in the sampling window. The

Algorithm 2 Min-Max-based detection algorithm

Input: window size: m
Output: Value and Uncertainty

```
1   i++
2   x = GetRawSample()
3   if (x > Max) Max = x
4   if (x < Min) Min = x
5   if (i == m)
6   {
7       output (Max + Min) >> 1 // Value
8       output (Max - Min)      // Uncertainty
9       i = 0
10      Max = -inf
11      Min = inf
12  }
```

"average" value computed by Algorithm 2 is the midpoint between the extreme values, i.e., (Max + Min)/2.

Calculating these two values allows us to completely eliminate the need for computation when sampling raw sensor values, except to check if the newly sampled value needs to replace the currently stored Min or Max. At the end of the sampling window, we only have to perform two very simple computations to convert our Min and Max values into the two output of the algorithm — value and uncertainty. Furthermore, this approach allows us to relax the requirement that the sampling window must be a power of two.

Algorithm 2 starts out similar to the previous approach in that on line 1 we increment the sample number i to indicate that we are starting a new sample, and store the raw sample x on line 2. All that is left to do is to update Min or Max if needed, which we do on lines 3 and 4. At the end of the sample window, i.e., when $i = m$, we compute the two outputs on lines 7 and 8, taking advantage of the fact that a division by 2 is the same as right-shifting by one. Finally, we need to reset the sample counter i as well as Min and Max (lines 9-11).

This approach comes with some trade-offs compared to the more robust variance-based approach. Algorithm 2 is much more sensitive to outliers or spikes in the measurement, with a single spike being enough to offset the "average" for the block of samples. Such a spike however would also cause the uncertainty to be high, so the consumer of the values, e.g., a flight controller, would likely not use such erroneous values. If the raw samples are relatively consistent this can yield results that are almost as good as the variance-based approach, with much less computational and storage overhead. The storage requirements for Algorithm 2 are only five values: m, i, x, Min, and Max. This might seem like a trivial amount of improvement over Algorithm 1 but for some tiny microcontrollers every register counts, especially if sampling the sensor is not the only task it needs to perform.

Note that both approaches are very lightweight and many sensor systems could run either without any problems. In Section 8, we experimentally evaluate both approaches on several such sensors.

7 Analysis of Detection Mechanisms

In this section, we discuss details of the detection methods and how to find appropriate thresholds for both approaches. We also state how we can use uncertainties in Algorithms 1 and 2 to calculate the reliability of measurements, which facilitates making more sophisticated decisions in the microcontroller. It should be noted that all computations in this section are performed offline and do not introduce any performance overhead to the algorithms when samples arrive.

7.1 Analysis of Variance-based Approach

The i-th sample of a sensor value can be modeled as follows,

$$x_i = X + n_i \tag{2}$$

where X is the value of the property that sensor measures, i is the sample number, and n_i is the noise. Here n_i models the aggregated effect of quantization noise, environment noise, and all the other possible noise that affect our measurement for i-th sample, which we assume that is Gaussian with variance n^2, we refer readers to Section 9.1 for more explanations regarding the noise distribution. At the presence of an attacker, we have sensor values with more variations compared to the case when there is no attacker as we see in Section 5.2.

If the sensor value variation caused by the attacker is substantially higher than n^2, it could be used as an indicator in the attack detection algorithm. In this approach, variance is a quantifier for the amount of this variation on the output of a sensor and based on its value we decide whether the sensor data is reliable. Assuming that, in the absence of an attacker, we have only a specific, limited variance in the sensor output, the attack can be detected if we measure a higher variance in the system. However, the variance over a block of m samples from the sensor also depends on the background noise that exists in the sensor data when there is no attack.

An ideal IEMI signal for an attacker is when the resulting variance Var is close to the background noise, such that it cannot be distinguished from it. According to Equation (2), when there is no attacker, each sample is composed of the exact amount of a physical quantity (X) and a Gaussian noise, n_i for i-th sample. Since n_i is a Gaussian variable, and $x_i - X = n_i$ for all i,

$$\frac{m\text{Var}}{n^2} = \sum_{i=1}^{m} \left(\frac{x_i - X}{n}\right)^2,$$

is a Chi-squared distribution with m degree of freedom. This means if we multiply variance ,Var, by $\frac{m}{n^2}$, the resulted variable, $\frac{m\text{Var}}{n^2}$, has a Chi-squared distribution. We form hypotheses $\mathcal{H}_0$, and $\mathcal{H}_1$, where $\mathcal{H}_0$, and $\mathcal{H}_1$ indicate sensor data is reliable, and not reliable, respectively. We also define p as the maximum tolerable false-positive rate. A threshold λ_V is calculated based on p, m, and n^2 as follows,

$$\lambda_V = \frac{n^2}{m}\text{cdf}^{-1}\left(\tilde{\chi}^2(p, m)\right), \tag{3}$$

where $\text{cdf}^{-1}(\tilde{\chi}^2(p, m))$ is inverse cumulative distribution function (cdf^{-1}) of the Chi-square distribution with degrees of freedom m. If $\text{Var} < \lambda_V$ holds, the sensor data in a block is reliable and $\mathcal{H}_0$ is true, otherwise $\text{Var} \geq \lambda_V$ and the sensor data is not reliable. The

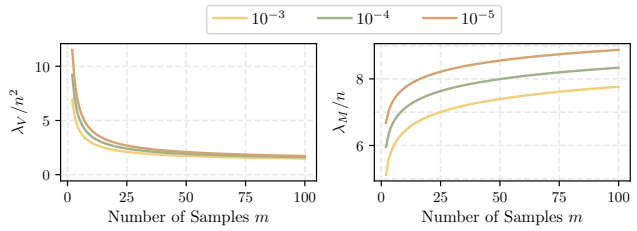


Figure 3: Normalized thresholds $\frac{\lambda_V}{n^2}$ and $\frac{\lambda_M}{n}$ vs. number of samples m in a block for 3 different p. The acceptable area is below curves. As we see by increasing p for both approaches the area below the curve decreases which means only less noisy signals can be accepted as legitimate.

probability that noise itself produces a variance larger than Var, which is the uncertainty in Algorithm 1, is calculated as follows,

$$C_V = \tilde{\chi}^2\left(\frac{m\text{Var}}{n^2}, m\right), \tag{4}$$

where $\tilde{\chi}^2(a, b)$ is the probability that a Chi-squared distribution with degree of freedom b is larger than a. If the microcontroller needs to make more sophisticated decisions than simply accepting or rejecting measurements based on threshold λ_V, it is possible to use the uncertainty Var in Equation (4) and make decisions based on the value of C_V as this is the probability that the sensor measurement is reliable.

As p increases, we can choose smaller thresholds and the attacker becomes more restricted. Therefore, p is a security parameter, which provides a trade-off between security and false-positive rate in the detection system. If p is chosen too small, it is easier for the attacker to bypass the detection system without getting detected, while if p is too large, we receive a high false-positive rate, which interrupts the sensor system and reduces the system's performance. m is also another system parameter. By choosing a small m for a fixed p, we reduce the detection time of an attack since we update decisions after receiving m samples, however, to obtain the same false-positive rate we need to choose a looser threshold, λ_V, which provide more freedom for an attacker to trick the sensor system (see Figure 3 on the left), which results in higher false-negative rate. According to Figure 3, for a fixed p as m decreases normalized threshold ($\frac{\lambda_V}{n^2}$) increases, which leaves more room for the attacker. In general for the variance-based approach, if m increases, the probability of a successful attack decreases, while the attack detection time increases.

7.2 Analysis of Min-Max-based Approach

In this approach, we consider the maximum variation in a window of m samples as a metric to decide whether the sensor data is reliable. We can define two hypotheses $\mathcal{H}_0$, and $\mathcal{H}_1$, where $\mathcal{H}_0$, and $\mathcal{H}_1$ indicate that the data is reliable, and not reliable, respectively. We also define p as the maximum tolerable false-positive rate and a threshold λ_M, which can be calculated based on m, p, and n. To calculate the probability of falsely indicating a block of data as unreliable, there should be at least two samples in a block that their difference is more than λ_M. The probability of a false-positive is as follows,

$$Pr_{f.p.} = Pr(\bigcup_{i,j\in[1,m],\ i<j} S_{ij}) \leq \binom{m}{2} Pr(S), \tag{5}$$

where $Pr_{f.p.}$ is the probability that we have at least two samples where their difference is at least λ_M and $Pr(S_{ij})$ denotes the probability that the absolute value of the difference between sample $i-$th and $j-$th (out of m samples is higher than λ_M). Since we have additive i.i.d. Gaussian noise for all m samples, $Pr(S_{ij})$ is the same for all i and j. Therefore, we can rename it to $Pr(S)$. After some simplifications and using the Chernoff bound $Q(x) \leq \frac{1}{2}e^{\frac{-x^2}{2}}$ given in [2], we can have an upper-bound for $Pr(S)$ as follows,

$$Pr(S) < 2Q\left(\frac{\lambda_M}{n}\right) + \frac{e^{-\frac{\lambda_M^2}{4n^2}}}{\sqrt{2}}, \tag{6}$$

where $Q(.)$ is Q-function. Here, combination of Equation (6) with (5) results in Equation (7) as the probability of false-positive using Min-Max-based approach with threshold λ_M.

$$Pr_{f.p} < \binom{m}{2}\left(2Q\left(\frac{\lambda_M}{n}\right) + \frac{e^{-\frac{\lambda_M^2}{4n^2}}}{\sqrt{2}}\right) \approx \binom{m}{2}\frac{e^{-\frac{\lambda_M^2}{4n^2}}}{\sqrt{2}}. \tag{7}$$

From Equation (7), we are able to calculate threshold λ_M based on p, m, n as follows,

$$\frac{\lambda_M}{n} > 2\sqrt{-\ln\left(\frac{p\sqrt{2}}{\binom{m}{2}}\right)}. \tag{8}$$

If Max-Min $< \lambda_M$ holds, the sensor data in a block is reliable and $\mathcal{H}_0$ is true, otherwise Max-Min $\geq \lambda_M$ and the sensor data is not reliable. The probability that noise causes maximum difference Max-Min, which is the uncertainty in Algorithm 2, is bounded as follows,

$$C_M = \min\left\{1, \binom{m}{2}\frac{e^{-\frac{(\text{Min-Max})^2}{4n^2}}}{\sqrt{2}}\right\} \tag{9}$$

Similarly if the microcontroller needs to make more sophisticated decisions than simply accepting or rejecting measurements based on threshold λ_M, it is possible to employ uncertainty Max-Min in Equation (9) and make decisions based on the value of C_M as this is the probability that the sensor measurement is reliable.

Figure 3 on the right illustrates the relation between λ_M/n (normalized threshold) and the number of samples for 3 different p. By increasing p we would be able to choose a smaller threshold and make the attacker more restricted.

8 Evaluation of Ripple

In this section, we present an in-depth evaluation to verify our hypothesis and support the initial experiments conducted in Section 5.2. Since sensors are usually inside an enclosure, attacks from a larger distance require high-power signals. While IEMI attacks with high power are possible, as demonstrated in [10], we decide to keep the transmission power low to ensure compliance with government regulations. Therefore, we instead evaluate Ripple on 3 different COTS temperature sensors, which gives us more control over the

experiments and the data captured. We conducted extensive experiments on thermocouple temperature sensor MAX3185, where we mount the sensor on a drone in various environments and evaluate the performance of our detection mechanisms by obtaining false and true positive rates from each algorithm. We then extend our experiments on two other temperature sensor in Appendix to show the generality and effectiveness of Ripple on detecting IEMI attacks on different temperature sensor types.

All experiments are conducted in the line-of-sight (LoS) scenario because the non-line-of-sight scenario would increase the channel fading effects, thus making detection easier. We note that our experiments consider all noise sources involved in the communication between a drone and a ground control station (GCS), as we conduct them on a real drone. In our experiments, we consider the worst-case scenario for detection, i.e., LoS between the drone and the attacker and movement of only a few millimetres. To verify the effectiveness of Ripple under more realistic settings, we conduct an attack against a temperature sensor attached to a flying drone.

8.1 Experimental Design

We run experiments in 4 different environments, a small lab, a medium-sized room, a park, and a backyard, with an actual flying drone, to confirm that Ripple is environment independent. The experiments in the medium-sized room, park, and backyard are at longer distances to demonstrate different fading environments and actual drone movements to confirm our results from the experiments in the small lab. We also conduct attacks using different antennas, omnidirectional and directional, to test the performance of Ripple. As more complex signals can be decomposed to sines, we use a single tone sine wave attack signal with a frequency that yields the maximum power transfer to the sensor. As described in Section 3, the success of IEMI attacks depends largely on the attack frequency F and the transmission power P. At the same, the frequency affects the channel variations caused by small-scale fading. To account for all these factors, we conduct an extensive list of experiments with various combinations of F and P. We run the experiment with frequencies between 600 MHz and 1 GHz (as these frequencies are the most effective ones against the MAX3185 temperature sensor) with steps of 25 MHz and different output powers. Due to the non-linear output of our amplifier, we measured the output power for each frequency used in our experiments and for different settings of the signal generator. The results are presented in Figure 10 in the Appendix for reference. The temperature sensor MAX3185 is attached to a X8+ drone from 3D Robotics, unless otherwise stated. In the small room, an omnidirectional antenna (VERT900) is placed roughly 50 cm away, while we use a directional antenna for experiments in the medium-sized room, park and backyard from a larger distance. To make our measurements reproducible and more reliable, we use a Python script to automate all measurements. Using Python, we can control the signal generator and change the frequency and the power of the attack signal and the drone motors at the same time. We record vibration data using an Arduino to be able to store everything as csv files. Our source code is publicly available.[1] An example setup of one of our experiments is depicted in the Appendix in Figure 13.

[1]The link has been removed for anonymization reasons.

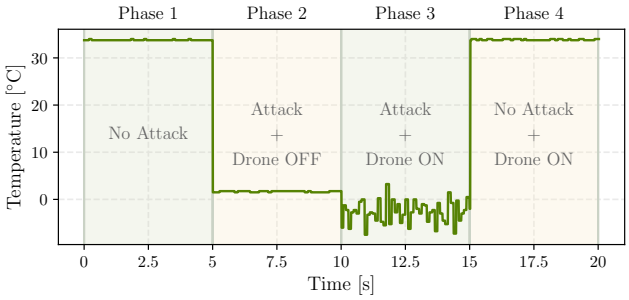


Figure 4: The four phases on one of our measurements at the attack frequency of 925 MHz**.**

We split each experiment into four phases of equal length. During each 5 s phase, we collect data as described above.

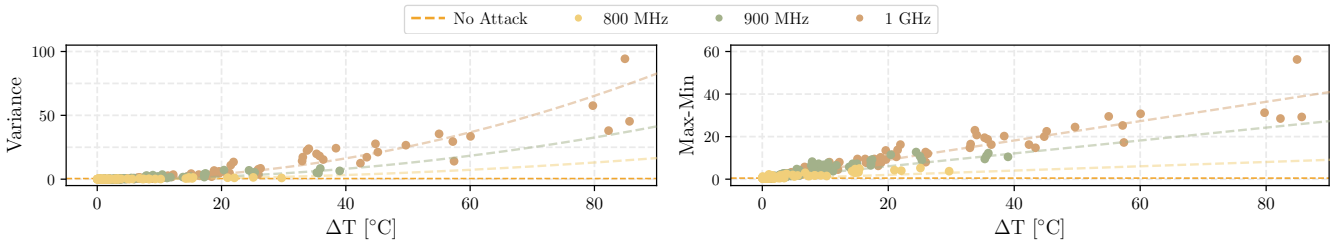
Phase 1 – Idle State. In the first phase, we collect temperature data during an Idle State, i.e., the attack signal is not present and the drone is switched off. This ensures that we account for potential electromagnetic interference coming from the environment.

Phase 2 – Attack & Drone Off. In the second phase, we turn on the signal generator and emit an attack signal at a given frequency F and power P.

Phase 3 – Attack & Drone On. In the third phase, we continue emitting the attack signal while we switch the drone on. We have imbalanced a propeller which causes the entire drone to vibrate slightly. This simulates the subtle movements a hovering drone experiences, but in a way that is more repeatable in our measurement setup.

Phase 4 – No Attack & Drone On. In the final phase, the attack ceases, but the drone is still switched on. Collecting data during this phase enables us to eliminate potential interference from the drone itself or from other sources in the environment.

For each phase we give the system a few seconds for everything to stabilize and then we start recording data. This is especially important for the third phase, to get a consistent amount of vibration when the propellers start spinning. In fact, during the transition period from the phase 2 to the phase 3, we have substantially larger sensor output variations as the range of movement is larger and the detection is easier. Figure 4 illustrates a typical sensor data from the four phases described above. From 0 to 5 s we record data when there is no attack signal by the signal generator and the temperature is around 34 °C. When the attack starts at 5 s in the second phase the sensor output drops to 2 °C and stays almost without any change since the channel between the antenna and the sensor is fixed and not varying because everything is static and drone is not moving. However, in the third phase, the propellers start to spin, causing vibration, and the channel continuously changes between the antenna and the sensor. Thus we see temperature fluctuation since the received power varies. In the last phase, the drone is still moving, but there is no attack signal and the sensor shows the true value of temperature again which confirms that temperature



(a) Relationship between variance and change of temperature.

(b) Relationship between Max-Min and change of temperature.

Figure 5: (a) illustrates the relationship between variance and temperature change during the third phase for frequencies 800 MHz, 900 MHz, 1 GHz and for different transmission powers. The antenna position is fixed at a distance of 50 cm from the sensor. As the transmitted power increases, the attacker's temperature offset increases and consequently variance of temperature. (b) illustrates the relationship between Max-Min and temperature change during the third phase. Similarly, as the transmitted power increases, the attacker's temperature offset increases, consequently causing Max-Min to increase. Number of sample per block is $m = 32$.

variations in the third phase only resulted from IEMI and not an interference from the drone itself.

8.2 Experimental Results in Small Lab

Figure 5a illustrates the relationship between variance of temperature and the temperature offset caused by the attacker in the third phase, for all measurement frequencies of 800 MHz, 900 MHz, and 1 GHz. As we increase the transmitted power, the average temperature DC offset caused by the attacker increases, and the variance of temperature due to variation of the channel as the result of drone movement increases. This means that, unsurprisingly, it is easier to detect attacks when the attacking signal is stronger. A similar result can be seen when we use our Min-Max-based approach, as can be seen in Figure 5b.

To evaluate our algorithms we first calculate noise power n^2 and then thresholds λ_V and λ_M for both the variance and Min-Max-based approaches. These depend on noise variance n^2 defined in Equation (2), and parameters m and p. To this end, we use the temperature sensor output in the first phase of the experiment (where the attacker is not present) to calculate the variance (over a 3 s window of data, i.e., $m = 32$). We take average these variance measurements as the noise power. The result of this calculation is 0.01, which means $n^2 \approx 0.01$. This indicates that thresholds, λ_V and λ_M for the variance- and Min-Max-based approaches should be sufficiently higher than $n^2 = 0.01$ and $n = 0.1$, respectively, to obtain a small false positive rate according to the analysis in Section 7. For instance, $\frac{\lambda_V}{n^2}$ and $\frac{\lambda_M}{n}$ must be larger than 2 and 7, respectively, to yield the false positive 10^{-3} for $m = 32$ (See Figure 3). Note that there are other factors such as sensor failures and interference that might result in a higher false positive rate, however, we have only considered the effect of noise in Section 7. Therefore, these are minimum thresholds to satisfy the false positive requirement.

To show the performance of Ripple for all possible thresholds, we plot Receiver Operating Characteristic (ROC) curves for each approach. A ROC curve represents true positive rates versus false positive rates of a classifier for all classification thresholds. The best possible performance of a classifier is when the true positive rate is 1 and the false positive rate is 0. Hence, as the distance of a ROC curve to point $(0, 1)$ decreases the performance of the classifier improves. Figures 6a and 6b illustrate ROC curves for variance and Min-Max-based approaches, respectively. We classify different attacks based on the amount of temperature offset they add to the sensor output. As shown in Figures 5a and 5b, the performance of both approaches improves as the temperature offset, and consequently the variance and Max-Min, increases due to the attack. This finding is further confirmed in Figures 6a and 6b. For instance, using the variance-based approach, we can detect almost 98% of the attacks that change temperature above 2°C with a false-positive of 0.54% with choosing $\lambda_V = 0.024$. If the sensor system is resilient against a few degrees of temperature change and only offsets above 5°C are important then we can detect 99.5% and 98.4% of attacks with false-positive rates of 0.54% and 0.27% choosing $\lambda_V = 0.024$ and $\lambda_V = 0.04$, respectively.

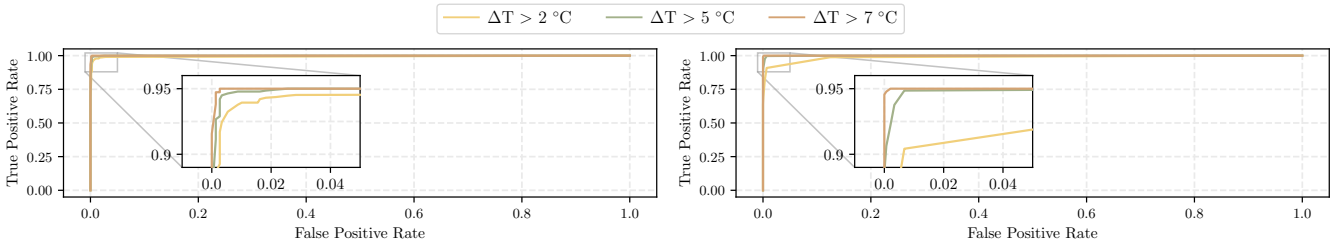
For the Min-Max-based approach, we are able to detect all the attacks that add offsets above 7°C within 1482 times of repeating attack signal with a false-positive rate of 0.2% when we choose $\lambda_M = 1.25$. For Attacks with $\Delta T > 5$°C, we can detect 99.7% and 97.5% of attacks with false positive rates of 0.67% and 0.34% choosing $\lambda_M = 0.75$ and $\lambda_M = 1$, respectively.

As seen from Figure 6, the Min-Max-based approach slightly under performs the variance-based, especially to detect smaller attacks, i.e., $\Delta T > 2°C$. However, it uses less resource such as storage space, and computation which is its advantage.

8.3 Experiment in Other Environments

To evaluate Ripple in other environments with the minimum multi-path effect we conduct some experiments in a medium-sized room and in a park, where there are no walls in the vicinity of the antenna or the drone.

Medium-sized room. We run the same four phase experiment similar to the small lab, using the same frequencies and transmission powers, see Figure 10 in the Appendix, but with a directional antenna, fewer repetition, and a different distance 3.1 m. The results align with those obtained in the small lab, indicating that Ripple is not dependent on the size of the room in which the experiments are conducted.



(a) Performance of variance-based approach.

(b) Performance of Min-Max-based approach.

Figure 6: True-positive vs. False-Positive Rates for both of the presented approaches — Variance and Min-Max. Number of samples per block is $m = 32$.

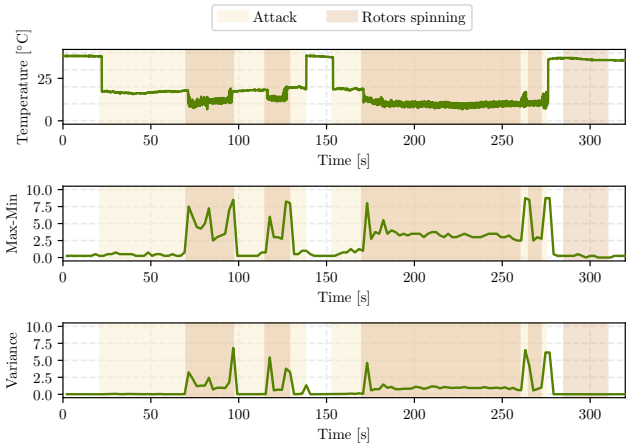


Figure 7: Different metrics collected during an experiment outdoor on the MAX3185 temperature sensor in a large open field. Even though multipath is reduced to a minimum due to the open environment, high temperature fluctuations, which results in high variance and Max-Min, are observed during the attack when the rotors are spinning and the drone is vibrating. The antenna is 4 m away from the drone, emitting a signal with power of around 23 dBm at 915 MHz.

Outdoor Experiments. To further verify our experimental results from the small lab in an environment with little to no multipath effect, we re-run the same experiment in a park. We position the directional antenna around 4 m away from the drone to transmit the attacking signal toward the drone. Figure 7 shows the result of the experiment and the output of each detection mechanism. The top figure shows the temperature recorded during the experiment. The middle figure shows Max-Min of sensor output for each 2.3 s and the bottom figure shows the variance of samples during each 2.3 s. As we see, during the time that the attacker is present and rotors are spinning, variance and Max-Min are substantially higher than the time that the attacker is not present and both detection mechanisms can detect attacks. During a small time window between 250 s and 300 s, where rotors stop and then shortly start to spin again while the attack is on we have a higher variance and Max-Min since the drone is not in a steady state situation and the range of movements are larger. This indicates that detecting an attack is easier in situations where the drone is not in a steady state, such as when the wind has just started to blow. This experiment shows that even in an environment with the minimum multipath effect Ripple is efficient and can distinguish reliable sensor data from unreliable ones. We refer to the Appendix for results of experiments on other temperature sensors.

8.4 Ripple in Flying Drones

In previous experiments, we have artificially created movement using unbalanced propellers. This helped us to ensure reproducibility and only cause subtle movements, which is considered to be the worst case scenario for Ripple. However, to evaluate the performance of Ripple for more realistic movements, we perform experiments on a flying drone. We attach the MAX3185 temperature sensor to a DJI Phantom 3 and attack the sensor at a frequency of 875 MHz with an output power of 30 dBm from a distance of 3 m while the drone hovers at a height of about 1 m. We use a high-gain directional antenna to deliver maximum energy to the target sensor, reduce multipath, and facilitate targeting while the drone is slightly moving. The result of this experiment is shown in Figures 8 and 9. In the first experiment, the antenna position is fixed, while in the second experiment we constantly adjust the antenna direction to follow the drone's position. Consistent with our expectations and previous experiments with artificially induced movements, temperature fluctuations are easily observable, and even larger in amplitude due to the increased movements of the drone. This experiment confirms that detecting an attack in a flying drone is even easier for Ripple, and emphasizes that the results of the previous experiments are representative while being considered the worst-case scenario.

9 Discussion

In this section, we cover a few loose ends and discuss potential limitations and extensions to Ripple.

9.1 Noise Distribution Assumptions

In our analysis of the two detection strategies (Variance and Min-Max in Section 7) we make the assumption that the distribution of environmental noise is Gaussian. This might lead some to wonder if this assumption is always true, and what happens if, for whatever reason, the distribution of environmental noise is different.

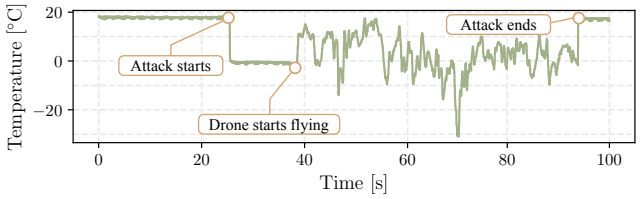


Figure 8: Temperature measurements during the attack with a fixed antenna when the drone is still and flying. Temperature variations are greater since the range of movements is larger compared to previous experiments.

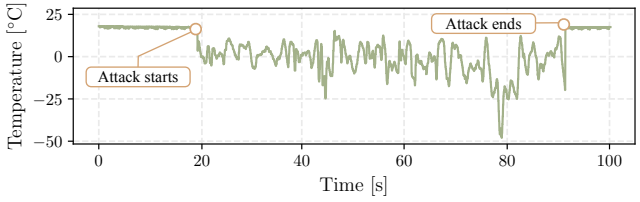


Figure 9: The drone hovers from the beginning to the end of the experiment. We constantly adjust the direction of the antenna to follow the drone. This experiment shows that even with adjustments in antenna position or direction, large variations are introduced, allowing Ripple to detect attacks.

The short answer is nothing. Neither of our two detection mechanisms rely on this assumption to perform their function, so it only comes into play when deciding what thresholds to pick for the uncertainty, in order for the flight controller to achieve a certain level of false positives.

Noise can only affect the performance of the detection if it is of sufficient amplitude to affect the sensor measurement, regardless of the distribution. In that case our algorithms will indeed output an increased uncertainty, since the noise source is external to the drone and therefore moves (ever so slightly) relative to it. As explained in detail in Section 5 this will cause an amplitude oscillation in the temperature measurement (again, regardless of the distribution) and so lead to increased uncertainty about the real measurement. However, that is by design since we have no way of knowing if the noise source is malicious.

9.2 Adversary Evasion Strategies

Throughout the paper we talk about how IEMI signals are detected because an attacker cannot avoid the fast fading effects of the channel between themselves and the target sensor system. However, we have not talked explicitly about what, if any, strategies are available to the attacker and how attacker behavior could affect the detection performance.

Could an attacker somehow predict the channel fading and compensate by modulating the transmitted signal? No. The attacker would have to know every detail about the environment, including the distance to the moving target with sub-millimeter precision. This is simply not feasible in practice. Work in other areas of security confirms this, as channel fading has been used as a basis for cryptographic key generation and as a source of entropy for random numbers.

Could an attacker increase the transmission power to a point where the oscillations no longer happen? No. In fact, our data indicates that a stronger adversarial signal only leads to stronger oscillations, and thus a higher uncertainty value in our detection algorithms. In theory there is of course an upper bound where the induced current will cause a denial of service to the on board electronics of the drone, but we consider such EMP-like attacks to be out of scope.

Could an attacker transmit at such low power as to make the oscillations fall bellow the detection threshold of the flight controller? Yes. An attacker who transmits a sufficiently weak signal will not be detected, but that weak signal will also not have an effect on the sensor measurement. Since the detection algorithm works in the digital domain, after the sensor measurement has been digitized, we cannot have a situation where a signal can affect the measurement value, but not be detected. If the measurement is affected enough to cause a sample to change, it will cause the uncertainty to go up.

Could an attacker somehow change the value slowly to avoid detection? No. Even if the adversary transmits a signal that keeps the new temperature constant, the fact that an external signal, subject to fast fading, is introduced into the sensor system, means that there will be oscillations. No amount of careful manipulation of the values will prevent detection.

It is not practical to account for every single possible attacker strategy, and the above list does not attempt to be exhaustive. That being said, we strongly believe, and all the data we have so far is confirming that, *any* signal regardless of its shape, phase, timing characteristics, amplitude, frequency, or anything else we have been able to test, is subject to the same underlying constraints imposed by the channel fading. It will cause oscillations, and we can detect those if they are anyway near strong enough to affect measurements.

10 Conclusion

In this paper, we present Ripple, a fully software-based detection mechanism against IEMI attacks. Ripple is designed as a lightweight algorithm that can be executed on almost any microcontroller, and is intended to be integrated into, e.g., flight controller software as an optional hardening feature. Being software-based it can be deployed to legacy systems with a software update, and has negligible memory requirements. Ripple works because of a fundamental property of the wireless channel called fast fading, a type of small-scale fading that is unpredictable and random in nature. This means that it is (practically) impossible for an attacker to avoid detection in any environment. We propose two versions of our detection algorithm, one based on variance and one Min-Max-based approach. We evaluate them mathematically and experimentally using a X8+ drone from 3D Robotics, DJI Phantom 3, and 3 COTS temperature sensors. We demonstrate that Ripple is effective and robust in different environments. To show the effectiveness of the algorithms, we present ROC curves for different classes of attacks, which show excellent performance.

Acknowledgment

Sebastian was supported by the Royal Academy of Engineering and the Office of the Chief Science Adviser for National Security under the UK Intelligence Community Postdoctoral Research Fellowships programme.

References

[1] [n. d.]. https://forum.dji.com/forum.php?mod=viewthread&tid=269943.
[2] Herman Chernoff et al. 1952. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics* 23, 4 (1952), 493–507.
[3] Hongjun Choi, Sayali Kate, Yousra Aafer, Xiangyu Zhang, and Dongyan Xu. 2020. Software-based realtime recovery from sensor attacks on robotic vehicles. In *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*. 349–364.
[4] Daisuke Fujimoto, Yu-ichi Hayashi, Arthur Beckers, Josep Balasch, Benedikt Gierlichs, and Ingrid Verbauwhede. 2018. Detection of IEMI fault injection using voltage monitor constructed with fully digital circuit. In *2018 IEEE International Symposium on Electromagnetic Compatibility and 2018 IEEE Asia-Pacific Symposium on Electromagnetic Compatibility (EMC/APEMC)*. IEEE, 753–755.
[5] Javier Gago, Josep Balcells, David GonzÁlez, Manuel Lamich, Juan Mon, and Alfonso Santolaria. 2007. EMI susceptibility model of signal conditioning circuits based on operational amplifiers. *IEEE Transactions on Electromagnetic Compatibility* 49, 4 (2007), 849–859.
[6] Hamid Ghadamabadi, James J Whalen, R Coslick, C Hung, T Johnson, W Sitzman, and J Stevens. 1990. Comparison of demodulation RFI in inverting operational amplifier circuits of the same gain with different input and feedback resistor values. In *IEEE International Symposium on Electromagnetic Compatibility*. IEEE, 145–152.
[7] Ilias Giechaskiel and Kasper Rasmussen. 2019. Taxonomy and challenges of out-of-band signal injection attacks and defenses. *IEEE Communications Surveys & Tutorials* 22, 1 (2019), 645–670.
[8] Ilias Giechaskiel, Youqian Zhang, and Kasper B Rasmussen. 2019. A framework for evaluating security in the presence of signal injection attacks. In *Computer Security–ESORICS 2019: 24th European Symposium on Research in Computer Security, Luxembourg, September 23–27, 2019, Proceedings, Part I 24*. Springer, 512–532.
[9] Ali Grami. 2015. *Introduction to digital communications*. Academic Press.
[10] Joon-Ha Jang, Mangi Cho, Jaehoon Kim, Dongkwan Kim, and Yongdae Kim. 2023. Paralyzing Drones via EMI Signal Injection on Sensory Communication Channels.. In *NDSS*.
[11] Jinseob Jeong, Dongkwan Kim, Joon-Ha Jang, Juhwan Noh, Changhun Song, and Yongdae Kim. 2023. Un-Rocking Drones: Foundations of Acoustic Injection Attacks and Recovery Thereof.. In *NDSS*.
[12] Shiqin Jiao, Guiyang Zhang, Mei Zhou, and Guoqi Li. 2023. A Comprehensive Review of Research Hotspots on Battery Management Systems for UAVs. *IEEE Access* (2023).
[13] Chaouki Kasmi and Jose Lopes Esteves. 2015. IEMI threats for information security: Remote command injection on modern smartphones. *IEEE Transactions on Electromagnetic Compatibility* 57, 6 (2015), 1752–1755.
[14] Sebastian Köhler, Richard Baker, and Ivan Martinovic. 2022. Signal injection attacks against ccd image sensors. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*. 294–308.
[15] Sebastian Köhler, Giulio Lovisotto, Simon Birnbach, Richard Baker, and Ivan Martinovic. 2021. They see me rollin': Inherent vulnerability of the rolling shutter in cmos image sensors. In *Annual Computer Security Applications Conference*. 399–413.
[16] Denis Foo Kune, John Backes, Shane S Clark, Daniel Kramer, Matthew Reynolds, Kevin Fu, Yongdae Kim, and Wenyuan Xu. 2013. Ghost talk: Mitigating EMI signal injection attacks against analog sensors. In *2013 IEEE Symposium on Security and Privacy*. IEEE, 145–159.
[17] A Theodore Markettos and Simon W Moore. 2009. The frequency injection attack on ring-oscillator-based true random number generators. In *Cryptographic Hardware and Embedded Systems-CHES 2009: 11th International Workshop Lausanne, Switzerland, September 6-9, 2009 Proceedings*. Springer, 317–331.
[18] Saki Osuka, Daisuke Fujimoto, Yu-ichi Hayashi, Naofumi Homma, Arthur Beckers, Josep Balasch, Benedikt Gierlichs, and Ingrid Verbauwhede. 2018. EM information security threats against RO-based TRNGs: The frequency injection attack based on IEMI and EM information leakage. *IEEE Transactions on Electromagnetic Compatibility* 61, 4 (2018), 1122–1128.
[19] Yuri V Parfenov, Leonid N Zdoukhov, William A Radasky, and Michel Ianoz. 2004. Conducted IEMI threats for commercial buildings. *IEEE Transactions on Electromagnetic Compatibility* 46, 3 (2004), 404–411.
[20] Nirupam Roy, Haitham Hassanieh, and Romit Roy Choudhury. 2017. Backdoor: Making microphones hear inaudible sounds. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*. 2–14.
[21] Jayaprakash Selvaraj, Gökçen Yılmaz Dayanıklı, Neelam Prabhu Gaunkar, David Ware, Ryan M Gerdes, and Mani Mina. 2018. Electromagnetic induction attacks against embedded systems. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*. 499–510.
[22] Yasser Shoukry, Paul Martin, Yair Yona, Suhas Diggavi, and Mani Srivastava. 2015. Pycra: Physical challenge-response authentication for active sensors under spoofing attacks. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 1004–1015.
[23] Yunmok Son, Hocheol Shin, Dongkwan Kim, Youngseok Park, Juhwan Noh, Kibum Choi, Jungwoo Choi, and Yongdae Kim. 2015. Rocking drones with intentional sound noise on gyroscopic sensors. In *24th USENIX Security Symposium (USENIX Security 15)*. 881–896.
[24] Takeshi Sugawara, Benjamin Cyr, Sara Rampazzi, Daniel Genkin, and Kevin Fu. 2020. Light commands:{Laser-Based} audio injection attacks on {Voice-Controllable} systems. In *29th USENIX Security Symposium (USENIX Security 20)*. 2631–2648.
[25] Marcell Szakály, Sebastian Köhler, Martin Strohmeier, and Ivan Martinovic. 2023. Assault and Battery: Evaluating the Security of Power Conversion Systems Against Electromagnetic Injection Attacks. *arXiv preprint arXiv:2305.06901* (2023).
[26] Kevin Sam Tharayil, Benyamin Farshteindiker, Shaked Eyal, Nir Hasidim, Roy Hershkovitz, Shani Houri, Ilia Yoffe, Michal Oren, and Yossi Oren. 2020. Sensor defense in-software (SDI): Practical software based detection of spoofing attacks on position sensors. *Engineering Applications of Artificial Intelligence* 95 (2020), 103904.
[27] Timothy Trippel, Ofir Weisse, Wenyuan Xu, Peter Honeyman, and Kevin Fu. 2017. WALNUT: Waging doubt on the integrity of MEMS accelerometers with acoustic injection attacks. In *2017 IEEE European symposium on security and privacy (EuroS&P)*. IEEE, 3–18.
[28] Yazhou Tu, Sara Rampazzi, Bin Hao, Angel Rodriguez, Kevin Fu, and Xiali Hei. 2019. Trick or heat? Manipulating critical temperature-based control systems using rectification attacks. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2301–2315.
[29] Yazhou Tu, Vijay Srinivas Tida, Zhongqi Pan, and Xiali Hei. 2021. Transduction shield: A low-complexity method to detect and correct the effects of EMI injection attacks on sensors. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*. 901–915.
[30] Zhifei Xu, Runbing Hua, Jack Juang, Shengxuan Xia, Jun Fan, and Chulsoon Hwang. 2021. Inaudible attack on smart speakers with intentional electromagnetic interference. *IEEE Transactions on Microwave Theory and Techniques* 69, 5 (2021), 2642–2650.
[31] Chen Yan, Hocheol Shin, Connor Bolton, Wenyuan Xu, Yongdae Kim, and Kevin Fu. 2020. Sok: A minimalist approach to formalizing analog sensor security. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 233–248.
[32] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. 2017. Dolphinattack: Inaudible voice commands. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*. 103–117.
[33] Youqian Zhang and Kasper Rasmussen. 2020. Detection of electromagnetic interference attacks on sensor systems. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 203–216.

A Transmission Powers

Due to the non-linear output of our amplifier, we measured the output power for each frequency used in our experiments and for different settings of the signal generator. The results are shown in Figure 10.

B Evaluating other Temperature Sensors

We present some experimental results on two other COTS temperature sensors (KY-013 and MAX6675) and the performance of Ripple to further validate the applicability of variance- and Min-Max-based approaches. Below we briefly describe the setup and results.

We first sweep from 50 MHz to 1 GHz to find the best attack frequency for these sensors. Then we place them on the drone and put the antenna 30 cm away from sensors to see the effect of movement on the sensor measurement when the antenna is transmitting. In the first experiment, we mount a KY-013 NTC thermistor temperature sensor on the drone and position the attacker antenna. The

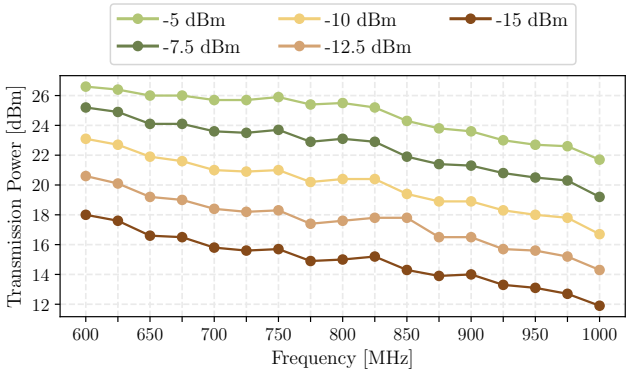


Figure 10: Power transmission in the small lab with distance 50 cm from the sensor for different frequencies. Powers in legend are signal generator powers before amplification.

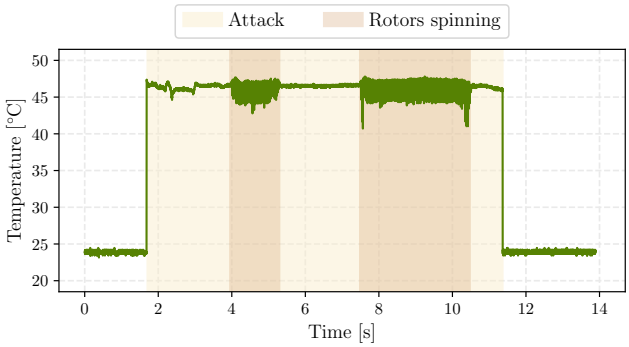


Figure 11: Illustration of how spinning rotors of the drone affect KY-013 thermistor output in the presence of an attacker. As it seen, even a small movement of a person in the room from $t = 2$s to $t = 4$s while the antenna is transmitting but the drone is not moving resulted in temperature fluctuations. This indicates that not only the drone movements but also the position and movement of objects in the environment affect the wirelesses channel and consequently the received power by a victim sensor, which is unpredictable and make it (practically) impossible to avoid by the attacker.

attack was performed at 30.8 dBm at frequency 346 MHz. In the second experiment, we do a similar experiment on the MAX6675 temperature sensor transmitting 25 dBm at frequency 361 MHz. Figures 11 and 12 show the temperature readings on sensors KY-013 and MAX6675, respectively, when the attack is on, and the drone is moving. As can be seen in these figures, there is a significant noise when rotors are spinning while the attack is on, which indicates that Ripple works well for attack detection on different temperature sensors. Even a small movement of a person in the room from $t = 2$s to $t = 4$s while the antenna is transmitting but the drone is not moving resulted in temperature fluctuations according to Figure 11. This indicates that not only the drone movements but also the position and movement of objects in the environment affect the wirelesses channel and consequently the received power by

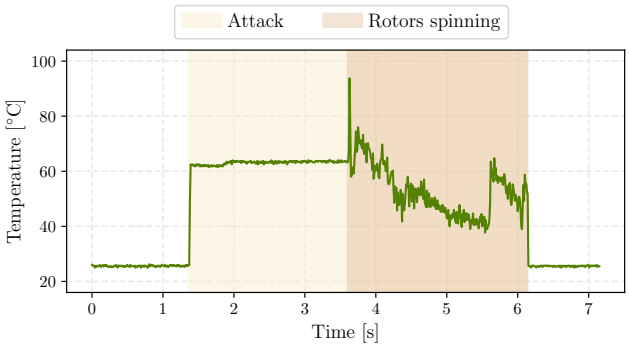


Figure 12: Illustration of how spinning rotors of the drone affect MAX6675 thermocouple sensor output in the presence of an attacker.

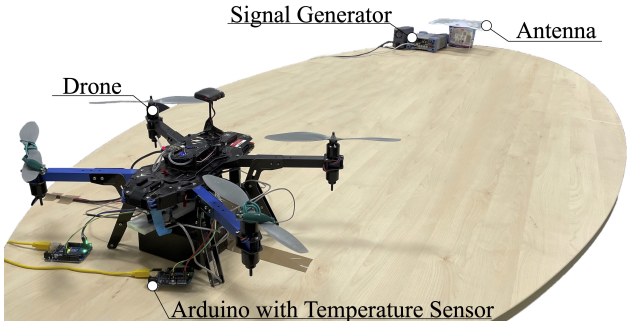


Figure 13: Experiment setup in the medium-sized room. We have mounted the temperature sensor on the drone and set a directional antenna at a distance of 3.1 m from the drone.

a victim sensor, which is unpredictable and make it (practically) impossible to avoid by the attacker.