# Protecting the Security of Sensor Systems

by

Connor Bolton

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in The University of Michigan
2022

Doctoral Committee:

Associate Professor Kevin Fu, Chair
Professor Mingyan Liu
Professor Morley Mao
Assistant Professor Sara Rampazzi

Connor Bolton

mcbolto@umich.edu

ORCID iD: 0000-0003-4079-7846

# TABLE OF CONTENTS

# LIST OF FIGURES

**Figure**

# LIST OF TABLES

**Table**

# ABSTRACT

Sensors are a ubiquitous part of modern life, providing crucial data about the physical state of the world in application areas including entertainment in smartphones and virtual reality, transportation in aviation or (semi-)autonomous vehicles, manufacturing, smart infrastructure, and more. Thus a crucial aspect of ensuring the availability, integrity, and confidentiality of these applications is to ensure the same qualities in sensor systems. However, research shows how sensors may produce undesirable output that compromises security or privacy due to interaction with physical signals. For example, research shows that microphone output, representing sound, can instead represent light, *a completely different quality*. Adversaries can use this vector to launch attacks on sensor-reliant systems.

This dissertation posits the question, "How is systemic design for mitigating physically-based sensing vulnerabilities possible?" and sets a goal of laying the groundwork to enable such systemic design. This work contributes:

1. *Methods, models, and language* to categorize and analyze the space of physical sensor security. The primary categorization is between transduction and oversensing vulnerabilities. The Transduction Attack Model (TAM) provides a mathematical model to describe and categorize existing transduction vulnerabilities. For oversensing, the Anti-Oversensing System (OA-Sys) identifies categories of oversensing.

2. *Mitigation design patterns* for many physical sensor vulnerabilities to aid manufacturers and operating system designers. Specifically, existing mitigations

for transduction vulnerabilities are categorized using TAM to reveal common design patterns to mitigate most oversensing vulnerabilities. OA-Sys provides preliminary mitigation designs for common sensor use-cases in smartphones.

3. *Specific case studies* of how to apply higher-level knowledge on transduction and oversensing vulnerabilities learned in TAM and OA-Sys to specific problems. Blue Note described two transduction vulnerabilities using acoustic waves to interrupt hard disk drive availability. Touchtone Eavesdropping uses motion sensor data to sense user input in smartphones via how motion sensors capture mechanically coupled sound.

# CHAPTER I

# Introduction

This thesis investigates physical vulnerabilities in sensors and how to defend against such vulnerabilities. Sensors often act as the bridge between the physical and digital worlds, quantifying physical phenomena for computer systems. Trillions of these sensors [29] are used to collect data for critical decisions in important infrastructure such as airplanes [10, 43], autonomous vehicles [140, 137], smartphones, medical devices, and more. However, sensor use also heightens the risk of physically-based vulnerabilities in computer systems.

This dissertation envisions systemic mitigation for threats against the integrity, availability, and privacy of sensor data and systems to enable manufacturers to consider and mitigate several physically-based vulnerabilities in the design phase. Existing approaches to defending sensor security tend to follow an endless cat and mouse game—security researchers find a physics-based exploit, then manufacturers deploy an exploit-specific patch rather than create an overarching and measurable security goal to address the root causes of that specific exploit. The lack of a measurable, goal-oriented approach to analog sensor security makes it difficult to apply science to defensive design.

One of the great challenges derives from the great number of sensor designs and the variety of possible threats. Origins for a vulnerability may reside in a sensor's

physical construction to an operating system's handling of sensor data. The plethora of sensor designs may all react differently to physical stimuli (*e.g.,* light or sound). Attacker goals can vary from disrupting sensor availability to discerning sensitive information. Thus, investigating and mitigating each of these individual vulnerabilities in a piecemeal approach is too costly for manufacturers. Thus, this thesis focuses on discovering common components or methods shared by classes rather than individual vulnerabilities, then using established concepts to systemically mitigate these classes of vulnerabilities.

## 1.1 Background and Motivation

**Sensor Heterogeneity:** Protecting sensor data is crucial because of the importance of sensor data in a wide variety of application spaces, but it is challenging for the same reason. There are more than 370 types of sensors on record [154] that rely on dozens of conversion phenomena for measurement [155]. Despite the variety, these sensors do share common components and methods to accurately convert their targeted physical quantities. Vulnerabilities often use one of these common components or methods, thus understanding these commonalities provides a foothold for general mitigation design.

**Physical Signals Affecting Sensor Output:** Sensors are by definition sensitive to at least physical quantity (*i.e.,* what they are designed to sense), but sensor output is often subject to other phenomena such as a microphone's output being subject to light. Such undefined or overlooked relationships between physical stimuli and sensor output can introduce vulnerabilities. Mitigation design must account for all sources of information embedded in sensor data.

**A Need for Systemic Mitigation:** Addressing vulnerabilities individually is an inefficient approach that is not appropriate for these physically-based vulnerabilities.

To provide an analogy, focusing on patching each individual vulnerability is akin to patching individual buffer overflow vulnerabilities; the more effective strategy has proven to be to design mitigations that the programmers did not have to consider in design such as address space randomization and stack canaries. Sensor systems need similar design changes that design the problem away. This need is heightened by how many of these vulnerabilities are best addressed in hardware, meaning that mitigations can rarely be applied retroactively via a software update.

## 1.2 Contributions

This dissertation divides the broad threat of physical vulnerabilities of sensors into two large groups and introduces terms for each: (1) transduction and (2) oversensing vulnerabilities. Transduction vulnerabilities allow an adversary to *write* extra information into sensor system output using physical signals such as light, sound, etc. Conversely, oversensing vulnerabilities allow an adversary to *read* extra information, often placed by interaction with a physical signal, in sensor system output. Contributions of this work span both transduction and oversensing vulnerabilities. Specifically, contributions include:

1. *Categorization and Analysis:* a high-level overview and analysis of transduction and oversensing vulnerabilities. This overview includes a detailed taxonomy of transduction vulnerabilities and categorization of common oversensing vulnerabilities in smartphone operating systems. This categorization and systematization provide a basis for more general mitigation design through clear dictation of common vulnerability components. Furthermore, this contribution provides common terms and language to unify vocabulary across the field.

2. *Mitigation Design Patterns:* several designs to mitigate several vulnerabilities across sensors, attack modalities or goals, operating systems, and more. This

dissertation includes a theoretical explanation of why each of these mitigations can be effective by relying on established principles and a detailed evaluation of many of these mitigations.

3. *Detailed Case Studies:* in-depth analysis of transduction vulnerability on hard disk drives and an oversensing vulnerability in smartphone motion sensors. These studies explain why each of these vulnerabilities can lead to an attack and how to apply the mitigation design patterns to specific problems.

## 1.3 Thesis Outline

The contributions of this work are roughly divided between transduction and oversensing vulnerabilities. Each has a chapter devoted to the study of the type of vulnerability (*e.g.,* transduction or oversensing) including attack categorization and mitigation design patterns, then a chapter detailing a particular attack and mitigation analysis for that attack. Specifically, the chapters of this work are:

1. **Background and Related Work:** Chapter 2 provides an overview of sensing, signal processing, and transduction and oversensing vulnerabilities. Included is a general model for component-level sensor design and common properties for sensing that can lead to vulnerabilities. Additionally, it provides related work to all areas discussed in this dissertation.

2. **Transduction Attack Model:** Chapter 3 describes a mathematically-based model to describe transduction attacks and mitigations. It systematizes past work according to this model and shows how many previously suggested mitigations map to multiple transduction attacks. These existing mitigations collectively serve as general defense patterns for most vulnerabilities.

3. **Blue Note:** Chapter 4 provides a transduction attack case study in using

acoustic waves to disrupt hard disk drive operation. Included is an investigation into the physical causality and its effects on the reliant operating system as well as physical and software mitigations.

4. **OA-Sys:** Chapter 5 describes oversensing and categories of oversensing, particularly pertaining to smartphones. Then it provides a preliminary permission system for smartphone operating systems to prevent oversensing vulnerabilities in smartphones.

5. **Touchtone Eavesdropping:** Chapter 6 provides a detailed oversensing case study in using smartphone motion sensor access to eavesdrop on numerical user input. It investigates the physical causalities and then provides signal processing-based mitigations that could be generally applicable to oversensing vulnerabilities.

6. **Discussion:** Chapter 7 discusses the conceptual similarities between the work presented elsewhere in the thesis, such as how mitigations for oversensing could relate to mitigations for transduction vulnerabilities and vice versa.

7. **Conclusion:** Chapter 8 summarizes the contributions and impact of this work.

# CHAPTER II

# Background and Related Work

This dissertation analyzes physics-based attacks on sensor systems with the ultimate goal of enabling mitigation strategies that prevent or detect different attacks across multiple sensor types. While of concern since at least the 1960s, in the past ten years there has been an increase in research focused on these attacks that exploit how physical signals (*i.e.,* acoustic waves, radio waves, light) interact with sensor systems. Applications have ranged from safety-critical systems (*i.e.,* airplanes) to Internet-of-Things (IoT) devices in the home. Designing mitigations for such attacks is difficult due to the high variation in targeted sensor systems, attacker goals, and attack methods. This variation often leads to attack-specific and sensor-specific mitigation design. However, deploying specific mitigations is often impractical as software updates cannot address hardware changes or interact with many deployed devices.

This dissertation groups these physics-based sensor vulnerabilities into two non-exclusive categories.

**Transduction attacks** manipulate sensor system output using a physical signal to *write* extra information into the data. Transduction attack research is related to the established field of electromagnetic injection on devices like smart cards and cryptographic hardware, applying similar concepts to different signal modalities and sensors. The work in this dissertation extends foundational transduction attacks,

such as Rocking Drones [135] in 2015, by providing a description and categorization methodology in the Transduction Attack Model.

**Oversensing attacks** *read* sensitive information in sensor system output, often an (unwanted) byproduct of physical signals with the sensor. Oversensing attacks fall under the established field of side-channel attacks, including work such as Differential Power Analysis [71], focusing on sensors and sensor data. This dissertation extends previous research that would fall into this Oversensing attack categorization, such as those that use accelerometer or gyroscope readings in smartphones to infer sensitive data, by providing preliminary defense patterns that could apply to many of these attacks.

Understanding these vulnerabilities requires an understanding of sensor construction, signal processing, and other related principles. This section provides a brief background into (1) sensor construction and (2) signal processing followed by related work for (3) transduction and (4) oversensing research.

## 2.1 Sensors

A sensor is defined as a device that outputs usable measurement in response to a specific measurand [49]. An ideal sensor would provide perfect data on the measured phenomenon but in reality sensors *approximate* reality. This approximation provides a gap for vulnerability. Despite a great variety in sensor design, sensors often share components and properties that facilitate general vulnerabilities. This promotes the idea that sensors share commonalities. These commonalities can be used as a basis for general mitigation strategies for both oversensing and transduction vulnerabilities.

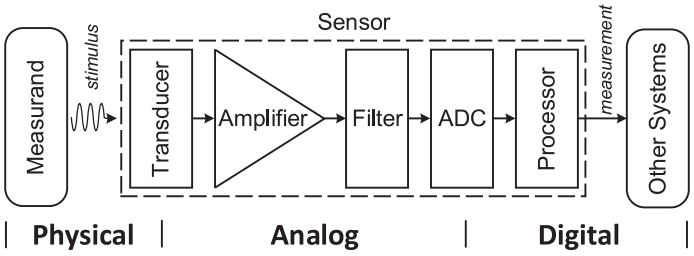



Figure 2.1: A general signal conditioning chain of sensors. Signals flow from left to right through each component and transform from the physical stimulus (input) to an analog intermediate and finally to a digital representation (output). Depending on the specific design, variations to this schematic may include multiple amplifiers or filters, no filters, filters before the transducer (e.g., CMOS) or amplifier, other circuits (e.g., comparators), etc.

### 2.1.1 Sensor Construction

Figure 2.1 shows a general component level model for sensor construction. In this model a sensor is represented as an interconnection of essential electronic components. Sensors transform a physical stimulus (input) to an analog intermediate and finally to a digital representation (output). Components include:

**Stimulus and Measurand:** The measurand is a quantity that a sensor intends to measure, and a stimulus is a physical signal involved in measuring the measurand. For example, an accelerometer's measurand and stimulus are acceleration and force, respectively. A thermocouple's measurand and stimulus are temperature and heat. Despite the wide variety of stimuli, sensors generally fall into one of two patterns to capture input:

1. *Passive sensors* passively accept physical stimuli and do not emit external stimuli. For example, microphones are passive sensors that capture sound from the environment. The stimuli of passive sensors, e.g., light, sound, force, and chemicals, are generated by other objects in the environment or already exist.

2. *Active sensors* emit physical stimuli to an environment and actively measure the response after the stimuli's interaction with the environment. For exam-

ple, ultrasonic sensors/lidars measure the distance to objects by emitting ultrasound/lasers and receiving the reflection. Active sensors are often used to measure quantities of tangible objects, such as obstacle distance, rotation speed, and liquid drop. For simplicity, we do not show the emitter in Figure 2.1.

**Transducer:** Commonly a sensor's first component, a transducer produces an analog electrical representation of the measurand by measuring a physical stimulus. Transducer heterogeneity, even for the same measurand or physical stimuli, is the primary source for sensor diversity. For example, dynamic, condenser, and piezoelectric microphones all can capture sound, but they rely on entirely different conversion phenomena [36].

**Analog Signal Processing Circuits:** Typically, a sensor must process a transducer's analog signals to reduce noise while amplifying useful information. Standard components include amplifiers to increase the signal amplitude, filters to remove noise, envelop detectors, comparators, etc.

**ADC:** An analog-to-digital converter (ADC) digitizes analog signals for digital processing, storage, etc.

Note that a sensor may not contain all components shown in Figure 2.1. For instance, some sensors do not have filters by design. Nevertheless, Figure 2.1 represents a simplified yet functionally comprehensive structure of a modern sensor.

### 2.1.2 Common Properties and Concepts for Sensor Exploits

**Sensitive to Physical Signals:** By design, sensors are sensitive to the target physical stimulus, even if the stimulus is unintended for measurement. This effect guarantees that at least one type of physical signal will affect the sensor.

**Similar Analog Signal Processing:** Analog signal processing circuits often remain similar despite transducer heterogeneity. For example, sensors commonly use ampli-

fiers and filters, even if designed to measure different phenomena. Thus, exploits on similar signal processing circuits may remain similar even on different sensors, e.g., microphones and thermocouples.

**Same Signal Modalities:** There are three signal modalities for the signal conditioning chain: physical, analog, and digital, as shown in Figure 2.1. The shared signal modalities show that the same signal properties in each modality may be exploited for attacks across various sensors.

**Chain of Blind Trust:** Sensors are essentially proxies of reality. Most sensor designs use a series of electric components to approximate the measurand. Typically, each component blindly assumes its input is valid. However, this blind trust can allow malicious signals to exploit components in the signal conditioning chain without detection.

## 2.2 Signal Processing Concepts and Terms

### 2.2.1 Basic Signal Definitions

**Amplitude:** the magnitude of a given signal at a given point in time.

**Frequency:** a measure of oscillation of a given signal.

**Bandwidth:** in a communication channel or sensor output, bandwidth is the range between the upper and lower frequency bounds the signal can contain.

**In-band vs Out-of-band:** A signal is in-band for a given sensor or application if it is within the intended range of frequencies for a signal.

**Digital vs Analog:** A signal may be digital or analog. An analog signal is continuous. Digital signals are discrete.

**Distortion:** refers to when a signal is altered unintentionally, often undesirably.

### 2.2.2 Signal and Physical Phenomenon

#### 2.2.2.1 Resonance

Resonant or natural frequencies widely exist for many mechanical and electrical components, e.g., MEMS transducers [23] and wires (antennas) [46]. A signal with a given amplitude will have a greater impact on a given component if it is near that component's resonant or natural frequency. This can be used as a method of maximally affecting a component given limited transmission capabilities. Mechanical, acoustic, and electromagnetic resonances are used in existing attacks.

#### 2.2.2.2 Aliasing

According to the Nyquist-Shannon sampling theorem [123], if the frequency of a sampled signal is higher than half of the sampling rate, the signal will be indistinguishable from signals of other frequencies [97]. For example, if the sample rate of an ADC is $F_s$, then a signal at frequency $f$ will have the same sampling result as a signal at frequency $F_s - f$. Each of these indistinguishable signals is an alias, and this improper sampling of a signal [77, 145] is called aliasing.

#### 2.2.2.3 Filtering

Filtering is the process of reducing the bandwidth of a signal path by attenuating unwanted signals at certain frequencies (stopband) and only allowing desired frequencies (passband) to reach the destination. Basic filters include:

- *Low-pass filters:* attenuates frequencies greater than the filter's cutoff $f_c$.
- *High-pass filters:* attenuates frequencies lower than the filter's cutoff $f_c$.
- *Band-pass filters:* passes frequencies within a given range, attenuating all other frequencies.

- *Band-stop or Notch filter:* attenuates frequencies within a given range, passes all other frequencies.

A perfect filter would have a flat frequency response in the passband (*e.g.,* no distortion), and maximally attenuate the stopband. However, this is impossible to achieve in practice and so designs often have trade-off characteristics such as distortion in the passband versus attenuation in the stopband. There are many different analog or digital filter designs for various applications in software and hardware. A filter's *order* is another important quality. Higher-order filters attenuate the stopband frequencies more quickly. More advanced filters can change their parameters during operation to better adjust to an application, and this is called adaptive filtering.

## 2.3 Transduction Vulnerabilities

The term "transduction vulnerability" refers to when an adversary can alter a sensor system's output [44] using a malicious physical signal and the term "transaction attack" refers to when an adversary designs a physical signal to take advantage of such a vulnerability. In particular, an attacker generates malicious physical signals that are transduced to malicious analog signals in the sensor circuitry, either explicitly through transducers or implicitly through other components in the sensor system. An attacker may generate malicious physical signals of any modality for transduction attacks, regardless of whether the signal is of the same type as the intended stimulus by design. For example, attackers can use infrared (IR) to attack lidar (which measures obstacles with IR) or RF signals to attack microphones (which measures sound).

These terms are based on the word "transducer" (Section 2.1) in a transduction attack the adversary's physical signal is transformed into a physical signal in the sensor system. As such, these attacks can be thought of as an adversary "writing" information into sensor output using physical signals.

| Application | Sensor | Attack Modalities | Papers |
|---|---|---|---|
| Automobile | Lidar | ✳ | [127, 104] |
| | Camera | ✳ | [104, 162] |
| | Radar | 📶 | [162, 24] |
| | Ultrasonic Sensor | 🔊 | [162, 158] |
| | Magnetic Encoder | 🧲 | [129, 117] |
| Drones or Smart Devices | Optical Flow Sensor | ✳ | [31] |
| | MEMS Gyroscope | 🔊 | [135, 146, 151, 91, 146] |
| | MEMS Accelerometer | 🔊 | [143, 146, 91] |
| | Microphone | 📶🔊 | [40, 66, 114, 126, 26, 165, 163, 136, 115] |
| | Touchscreen | ⚡ | [80] |
| Hard Disk | MEMS Shock Sensor | 🔊 | [19] |
| Energy | Infrared Sensor | 📶 | [121, 120] |
| Medical Devices | Pacemaker Lead | 📶 | [40] |
| | Defibrillator Lead | 📶 | [40] |
| | Drop Counter | ✳ | [101] |

✳ Visible light or infrared 📶 RF waves 🔊 Audible sound or ultrasound
🧲 Magnetic field ⚡ Electric field

Table 2.1: Systematization of transduction attacks with the simple sensor security model.

### 2.3.1 Attacks

There is significant related work that focuses on specific transduction attacks, but less work focusing on the common properties shared by transduction attacks as described later in this dissertation in Chapter III. Shown in Table 2.1 is an overview of transduction vulnerabilities. The types of malicious physical signals that have demonstrated use in transduction attacks include but are not limited to the following items.

- **Electromagnetic radiation** refers to the waves of an electromagnetic (EM) field that propagate through space. It includes radio frequency (RF) waves, infrared, (visible) light, ultraviolet, X-rays, and gamma rays. Prior work has demonstrated how to manipulate radar [162, 24], microphones [41, 66], infrared

sensors [121, 120] and pacemakers [41] with RF waves. Light has been used to manipulate LIDAR [127, 104], cameras [104, 162], microphones [138], optical flow sensors [31] and medical drop counters [101].

- **Sound** is a vibration that propagates as a wave of pressure through mediums, including gas, liquid, and solid. It includes audible sound (20 Hz to 20 kHz), ultrasound (>20 kHz), and infrasound (<20 Hz). Sound has been shown to manipulate ultrasonic distance sensors [158, 162] microelectromechanical systems (MEMS) gyroscopes and accelerometers [135, 146, 151, 91, 143], microphones [114, 126, 26, 159, 165, 115, 136] and MEMS shock sensors [19].

- **A magnetic field** is created by magnetized materials and by moving electric charges (currents) such as those used in electromagnets. Research shows how magnetic fields can manipulate magnetic encoders [129, 117].

- **An electric field** is generated by particles that bear electric charges in any form. Previous work has shown how electric fields can manipulate touchscreens [80]

### 2.3.2 Systematization and Mitigation Strategies

Many of the papers listed in Section 2.3.1 give a specific mitigation idea for whichever attack they propose; however, there have been few mitigation strategies that focus on thwarting classes of attacks (i.e., attacks using different physical signal types against different kinds of sensors), as presented in Chapter III.

The most well-known work investigating a mitigation strategy that spans multiple types of attacks and sensors is shown by Shoukry et al. [130], which focuses on active sensors. In the scheme, the sensor randomly ceases all stimulus transmission. Then, any received stimulus during this pause indicates the presence of an attack. While attackers with higher capabilities may still overcome this scheme [128], it would greatly

increase attack difficulty. Zhang et al. [168] extend this idea by modulating sensor output with a random pattern to detect attack output.

There has been little previous work looking at the whole of transduction attacks through systematization or modeling. The work provided by this dissertation instead presents a new mathematical representation of the analog signal through the sensor, then systematizes in part based on this math to show more conceptual similarities and differences between attacks and defenses. This method of systematization is aimed to make designing systemic mitigation easier.

### 2.3.3 Acoustic Attacks on Hard Drives

The detailed transduction attack case study presented in Chapter IV investigates how acoustic waves can alter and even disable hard disk drive operation. Sandahl et al. [119], Siemens [131], and Rawson and Green [111] have investigated HDD throughput loss due to acoustic interference; however, they did not consider malicious actors and did not test ultrasonic signals as seen in Chapter IV. An engineer demonstrated how yelling at HDD arrays can lead to perceptible drops in I/O throughput[1]. Ortega [98] demonstrated how hard disk drives can be interfered with by finding their resonant frequency, but focused on lower frequencies than those investigated in Chapter IV; the high-frequency ultrasonic tones presented in this work are importantly inaudible to human hearing and so is more covert. Additionally, this work discovers the root causes of *why* these phenomena happen as well as what parameters are most important in conducting such an attack and investigates mitigation strategies.

Other work has made use of HDD components' other analog features to establish covert channels. Guri et al. [52] utilized the HDD's built-in thermal sensors to receive data transmitted over the machine's heat emissions. Guri et al. [53] used the movements of a hard drive's actuator arm to generate audible emissions that were used to

[1] `https://www.youtube.com/watch?v=tDacjrSCeq4`

exfiltrate data from air-gapped machines. Since the head of a hard drive is made up of magnetic materials, the movement of the head can produce a sufficiently strong magnetic field that can be detected by a smartphone's magnetic field sensors. Matyunin et al. [82] utilized this phenomenon to build a covert channel by manipulating the movement of the head.

Much less attention has been devoted to side-channel information-leakage attacks on HDDs. Biedermann et al. [16] showed how an attacker could use a smartphone's magnetic field sensors to deduce information about a machine's operations. Previous research has demonstrated how to establish a covert channel, this work explores the effects induced by *injecting* acoustic waves into HDDs.

### 2.3.4 Acoustic Waves Affecting MEMS Motion Sensors

Acoustic waves alter the output of microelectricalmechanical systems (MEMS) accelerometers and gyroscopes due to how these sensors approximate motion (Figure 5.4). MEMS accelerometers and gyroscopes approximate the motion of a body via the motion of a small sensing mass(es) attached to capacitive springs. When the mass(es) moves, the springs create a representative voltage which is then amplified, filtered, digitized, and sent to the processor. However, while the linear or angular acceleration of the sensing mass(es) are usually accurate representations of the body's acceleration, they are not exact. The smaller mass may become affected by small acoustic vibrations via the air or through contacted surfaces [83, 143, 12]. The sensor captures this acoustic information and reports it as the acceleration of the body itself, which is unlikely the intent of the motion sensor.

## 2.4 Oversensing Vulnerabilities

An *oversensing vulnerability* is when authorized access to sensor data can provide an application with superfluous and potentially sensitive information; an *oversensing*

*attack* is when a malicious adversary obtains sensitive information using authorized sensor data. Oversensing vulnerabilities are common in smartphones and the growing Internet-of-Things [58]. To prevent blatant data misuse, researchers design and companies employ permissioning systems intending to limit each application's sensor data access [122, 48, 54, 75]. Unfortunately, granting ostensibly benign sensor access to applications may allow adversaries to acquire sensitive information hidden in sensor data. For example, attackers can use a smart-phone's power consumption to infer a victim's location [85], motion sensor data to infer keystrokes [99, 57, 21] or nearby human speech [83, 56, 166, 12], and microphones to infer keystrokes [116] or physical keys [108].

Often, traces of this sensitive information are subtly written into the sensor data by some untended or unaccounted for phenomena, i.e., nearby speech affecting accelerometer output to place traces of nearby speech in the accelerometer data. Thus, many oversensing vulnerabilities are "reading" the extra information in a sensor signal given by physical interaction. I provide a brief overview of oversensing attacks and defenses, particularly those relevant to our case study on eavesdropping via accelerometers.

### 2.4.1 Touchtone Eavesdropping Attacks

Previous work has gone into detail on oversensing vulnerabilities that infer human speech utilizing motion sensors, but do not specifically investigate touchtones as shown in Chapter VI or categorize these attacks as part of a larger set of vulnerabilities (oversensing) as discussed in Chapter V. Further, these works do not present a comprehensive analysis and evaluation of mitigations for these vulnerabilities.

Gyrophone [84] demonstrates an attack that recognizes human-spoken digits using smartphone gyroscopes by utilizing speech spectral information. Similarly, AccelWord [166] and Spearphone [12] investigate the feasibility of leveraging a smartphone's

accelerometer to capture acoustic signals for hotword detection and human spoken digits recognition, respectively. PitchIn [56] fuses across multiple unimodal sensors (e.g., only accelerometers or gyroscopes) to reconstruct intelligible human speech. However, our oversensing case study related to these attacks combines multidimensional information from different axes of sensors. More specifically, this work finds that different channels of the same sensor may serve as additional information sources to enhance the attack.

### 2.4.2 Other Vulnerabilities and Attacks

Recent research demonstrates oversensing attacks utilizing smartphone motion sensors to infer victims' location or keystrokes. Similar to the work in this paper, many of these papers use machine learning to make use of subtle information hidden in sensor data by oversensing. ACComplice [57] leverages smart-phone accelerometer to infer victim driver's driving routes as well as the starting point. Narain et al. further extended findings of ACComplice and demonstrated the feasibility of such attacks on a larger scale across ten cities [89]. (sp)Iphone [79] accesses accelerometer readings to infer typed text on nearby keyboards by observing the relative physical position and distance in between each vibration detected. Similarly, ACCessory [99] utilizes an accelerometer to infer keystrokes as the victim user types on his/her smartphone. Due to minute differences in taps, it can sufficiently infer the typed keys. Tapprints [86] further extends the findings of ACCessory by incorporating both accelerometer and gyroscopes as well as conducting larger experiments at scale with more practical use case scenarios.

PowerSpy [85] shows how an smartphone application can use valid permissions to access a phone's current power consumption because such information was originally thought to be harmless. However, this permission enables the application determine the user's location through use of machine learning.

Microphones have also been shown to be vulnerable to oversensing attacks. Ultrasound has been shown to affect microphone output [165, 163] and may be used to track user location [13]. SpiKey [109] is an attack that can determine the shape of a physical key from the time difference of clicks made by a victim using the key to unlock their door. Backes et al. use microphones to determine the text being printed by a printer from sound alone [15] and other work has shown the same on 3D printers [11]. Several works have used microphones to infer pins and keystrokes [132, 90].

### 2.4.3 Categorization and Mitigation Strategies

Chapter V provides a generalized approach to categorizing and limiting oversensing vulnerabilities versus very specific solutions for individual problems. It does so by demonstrating backward-compatible operating system changes in the sensor permissioning system that support several patterns of general use cases of sensor data. Little other work has been done in this space. Other work [86, 99] proposes very specific mitigations to oversensing by limiting sample rates and [84] applying low-pass filters. However, as later discussed in Chapter VI, this could affect the functionality of benign applications. Mitigations presented in Chapter VI are functionality aware, and so attempt to minimize the effect on benign applications. Spearphone [12] draws attention to the fact that simply applying filters does not defend against oversensing due to the complexity of signal aliasing.

# CHAPTER III

# Transduction Attack Model

This chapter, based on my past work[1], introduces the Transduction Attack Model (TAM) to detail how transduction attacks operate and systematizes existing vulnerabilities and defenses to reveal which defenses can be used more broadly across various sensors effectively. TAM is a model to designed simplify transduction attack analysis using a combination of (1) transfer mathematical models of sensor physics and (2) the abstracted methods to exploit these mathematical sensor models. This model enables a detailed description of the methods transduction attacks use such that sensor engineers can more deliberately and concisely write down security requirements and limitations concerning analog security risks to sensors. This process helps an engineer to more quickly identify the security limitations of sensor design and to have a way to debate the effectiveness of various defenses. A systematization of these attacks using the model shows that many of these attacks share common methods, even if they use different signal modalities (*e.g.,* light vs sound) on different types of sensors.

As mentioned, TAM relies on mathematical transfer functions that map to the components within a sensor's signal conditioning chain to describe both how a signal changes as it is captured and how adversaries could alter the signal arbitrarily. Each transduction attack exploits at least one transfer function. One can abstract

[1]Chen Yan, Hocheol Shin, Connor Bolton, Wenyuan Xu, Yongdae Kim, Kevin Fu. "SoK: A Minimalist Approach to Formalizing Analog Sensor Security," In *Proceedings of 41st IEEE Symposium on Security and Privacy*, 2020.

and categorize these methods to exploit the physics of sensor circuits based on two categories of basic steps to provide an easier-to-understand, higher-level description. Each step describes a method to exploit a transfer function and is an abstraction above the physical component level. Thus, different transduction attacks may share some basic steps that exploit the same vulnerability in transfer functions regardless of sensor components or sensor type. This mathematical abstraction simplifies the comparison of attacks across different sensors and signal modalities, as many attacks may be described as a chain of five or fewer steps.

Mitigation efforts can exploit these shared methods to enable defense abstraction across different sensor types. Successful attacks require each piece in the chain of steps to function correctly; thus the key to defending an attack is to mitigate at least one step in the attack chain. Consequently, designing mitigations for the most common individual attack steps defends multiple transduction attacks simultaneously. This insight results in a language to describe mitigations that is abstracted from particular sensor types and components.

## 3.1 Model Scope

### 3.1.1 Model Goal

The model presented in this chapter specifically models the analog properties of sensor security, focusing on the trustworthiness of sensor measurement under the threat of analog attacks. Trustworthiness refers to whether the measurement accurately provides the sensor's intended measurand, *e.g.,* the phenomenon the sensor is trying to measure. For example, the model should be able to describe a vulnerability that causes a lidar to detect non-existing obstacles or a microphone to report non-existing sounds. However, the model does not consider attacks that intentionally modify the measurand, which may include some transduction attacks such as those

that manipulate the actuator in a sensor system. For example, it does not consider acoustic resonance of a hard-disk drive read-write head in the head-stack assembly sensor system (such as described in Chapter IV) even though that is a transduction attack (physical signal to manipulate sensor system). The *sensor's* measurements, though manipulated, still reflect reality. Additionally, it would not cover attacks such as how attackers can use a hairdryer to heat a thermocouple such that the temperature measurements are higher than the ones of the distant environment. Similarly, the model does not consider fake fingers for fingerprint sensors [37], IR decoy flares for infrared homing missiles [70], and melamine adulteration of milk for measurement of nitrogen content [124], since all involve measurand manipulation.

### 3.1.2 Threat Model

#### 3.1.2.1 Attacker Objectives

The model considers two adversarial objectives using transduction vulnerabilities.

**Denial-of-service (DoS):** The goal is to prevent a sensor from acquiring usable measurements. For instance, a strong acoustic signal can cause unreliable, seemingly random gyroscope output if the signal's frequency is close to the gyroscope's resonant frequency. Thus, such a signal may prevent proper flight for drones relying on gyroscope output [135].

**Spoofing:** The goal is to trick a sensor into providing seemingly legitimate but erroneous measurements. For example, for an RC car controlled by a phone's gravitational orientation, malicious acoustic signals can induce false acceleration measurements and control the RC car's movement while the phone remains stationary [143].

#### 3.1.2.2 Threat Capabilities

This chapter considers adversaries with the following assumptions.

**Analog Attacks:** An adversary focuses on affecting the analog signals in a sensor and does not interfere with digital measurement processing or transmission.

**Sensor Assessment:** Adversaries cannot tamper with victim sensors but can obtain similar sensors for assessment. The adversary may reverse engineer sensor design parameters, such as operational frequencies, bandwidth, signal format, etc., and explore vulnerabilities.

**Attack Range:** Transmission power generally bounds effective attack range, but an adversary may extend the range by emitting stronger physical signals at extra costs. Thus, this work focuses on other aspects contributing to attack feasibility rather than strictly range.

## 3.2 Model Design and Attack Systematization

The Transduction Attack Model (TAM) seeks a balance between describing the salient security properties of sensors while requiring minimal additional cognitive effort by established sensor experts. To achieve this, the model adapts well-established language from the sensor and signal processing communities in transfer functions, then abstracts common methods from these transfer functions.

### 3.2.1 Model Overview

TAM addresses the challenge of how to identify and describe transduction attacks using the existing representation of transfer functions by modeling transduction attacks as a vector of malicious noise (Figure 3.1). Transfer functions [94] are a well-established concept in signal processing community to model system input and output relationships [42, 96, 28]. The model extends this approach by encapsulating the notion of malicious interference as an additional vector of noise that maps to each stage of a sensor's signal conditioning chain and related support circuitry. Additionally,

the model abstracts the most common methods of exploiting these transfer functions into a set of basic steps that are sufficient to describe vulnerabilities, attacks, and mitigations succinctly. These steps can be divided into two categories:

1. *Signal injection steps* describe key methods to inject malicious analog signals into sensors by emitting malicious physical signals and are vital to all transduction attacks.

2. *Measurement shaping steps* describe key methods to shape injected analog signals to benefit the adversary and may be optional depending on the target sensor and attacker goal.

### 3.2.2 Transfer Function Representation

This section describes the traditional transfer function model used as a base for TAM and the additions needed to represent transfer functions in the model (Figure 3.1). Table 3.1 summarizes established notation and the minimal additions to capture security properties.

#### 3.2.2.1 Traditional Sensor Model Representation

TAM models each sensor component as a transfer function, which characterizes its input-output relationship. Transfer functions are normally expressed in the Laplace domain for analysis of the dynamic response [94, 95] and in the time domain for analysis of the static response [42, 102, 133]. Note the use of the time-domain representation of transfer functions, because transduction attacks usually damage or exploit the static characteristics of sensors, e.g., accuracy and nonlinearity.

As shown in the middle of Fig. 3.1, traditional approaches characterize the $i$th sensor component as a mathematical transfer function $f_i$:

$$x_{i+1} = f_i(x_i, n_i) \tag{3.1}$$

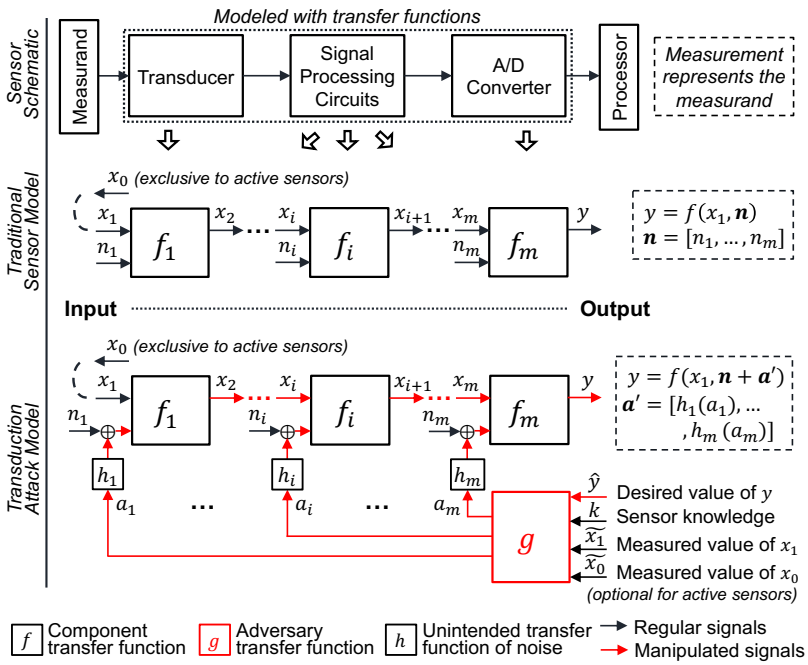



Figure 3.1: A generalized sensor schematic (top), the traditional and established way of modeling a sensor (middle), and the adaptation to model a transduction attack (bottom). The signal processing community models a sensor as cascading transfer functions. Each transfer function (e.g., $f_i$) has two inputs—the legitimate signal $x_i$ and the noise signal $n_i$. The output signal $x_{i+1}$ becomes the input to the next transfer function $f_{i+1}$. $x_1$ is the stimulus to the transducer ($f_1$), and it is either pre-existing or transformed from a probing signal $x_0$ generated by the sensor if it is an active one. The noise $n_i$ may come from the electrical circuit or the external environment. The transduction attack model consists of a simple adaptation of established practices by representing analog security exploits as a vector of intentional noise $[a_1, a_2, \ldots, a_m]$ that leads the output $y$ to an attacker-desired value of $\hat{y}$.

where the inputs consist of a legitimate signal $x_i$ and a noise signal $n_i$, and the output signal is $x_{i+1}$. Note that $x_{i+1}$ is also the input to the $(i+1)$th component. All signals are time-varying, and this figure omits the time variable $t$ in $x_i(t)$ for simplicity. The noise $n_i$ is an unwanted disturbance to $x_i$ and may come from the electrical circuit itself (electronic noise) or the external environment (coupled noise). In practice, the noise signal $n_i$ and legitimate signal $x_i$ are mixed and cannot be separated easily [148]. The conceptual model serves to emphasize that noise inputs can affect the outputs of the components and introduce errors to the measurement.

**Examples:** To provide intuition on model use, here are common examples of estab-

| Established notation from the traditional sensor modeling | |
|---|---|
| $f_i$, $f$ | The transfer function of the $i$th component and the sensor |
| $m$ | The number of components modeled as transfer functions |
| $x_i$, $\mathbf{x}$ | The legitimate input to the $i$th component and their vector |
| $x_0$ | The physical stimulus generated by active sensors |
| $n_i$, $\mathbf{n}$ | The noise input to the $i$th component and their vector |
| $y$ | The sensor measurement (at the ADC) |
| $h_i$ | The unintended transfer function of external noise |
| **Notation introduced by transduction attacks** | |
| $a_i$, $\mathbf{a}$ | The malicious physical signals emitted for the $i$th component and their vector |
| $a'_i$, $\mathbf{a}'$ | The malicious noise signals injected into the $i$th component and their vector |
| $g$ | The adversary transfer function |
| $\hat{y}$ | The sensor measurement desired by the attacker |
| $k$ | Knowledge of the sensor model |
| $\widetilde{x_1}$, $\widetilde{x_0}$ | The real measurement of $x_1$ and $x_0$ (optional for active sensors) |

Table 3.1: Transduction Attack Model Notation.

lished transfer functions when the noise is additive, which is the most widely used noise model.

1. A linear transfer function is defined as $x_{i+1} = c_0 + c_1(x_i + n_i)$, where $c_0$ is the intercept and $c_1$ is the slope. The linear relationship is expected for many components in their ideal states, such as position transducers and amplifiers [42].

2. Nonlinear relationships apply to most components in practice. Common transfer functions include [42]: logarithmic function $x_{i+1} = c_0 + c_1 \ln(x_i + n_i)$ (e.g., photodiodes), exponential function $x_{i+1} = c_1 e^{k(x_i+n_i)}$ (e.g., thermistors), and power function $x_{i+1} = c_0 + c_1(x_i + n_i) + c_2(x_i + n_i)^2$ (e.g., silicon resistive sensors). Higher-order polynomial functions may be employed for other cases.

As shown in the middle of Fig. 3.1, traditional signal processing formalizes the transfer function of a sensor as the cascading components' functions:

$$y = f_m(\cdots f_2(f_1(x_1, n_1), n_2) \cdots, n_m) \tag{3.2}$$

where the output $y$ is the sensor measurement and $m$ is the number of components. The inputs to a sensor include the physical stimulus $x_1$ at the transducer and the noise signals at each component. Let a vector of the noise signals be $\mathbf{n} = [n_1, n_2, \ldots, n_m]$ and the overall transfer function of the sensor be $f$, Equation 3.2 simplifies to:

$$y = f(x_1, \mathbf{n}) \tag{3.3}$$

#### 3.2.2.2 Adding Transduction Vulnerabilities to the Traditional Model

Although noise reduction with signal processing circuits is a common practice, sensor designers typically do not expect the intentional noise injected by transduction attacks. The goal of a transduction attack is to modify a sensor's measurement $y$ by injecting intentional noise to the legitimate physical or analog input $x_i$. Formally, I define transduction attacks as follows:

A transduction attack alters the measurement of a sensor to approach an attacker-desired value $\hat{y}$ by injecting intentional interference to the sensor. The measurement under a transduction attack can be represented as

$$y = f(x_1, \mathbf{n} + \mathbf{a}') \tag{3.4}$$

where $\mathbf{a}' = [a'_1, a'_2, \ldots, a'_m] = [h_1(a_1), h_2(a_2), \ldots, h_m(a_m)]$ is a vector of malicious noise signals injected by the attacker at each of the $m$ components, and $\mathbf{n} + \mathbf{a}' = [n_1 + a'_1, n_2 + a'_2, \ldots, n_m + a'_m]$. $h_i$ is the unintended transfer function of noise that is hidden before the $i$th component and models the propagation and coupling of external noise.

A transduction attack is modeled as shown at the bottom of Figure 3.1. The injected signal $a'_i$ is of the same type as the original noise $n_i$. The combination of the injected signal $a'_i$ and noise $n_i$ may affect the output of components and eventually

the sensor measurement. This output will be accepted because subsequent signal conditioning components blindly trust inputs. A vector of the malicious physical signals emitted by the attacker can be formalized by an *adversary transfer function* $g$:

$$\mathbf{a} = [a_1, a_2, \ldots, a_m] = g(\hat{y}, k, \widetilde{x_1}, \widetilde{x_0}) \tag{3.5}$$

where $\hat{y}$ is the desired sensor measurement, $k$ is the knowledge of the sensor model, $\widetilde{x_1}$ is the measured stimulus from the measurand, and $\widetilde{x_0}$ is the measured stimulus from the victim sensor if it is active. Independently, attackers may measure both $x_1$ and $x_0$. Knowledge of the existing stimulus $x_1$ is necessary to achieve accurate control over the sensor's measurement, and measuring the stimulus $x_0$ emitted by active sensors is optional, e.g., in spoofing sensors such as lidars [127] and ultrasonic sensors [158].

#### 3.2.2.3 Discussion

Without transfer functions, designing a viable physical signal for transduction attacks is empirical and requires trial and error. With the transfer functions, I envision that the simple sensor security model facilitates the design of malicious physical signals $\mathbf{a}$ that can modify the sensor measurements to a given value and quantify the effectiveness. In particular, given the desired sensor measurement $\hat{y}$, an attacker can derive $\mathbf{a}$ by mathematically solving an optimization problem[2]. The model can quantify the effectiveness of an attack by the expected error, i.e., $e = |\widetilde{y} - \hat{y}|$, where $\widetilde{y}$ is the real sensor measurement. An attacker's goal is to minimize the expected error $e$ by optimizing the emitted physical signals $\mathbf{a}$. Conversely, sensor designers tasked with threat mitigation may consider methods to attenuate undesirable signals

[2]An attacker may acquire the transfer functions from public datasheets or by conducting sensor assessment. If accurate modeling of a transfer function is challenging, e.g., the unintended EM coupling effect, approximate solutions are applicable. Though inaccurate transfer functions may degrade the model, they seldom impair the effectiveness of attacks, especially when an attacker does not demand a highly accurate control over the sensor output.

injected through each transducer, including unintended transducers. In addition, designers should validate that real components match their theoretical behavior, even for abnormal input.

To construct a transduction attack, the simple sensor security model indicates two categories of basic steps.

1. The signal injection steps involve determining the necessary characteristics of the malicious physical signals that can inject noise to the signal conditioning path efficiently, including the injection point and signal parameters.

2. The measurement shaping steps consider how an attacker can construct physical signals such that the injected analog signals ("injected signals" hereafter) can shape the sensor measurement to desired values.

The following elaborates on the two categories of steps.

### 3.2.3 Signal Injection Steps

To successfully and efficiently inject malicious analog signals into a sensor, an attacker must consider (1) the injection point, i.e., before or after the intended transducer, and (2) which signal types, amplitudes, and frequencies of the emitted physical signals are most efficient.

#### 3.2.3.1 Injection Point and Signal Type

Since transducers are by design the only component that intends to accept external physical inputs, the model divides the injection point into pre-transducer (at or before the intended transducer) and post-transducer (after the intended transducer). The injection point determines the required signal type.

**Pre-Transducer:** Attackers may exploit how transducers are designed to be sensitive to at least one type of physical signal (Section 2.1). Most existing transduction

attacks exploit injection points at or before the intended transducers. Intuitively, the malicious physical signal $a_1$ can be the same type of signal as the legitimate stimulus $x_1$ so that they are inherently noise inputs without conversion. For example, Petit et al. [104] and Shin et al. [127] succeeded in attacking lidars with light. Yan et al. [162] managed to jam and spoof ultrasonic sensors with ultrasound. Attackers can also use acoustic waves to attack sensors that measure movement vibrations, e.g., MEMS gyroscopes [135, 151, 146] and accelerometers [143, 146].

**Post-Transducer:** Components after the intended transducer may act as unintended transducers, converting noise or an attacker's physical signal $a_i$ into an electrical signal inside the sensor despite not being designed to do so. For example, wires in a circuit may act as unintended antennas by converting electromagnetic waves into analog electrical signals in the sensor via inductive or capacitive coupling [64]. Since the injected signal $a_i'$ is usually weak after conversion, injection points before the pre-amplifier are typically preferred so that the injected signal may have a more dominant effect on the output than the legitimate input $x_i$.

Rasmussen et al. [110] first suggested adversarial usage of wires being unintended antennas, but as a possible pathway for wormhole attack [61] rather than in the context of transduction attacks. Foo Kune et al. [40] first exploited this effect for a transduction attack by injecting a voice signal into a Bluetooth headset. This attack employed intentional electromagnetic interference (EMI) on the conducting wire between the microphone and amplifier. Later work employed electromagnetic coupling from outside [66] and inside [38] a smartphone to inject malicious signals to its microphone, and compromised actuator control signals [121, 120, 152].

Attackers must consider how injection using a wave or field may affect several sensor components simultaneously, as it may complicate the attack in terms of the needed measurement shaping steps. For example, sound can vibrate all sensor components besides the transducer, and electromagnetic injection may induce currents at

many parts of a circuit. However, as we will discuss later, an attacker with knowledge of the sensor may craft the malicious physical signal, such as choosing specific frequencies, to affect targeted components more than others.

#### 3.2.3.2 Efficient Injection and Signal Frequency

The effectiveness and efficiency of signal injection depend on the physical signal's form, especially the amplitude and frequency.

**Amplitude:** An injection is likely to be more effective with physical signals of higher amplitudes. Shorter attack distances or increased transmission power may result in higher physical signal amplitudes at the target sensor due to the power-attenuation law for physical signals, e.g., sound and electromagnetic waves [92]. However, in practice, one cannot easily increase the power indefinitely. For example, a high-power radio frequency signal, laser, or sound may cause damage to human bodies [22]. Inaudible ultrasound at high power may create audible byproducts [163, 159] due to nonlinear acoustics [17, 55], which makes inaudible attacks audible.

**Frequency:** An attacker may increase the efficiency of injection using the resonant frequencies of the target injection point. A standard method to find the resonant frequency is to conduct a frequency sweep and search for the highest amplitude response [143, 40, 135, 146, 19]. In addition to using a resonant frequency, an attacker may choose specific frequencies to satisfy the special requirement of attacks, e.g., to be hidden. For example, attacks that employ ultrasound ($> 20$ kHz) are inaudible to humans. Similarly, injecting infrared into cameras makes the attack invisible. Depending on the frequency, signals can be divided into two categories: in-band and out-of-band.

#### 3.2.3.3 In-Band and Out-of-Band Signals

Typically, injecting an out-of-band signal can have two problems. (1) Out-of-band signals do not directly interfere with ordinary inputs because they are at different frequencies. For example, one cannot discern ultrasonic noises in audible sounds. (2) Out-of-band signals are often attenuated by filters designed to remove signals outside the expected frequency band. For an injected out-of-band signal to affect the measurement, an attacker will require measurement shaping steps to convert the injected signals to the expected frequency range, i.e., in-band.

### 3.2.4 Measurement Shaping Steps

Signal injection alone may be insufficient to meet all attacker goals; thus, an attacker may have to take additional steps to shape the injected analog signals to DoS or spoof the sensor. A common example is how an attacker may use steps to shape out-of-band signals to in-band for sensor spoofing. To shape the signal between transduction and digitization, an attacker may exploit specific properties of sensor components by properly designing the malicious physical signals. Measurement shaping steps are defined as these exploits of sensor component properties that shape the injected signals. The following elaborates on how one can group several common measurement shaping steps across a wide variety of sensor types and existing studies, and show how these may be mathematically abstracted to fit into the model.

#### 3.2.4.1 Saturation (Sat.)

Saturation is a common phenomenon in analog electronics referring to how a quantity cannot exceed an upper or lower bound [42].

For example, as shown in Equation 3.6, an amplifier may become saturated when the input is beyond a threshold. In this case, the input increment no longer proportionally increases the output, which leads to clipping:

$$f_i(x_i, n_i + a'_i) = \begin{cases} c_1\mathcal{A}(x_i, n_i + a'_i) & \text{if } \mathcal{A}(x_i, n_i + a'_i) \leq k \\ const & \text{if } \mathcal{A}(x_i, n_i + a'_i) > k \end{cases} \tag{3.6}$$

where $\mathcal{A}(x_i, n_i + a'_i)$ denotes the intensity of combined $x_i$ and $n_i + a'_i$, $c_1$ is the amplification factor, and $k$ is the saturation point. For amplifiers, the clipping voltage is normally determined by the power supply. Similar effects can happen to other components such as transducers, ADCs, etc. Saturation is undesirable, and sensors are designed to operate below the saturation point. However, an attacker can intentionally saturate a component by injecting a strong interference $a'_i$. In this way, the adversary may mask legitimate input and DoS a sensor [101, 127, 104, 162], or let in a DC signal component for spoofing when there was none [143].

#### 3.2.4.2 Intermodulation Distortion (IMD)

IMD [156] can occur when a signal with two or more frequencies passes through a nonlinear component. For example, amplifiers, diodes, and transducers are generally known to be nonlinear; even ADCs show some level of inherent nonlinearity due to internal amplifiers [45]. IMD forms cross-products at new frequencies that are not present in the input signals. Specifically, the output signals include the sum and difference of the input frequencies. For example, consider a nonlinear transfer function in a simple 2nd-order power series:

$$x_{i+1} = c_0 + c_1(x_i + n_i + a'_i) + c_2(x_i + n_i + a'_i)^2 \tag{3.7}$$

Suppose the mixed signals of $x_i + n_i + a'_i$ contain two frequencies, $f_1$ and $f_2$ ($f_1 > f_2$). The output $x_{i+1}$ of this nonlinear transfer function contains frequencies at $f_1$, $f_2$, $f_1 - f_2$, $f_1 + f_2$, $2f_1$, $2f_2$, and a constant offset. Note that $f_1 - f_2$ may be below the original frequencies. An attacker can exploit IMD to convert malicious out-of-band

signals to in-band, e.g., demodulating amplitude modulated (AM) signals [134]. Note that in radio receivers, IMD is the desired effect by design for down-converting signals to intermediate frequencies, such as in frequency mixers [60].

An attacker may exploit any sensor component that is characterized by a nonlinear transfer function for IMD. For example, Foo Kune et al. [40] utilized amplifier nonlinearity to recover a baseband voice from the injected electrical signals coupled from an RF carrier. Similarly, other studies [165, 115, 136] managed to recover voice commands from ultrasound by exploiting nonlinear microphones.

#### 3.2.4.3 Envelope Detection (Env.)

Diodes and capacitors are essential in many circuits, especially for electrostatic discharge protection [141, 113]. However, they can also act as simple envelope detectors that demodulate AM signals. Foo Kune et al. [40] found several capacitor-diode pairs before a microphone's amplifier that could demodulate the injected signals.

#### 3.2.4.4 Aliasing (Ali.)

An adversary may exploit aliasing to convert the malicious out-of-band signals to in-band frequencies after the ADC. For example, Trippel et al. [143] and Tu et al. [146] managed to control the output of ADCs in MEMS accelerometers or gyroscopes by tuning the amplitude, frequency, or phase of the injected signals. Foo Kune et al. [40] managed to demodulate the injected signals after ADC by setting the carrier frequency equal to the sample rate.

#### 3.2.4.5 Filtering (Fil.)

Ideally, filters before an ADC should remove all out-of-band signals and prevent aliasing. Yet in practice, it is difficult to manufacture a filter that can remove all out-of-band frequencies based on the designed cut-off frequencies while passing all in-

| Sensor | | Exploited Component | | | | | | Signal Injection | | | Measurement Shaping | | | | | Outcome | | Paper |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Application | Type | C. | Trans. | Wire | Amp. | Filter | ADC | Point | Type | Freq. | Sat. | IMD | Fil. | Env. | Ali. | DoS | Spoof | |
| Automobile | Lidar | A | ● | ○ | ◐ | ○ | ○ | Pre | ✺ | In | ● | ○ | ○ | ○ | ○ | ● | ○ | [127] |
| | | | | | | | | | | | ○ | ○ | ○ | ○ | ○ | ○ | ● | [127, 104] |
| | Camera | P | ● | ○ | ◐ | ○ | ○ | Pre | ✺ | In | ● | ○ | ○ | ○ | ○ | ● | ○ | [104, 162] |
| | Radar | A | ● | ○ | ◐ | ○ | ○ | Pre | 🛜 | In | ◐ | ○ | ○ | ○ | ○ | ● | ○ | [162] |
| | | | | | | | | | | | ○ | ○ | ○ | ○ | ○ | ○ | ● | [162, 24] |
| | Ultrasonic Sensor | A | ● | ○ | ◐ | ○ | ○ | Pre | 🔊 | In | ◐ | ○ | ○ | ○ | ○ | ● | ○ | [162, 158] |
| | | | | | | | | | | | ○ | ○ | ○ | ○ | ○ | ○ | ● | [162, 158] |
| | Magnetic Encoder | A | ● | ○ | ○ | ○ | ○ | Pre | 🧲 | In | ○ | ○ | ○ | ○ | ○ | ● | ● | [129, 117] |
| Drones or Smart Devices | Optical Flow Sensor | P | ● | ○ | ○ | ○ | ○ | Pre | ✺ | In | ○ | ○ | ○ | ○ | ○ | ○ | ● | [31] |
| | MEMS Gyroscope | P | ● | ○ | ○ | ● | ● | Pre | 🔊 | Out | ○ | ○ | ● | ○ | ○ | ● | ○ | [135, 146] |
| | | | | | | | | | | | ○ | ○ | ● | ○ | ● | ○ | ● | [151, 91, 146] |
| | MEMS Accelerometer | P | ● | ○ | ● | ● | ● | Pre | 🔊 | Out | ○ | ○ | ● | ○ | ○ | ● | ○ | [143],[146] |
| | | | | | | | | | | | ○ | ○ | ● | ○ | ● | ○ | ● | [143],[146, 91] |
| | | | | | | | | | | | ● | ○ | ● | ○ | ○ | ○ | ● | [143] |
| | Microphone | P | ● | ● | ● | ● | ● | Post | 🛜 | Out | ○ | ○ | ● | ● | ○ | ○ | ● | [40] |
| | | | | | | | | | | | ○ | ● | ● | ○ | ○ | ○ | ● | [40, 66] |
| | | | | | | | | | | | ○ | ○ | ● | ○ | ● | ○ | ● | [40] |
| | | | | | | | | Pre | 🔊 | Out | ◐ | ● | ○ | ○ | ○ | ● | ○ | [114, 126, 26] |
| | | | | | | | | | | | ○ | ● | ○ | ○ | ○ | ○ | ● | [165, 163, 136, 115] |
| | Touchscreen | A | ● | ○ | ○ | ◐ | ○ | Pre | ϟ | N/A | ○ | ○ | ◐ | ○ | ○ | ● | ● | [80] |
| Hard Disk | MEMS Shock Sensor | P | ● | ○ | ◐ | ● | ○ | Pre | 🔊 | Out | ◐ | ○ | ● | ○ | ○ | ○ | ● | [19] |
| Energy | Infrared Sensor | P | ○ | ● | ○ | ◐ | ● | Post | 🛜 | Out | ● | ○ | ◐ | ○ | ○ | ◐ | ◐ | [121, 120] |
| Medical Devices | Pacemaker Lead / Defibrillator Lead | P | ○ | ● | ○ | ○ | ○ | Post | 🛜 | In | ○ | ○ | ○ | ○ | ○ | ○ | ● | [40] |
| | Drop Counter | A | ● | ○ | ◐ | ○ | ○ | Pre | ✺ | In | ● | ○ | ○ | ○ | ○ | ● | ● | [101] |

✺ Visible light or infrared 🛜 RF waves 🔊 Audible sound or ultrasound 🧲 Magnetic field ϟ Electric field ● Applicable ◐ Probable ○ Not applicable

**C.** Category **A** Active sensor **P** Passive sensor **Pre** Pre-transducer **Post** Post-transducer **In** In-band **Out** Out-of-band **N/A** Not available

Table 3.2: Systematization of Transduction Attacks.

band frequencies. Instead, there is a range of frequencies around the cut-off frequency that is attenuated but not completely removed. For example, lower-order filters have a wider transition range where signals remain only partially attenuated [60]. An attacker could exploit this property to design signals that pass the filter, but the following components cannot handle properly. Trippel et al. [143] found many low-pass filters in MEMS accelerometers that show large transition ranges, and thus these filters do not sufficiently attenuate higher-frequency signals that affect sensor output. From the defense point of view, note that though high-order filters may hinder the exploit of aliasing, they can also mask the trace of high-frequency malicious signals if the exploit happens before these filters, e.g., IMD or saturation.

### 3.2.5 Constructing a Transduction Attack

An attacker can build a transduction attack based on the chaining explorations of various signal injection steps and measurement shaping steps, which are determined by the target sensor and the desired attack outcome. We summarize the basic steps that have been exploited by existing transduction attacks in Table 3.2. In the design of a transduction attack, an attacker can construct malicious physical signals by examining the possible steps as malicious signals go through the signal conditioning chain of a sensor. To this end, explicit knowledge of the sensor components and their transfer functions, i.e., the simple sensor security model, is necessary.

## 3.3 Defense Systematization and Patterns

Defense mechanisms against transduction attacks can be divided into two broad categories: detection mechanisms to discern attacks and prevention methods to ensure proper or trustworthy measurement during attacks. This section analyzes and systematizes existing defenses (Table 3.3) using these categories in addition to previously described transfer functions and injection/measurement shaping steps. Effective mitigations found during systematization can be utilized as general design patterns.

### 3.3.1 Detection Methods

Detection methods do not ensure proper sensor output if attacked. However, they can be a starting point of more robust system-wide defenses, e.g., a trigger for activating other preventive measures or a fail-safe mode. Detection methods do not modify any existing transfer functions, but can be modeled as an additional binary function, $\chi$, as shown in Equation 3.8, which takes $\mathbf{x}' = [x_1, x_2, \ldots, x_m, y(= x_{m+1})]$,

| Goal | Cat. | Subcat. | Related Component | | | | | | | Injection and Shaping Steps | | | | | | | | Xfer[1] Func. | Paper |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | TX | Trans. | Wire | Amp. | Fil. | ADC | Dig.[2] | Point | Freq. | Sat. | IMD | Fil. | Env. | Ali. | | |
| Detection | Inject. | TX randomiz. | ● | ○ | ○ | ○ | ○ | ○ | ● | Both | Both | ○ | ○ | ○ | ○ | ○ | N/A | [130, 101, 158] |
| | | Verif. Actuation | ● | ○ | ○ | ○ | ○ | ○ | ● | Both | Both | ○ | ○ | ○ | ○ | ○ | | [40, 87] |
| | | Detect OOB Sig. | ○ | ● | ○ | ○ | ○ | ○ | ● | Pre | Out | ○ | ○ | ○ | ○ | ○ | | [151, 19] |
| | Shap. | Saturation | ○ | ● | ○ | ● | ○ | ● | ○ | N/A | N/A | ● | ○ | ○ | ○ | ○ | | [101, 162, 127] |
| | | IMD Features | ○ | ○ | ○ | ○ | ○ | ○ | ● | N/A | N/A | ○ | ● | ○ | ○ | ○ | | [165, 115] |
| Prevention | Rand. | TX Randomiz. | ● | ○ | ○ | ○ | ○ | ○ | ● | Both | Both | ○ | ○ | ○ | ○ | ○ | **P1,P2** | [158, 104, 127] |
| | | RX Randomiz. | ○ | ○ | ○ | ○ | ○ | ● | ● | N/A | N/A | ○ | ○ | ○ | ○ | ● | **P1** | [143, 146, 162, 31] |
| | Shielding | Physical Barrier | ○ | ● | ● | ● | ● | ● | ● | Both | Both | ○ | ○ | ○ | ○ | ○ | **P2** | [40, 135, 151, 101, 19] |
| | | Spatial ASR[3] | ○ | ● | ○ | ○ | ○ | ○ | ○ | Pre | In | ○ | ○ | ○ | ○ | ○ | | [127] |
| | | Temporal ASR | ○ | ● | ○ | ○ | ○ | ○ | ○ | Pre | In | ○ | ○ | ○ | ○ | ○ | **P1** | [104] |
| | | Spectral ASR | ○ | ● | ○ | ○ | ○ | ○ | ○ | Pre | Out | ○ | ○ | ○ | ○ | ○ | **P1,P2** | [104] |
| | Filt. | LPF/BPF/HPF | ○ | ○ | ○ | ○ | ● | ○ | ○ | N/A | N/A | ○ | ● | ● | ○ | ● | **P1,P2** | [40, 165, 143] |
| | | Adaptive Filt. | ○ | ○ | ● | ○ | ○ | ○ | ● | Both | Both | ○ | ○ | ○ | ○ | ○ | **P2** | [40, 151, 165, 19, 135] |
| | | Out-of-phase Samp. | ○ | ○ | ○ | ○ | ○ | ● | ○ | N/A | N/A | ○ | ○ | ○ | ○ | ● | **P1** | [143] |
| | Fusion | Spatial Fusion | ● | ● | ● | ● | ● | ● | ● | Steps differ case by case | | | | | | | **P3** | [151, 162, 127, 158, 19] |
| | | Spectral Fusion | ● | ● | ○ | ○ | ○ | ○ | ● | | | | | | | | **P1,P3** | [104] |
| | | Temporal Fusion | ○ | ○ | ○ | ○ | ○ | ○ | ● | | | | | | | | **P1** | [158, 31, 104] |
| | Comp. | Quality Improv. | ● | ● | ● | ● | ● | ● | ○ | | | | | | | | **P1** | [143, 135, 151] |

[1] Denotes the three xfer func. models of Section 3.3.2. [2] Digital Backend [3] Attack Surface Reduction ● Applicable ○ Not applicable

Table 3.3: Systematization of mitigations for transduction attacks.

$\mathbf{n}$, and $\mathbf{a}'$ as inputs (See Table 3.1).

$$\chi\left(\mathbf{x}', \mathbf{n} + \mathbf{a}'\right) = \begin{cases} 1 & \text{if it is an attack} \\ 0 & \text{if it is not an attack} \end{cases} \tag{3.8}$$

Some detection methods may also include an active sensor's generated stimulus $x_0$ in addition to the basic parameters of $\mathbf{x}'$, $\mathbf{n}$, and $\mathbf{a}'$. This simple model represents a detection scheme based on the status (intermediate in/output signals and noise levels) of each component. Detection methods may be further categorized by which of an attack's steps the method defends.

#### 3.3.1.1 Detecting Signal Injection Steps

Currently, three types of methods detect the exploitation of injection steps.

**Detection by TX Randomization:** Adopting randomness to an active sensor's transmission (TX) can often assist in detecting transduction attacks. Sensors with random transmission schemes may detect attacks when an adversary should be unable to predict the random pattern. Often, the sensor transmits a randomized physical stimulus and checks that the randomness is intact in the received stimulus. Any major changes to the signal indicate the presence of an attack. To include signal transmitters, Equation 3.8 has to be expanded to embrace the transmitted stimulus, such that $\chi\left(\mathbf{x}', \mathbf{n} + \mathbf{a}'; r\left(x_0\right)\right)$. Here, $x_0$ represents the physical stimulus generated by active sensors originally, and $r\left(\cdot\right)$ denotes the randomizing function. A prime example of a TX Randomization Detection scheme is a time-based randomization scheme first shown by Shoukry et al. [130]. In the scheme, the sensor randomly ceases all stimulus transmission. Then, any received stimulus during this pause indicates the presence of an attack. While attackers with higher capabilities may still overcome this scheme [128], it would greatly increase attack difficulty. Other work has proposed the same time-based randomization scheme [101] for medical infusion pumps. In the same line, Xu et al. [158] proposed physical shift authentication to detect transduction attacks on automotive ultrasonic sensors by randomizing several waveform parameters.

**Verifying Actuation:** A system with both sensors and actuators may detect attacks by probing the surroundings periodically. Essentially, the actuator delivers a probing signal to the surrounding environment and compares the measured response with an expected response. A vast difference between expected and measured signals indicates the presence of an attack. The transfer-function notation is similar to that of detection by TX randomization, in $\chi\left(\mathbf{x}', \mathbf{n} + \mathbf{a}'; t\left(x_0\right)\right)$ form, where $x_0$ indicates the

preset probing signal and $t(\cdot)$ denotes the function that converts the probing signal to the environmental response. Prior work has suggested various forms of verifying actuation detection schemes. Foo Kune et al. [40] proposed the adoption of a cardiac probe that cross-checks whether cardiac signal readings coincide with the expected values after investigative actuation to the cardiac tissue. In the same line, Muniraj et al. [87] suggested an active detection method, which detects the attack based on "judiciously designed excitation signals" superimposed on the control signal.

**Detecting Out-of-band Signals:** Defenders may come up with an additional receiver that detects targeted out-of-band signals. For example, as suggested by prior work [151, 19], adopting an additional microphone can detect resonating sound against MEMS sensors. Since sensors require only in-band stimuli to function, detecting out-of-band signals does not affect sensors' functionality, and there can be many other variations according to the targeted out-of-band signals.

#### 3.3.1.2 Detecting Measurement Shaping Steps

Previous work describes how to detect certain measurement shaping steps.

**Saturation Detection:** Several previous studies suggest saturation detection as a defense [101, 162, 127]. Saturating a component leaves the component in an abnormal state, that may be easily detectable with hardware or software support. The saturation detection function, $\chi_{sat}$, monitors if the input of a vulnerable component exceeds a threshold, e.g., a voltage level. Assuming the $i$th component is saturated, $\chi_{sat}$ can be modeled as a logical inequality as shown in Equation 3.9, where $\mathcal{A}(x_i, n_i + a'_i)$ denotes the intensity of combined $x_i$ and $n_i + a'_i$, and $\varepsilon$ is the saturation threshold.

$$\chi_{sat}(\mathbf{x}', \mathbf{n} + \mathbf{a}') = \mathcal{A}(x_i, n_i + a'_i) > \varepsilon \tag{3.9}$$

**Detecting IMD Features:** Studies have shown that attacks exploiting intermod-

ulation distortion (IMD) for signal demodulation could leave identifying features in analog signals. Zhang et al. [165] proposed to search for features of IMD demodulated voice signals by the intensity at high frequencies (500 Hz to 1 kHz), and Roy et al. [115] suggested to search for signal correlation in the sub-50 Hz band. These features are introduced by IMD and may not be easily erased.

### 3.3.2 Prevention Methods

Prevention methods ensure proper sensor output even in the presence of a transduction attack, generally by attenuating malicious signals either inside or outside of the sensor. Prevention methods can roughly be modeled as three types:

**P1 - Component Modification:** A defender modifies an existing component to reduce an attacker's ability to exploit that function. Assuming the vulnerable $i$th component ($f_i$) is improved ($f_i'$), these defenses can be expressed as below.

$$\left|f_i'\left(x_i^{adv}, n_i + a_i'\right) - x_{i+1}\right| \ll \left|f_i\left(x_i^{adv}, n_i + a_i'\right) - x_{i+1}\right| \tag{3.10}$$

where $x_i^{adv}$ denotes the input to $f_i$ under attack and $x_{i+1}$ is the output of $f_i$ without an attack. Thus, modification ($f_i \rightarrow f_i'$) reduces an attack's effect on output (Equation 3.10).

**P2 - Component Addition:** A defender inserts a new component to reduce the effect of the attack on subsequent components. In terms of transfer function representation, this type of defenses can be represented as below.

$$|f'(x_1, \mathbf{n} + \mathbf{a}') - y| \ll |f(x_1, \mathbf{n} + \mathbf{a}') - y|, \tag{3.11}$$

where $f'$ is a new transfer function of the sensor obtained by adopting the new component, and $y$ is the sensor output without attack in Figure 3.1.

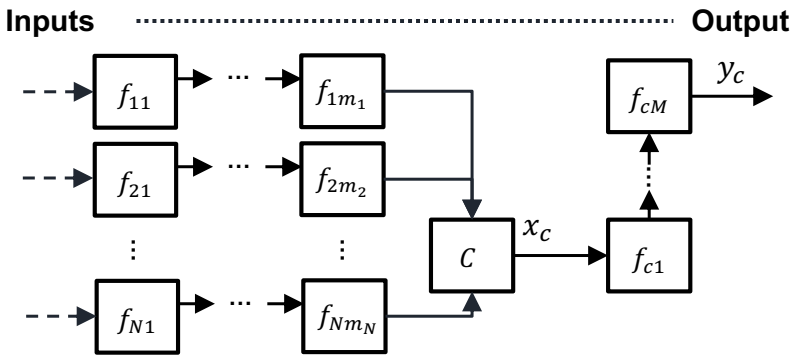



Figure 3.2: Example of fused transfer-function chains. $N$ chains with different numbers of internal components ($m_1$ through $m_N$) are synthesized together by the combining function $\mathcal{C}$ to produce an intermediary output $x_c$. This intermediary output then undergoes $M$ more transfer functions ($f_{c1}$ through $f_{cM}$) to yield the final combined output $y_c$.

**P3 - Component Fusion:** A defender fuses multiple chains of components, either full chains or parts of chains, to produce a new combined output. Figure 3.2 illustrates an example of combining chains. $N$ chains of transfer functions are synthesized together to produce an intermediary output $x_c$, which is then processed by additional $M$ transfer functions to yield the final output $y_c$. Throughout the combining process, the effect of the attack becomes suppressed in the final output, $y_c$.

The remainder of this section introduces various prevention methods built on Component Modification, Component Addition, and Component Fusion.

#### 3.3.2.1 Shielding

Shielding mitigates injection steps by reducing exposure to external physical signals. It can be a simple solution to mitigate transduction attacks but occasionally required hardware changes may be inadequate [19] or too costly to implement. Shielding typically adds additional hardware and thus corresponds to Component Addition.

**Physical Barriers:** A defender may add situation-specific barriers to attenuate external physical signals, e.g., using a Faraday cage to block electromagnetic signals. However, by design some sensors must be exposed to the outer environment, and thus

physical barriers may not always be applicable. For example, a defender cannot block a lidar from lasers as the lidar needs to sense echoes of its transmitted lasers to function. Previously discussed physical barriers include shielding conducting wires [40], acoustic damping [135, 151, 19], and optical shielding [101].

**Attack Surface Reduction:** Reducing the attack surface area selectively limits transducer exposure to external physical stimuli, thus increasing injection difficulty while allowing the transducer to remain exposed to intended physical stimuli. Examination of previous work shows that these reduction methods can be classified further into spatial, temporal, and spectral categories. Spatial attack surface reduction attempts to confine transducer exposure only to the direction of the physical stimuli to measure. Spatial methods are especially relevant to sensors that take readings from their field of view piece by piece. For example, Shin et al. [127] suggested increasing the directivity of internal receivers/transmitters and removing curved reception glass in lidars. Similarly, temporal reduction limits the duration of transducer exposure, and spectral reduction limits the bandwidth of stimulus the transducer is exposed to. Petit et al. [104] proposed limiting lidar reception time (temporal reduction) and filtering out unwanted light frequencies (spectral reduction). Unlike spatial reduction, the temporal reduction would correspond to Component Modification because reducing the effective duration can be implemented without additional hardware. Likewise, Spectral reduction may also be implemented by Component Modification when reducing the bandwidth to which the transducer is exposed is feasible without adopting new hardware.

#### 3.3.2.2 Filtering

Filtering aims to attenuate malicious analog signals within the sensor without attenuating the legitimate signals. These defenses are suitable for sensors that focus only on a specific part of the analog signal, e.g., a specific frequency band. In

terms of transfer-function representation, filtering defenses correspond to Component Modification (when improving existing filters) and Component Addition (when an additional filter is adopted).

**LPF/HPF/BPF:** A defender can use low/high/band-pass filters to attenuate frequency bands containing only noise or malicious signals. In addition, they may also mitigate IMD and aliasing by blocking frequencies that possibly induce such phenomena. Previous work suggests a variety of relevant applications for these filters [40, 165, 143].

**Adaptive Filtering:** A defender may be able to use adaptive filtering to attenuate injected signals when simple low/high/band-pass filters are inapplicable. In the context of transduction attacks, adaptive filtering methods typically find some reference of a malicious signal and then use this reference to filter it out. For example, a defender may augment a microphone with an additional wire to clearly receive an attacker's electromagnetic wave signal, and then use this reference to filter the malicious analog signal on the sensing path [40]. Alternatively, differential signaling can be employed to cancel out the injected signal [40, 135]. Other work has also employed adaptive filtering in additional contexts [19, 151].

**Out-of-phase Sampling:** A defender may make an analog-to-digital converter (ADC) adopt a special sampling pattern related to the frequency to which the injected signals are confined. This strategy can mitigate attacks that exploit ADC aliasing for output control (Section 3.2.4.4) as shown by Trippel et al. [143].

#### 3.3.2.3 Randomization

Adding randomness can often mitigate attacker influence on the sensor output. Randomness can be applied to various components: transducers, ADCs, and even digital backends such as microcontrollers. This defense can be subdivided into receiver-

chain (RX) and transmitter-chain (TX) randomization based on where the randomness is applied.

**RX Randomization:** RX randomization applies randomness to RX control parameters and can effectively deal with attacks where the injected signals can only be partially controlled, e.g., the raw injected signal is fluctuated randomly instead of a controllable bias [143] or only a fraction of the injected signal is controllable [31]. Under such partially controllable cases, attackers often exploit the predictable property of a certain component, e.g., sampling points of an ADC. Thus, randomizing the exploitable property may prevent full sensor output control. These defenses correspond to Component Modification because randomness can generally be adopted without any additional components. RX randomization can mitigate ADC aliasing as suggested by prior work on accelerometers and gyroscopes [143, 146]. Randomizing the ADC's sampling intervals prevents an attacker from predicting the exact quantization timing, increasing the difficulty of inducing controllable bias to the output. However, RX randomization may have other applications. Davidson et al. [31] suggested using a random sample consensus (RANSAC) [39] to defend a transduction attack on optical flow sensors based on the Lucas-Kanade method [76]. To extract true optical flow from the corrupted transducer output, RANSAC randomly picks a subset of features to form hypotheses, then makes all other features to vote for them.

**TX Randomization:** Prevention and detection by TX randomization operate on similar principles, but prevention by TX randomization focuses more on enhancing the resiliency of sensors against attacks rather than detecting them. When randomness is added to various parameters (e.g., direction, waveform, and frequency) of the transmitted signal, the sensor itself, aware of the random pattern, can selectively concentrate only on meaningful information. In contrast, attackers unable to identify the random pattern embedded in the signal may not be as effective. TX randomization often corresponds to Component Modification, but may also correspond

to Component Addition (e.g., adding actuators for spatial randomization). It also should be noted that the modified transfer function lies not in the receiver but in the transmitter chain. TX randomization has long been utilized for military applications, especially for radars [100], and for additional applications. The physical shift authentication [158] can recover real echoes in the received signal of an ultrasonic sensor. Petit et al. and Shin et al. [104, 127] suggested random-probing lidars, which correspond to spatial randomization, to defend spoofing attacks. Additionally, Shin et al. proposed randomizing lidars' ping waveform.

#### 3.3.2.4 Improving the Quality of Components

Sensor designers may choose to improve the performance of certain components to mitigate attacker signals, though typically this increases production costs. The details of a specific component improvement, including the sensor and attack to defend, allow this class of defense to mitigate a variety of injection and measurement shaping steps. For example, Trippel et al. [143] suggested using secure amplifiers whose dynamic range is large enough to cope with the exploited saturation. Son et al. [135] proposed to redesign MEMS gyroscopes to have resonance frequencies in non-critical frequency bands. Although specific approaches were not given, designing acoustic-resonance resilient MEMS gyroscopes were proposed as a defense by both Son et al. [135] and Wang et al. [151]. All three cases belong to Component Modification because no additional component is adopted.

#### 3.3.2.5 Sensor Fusion

Defense by sensor fusion enhances resiliency against transduction attacks by utilizing output from multiple sensors. They can be divided further into (1) spatial fusion which utilizes multiple sensors, sometimes different types of sensors, (2) spectral fusion that adopts multiple redundant frequencies/wavelengths for measurement,

and (3) temporal fusion which utilizes the history of measurement. Spatial fusion corresponds to Component Fusion as combines the output of multiple transducers, i.e., multiple chains of transfer functions. It was commonly suggested as a defense by prior work [151, 162, 127, 158, 19], and can be applied to systems that can bear the cost of adopting multiple sensors. Spectral fusion can be thought of as both Component Modification (when a single transducer suffices for such multi-band operation) and Component Fusion (when multiple transducers are required). Petit et al. [104] suggested utilizing multiple wavelengths to enhance lidars' resiliency against transduction attacks. Temporal fusion is typically more affordable than spatial as it does not typically require extensive hardware modification and would be close to Component Modification. Previous work has suggested the use of sensor fusion as a mitigation design. Xu et al. [158] proposed a special filter to remove maliciously injected echoes in ultrasonic sensors by examining echo consistency over multiple pulse cycles. Davidson et al. [31] suggested weighted RANSAC with momentum to increase resiliency against spoofing attacks. It not only utilizes RANSAC but also gives weights to each feature according to how consistent it was in earlier frames. Petit et al. [104] also discussed probing objects multiple times to limit the effectiveness of attacks. Additionally, it may be possible to combine spatial, spectral, and temporal fusion schemes to further enhance security.

## 3.4 Discussion

### 3.4.1 Improving Transduction Attack Model

The model serves as an initial step towards formalizing analog sensor security. For example, it may be desirable to abstract new measurement shaping steps exclusive to in-band attacks on active sensors that manipulate measurement by adjusting the injection time [127, 104, 162] or signal frequency [129]. Incorporating emerging and

potential attacks that modify the transfer functions, e.g., by injecting EMI to the power lines of amplifiers [147] or heating temperature-sensitive components, may also be a direction for future work.

### 3.4.2 Improving Research Methodology

I hope and believe the model and related analytical methods will improve the transduction attack research methodology. This model highlights the importance of discovering new signal injection and measurement shaping steps, as well as defenses for these steps. The methodology demonstrates how these steps can apply across a wide range of sensors. Thus, research based on the discovery or analysis of steps may have a broader impact on sensor design as a whole rather than only targeting a single sensor or system. Additionally, using the provided terminology can assist researchers and sensor designers in understanding new transduction attacks and how the attack may apply across sensors as a whole.

### 3.4.3 Predictive Defense Schemes

I discuss how the model enables two predictive defense schemes that enhance sensor resiliency to transduction attacks.

- **Predictive Attack Defense.** A sensor designer could employ the strategy of implementing a defense for every theoretical attack on a sensor. The model allows a designer to predict theoretical attacks on a sensor they are designing. From this, the designer can adopt a simple strategy of ensuring there is at least one defense for each theoretical attack. In addition to the benefit of mitigating possible future attacks, this approach may reduce the loss of time and money associated with redesigning, manufacturing, and distributing a new sensor each time a new attack is demonstrated against a sensor.

- **Predictive Step Defense.** Predictive step defense employs a different approach of designing a mitigation for every known signal injection or measurement shaping step in a sensor. In the model, a successful transduction attack requires all steps in its attack chain. Mitigating any step in the chain will mitigate the entire attack. So, a strategy that mitigates every known injection or shaping step will prevent all attacks from exploiting those mitigated steps, including attacks that have not yet been theoretically constructed. Thus, after this defense scheme is employed, an attacker would need to construct an attack chain entirely comprising newly discovered steps. Therefore, predictive step defense provides a scheme for designers to protect their devices against unknown theoretical attacks at design time.

## 3.5 Conclusion

Security researchers and practitioners can use the Transduction Attack Model to better express and understand attacks employing physical signals to manipulate sensor output, and defenses against them. This model employs transfer functions and a vector of adversarial noise to allow the comparison of attacks across sensors of different types. The model allows some predictive capability and enables new defense schemes to make sensors more resilient against future attacks.

# CHAPTER IV

# Blue Note: A Transduction Attack Case Study on Hard Disk Drives

This chapter, based on my past work[1], provides an in-depth example of two transduction vulnerabilities via an attack that disrupts hard disk drive (HDD) availability using acoustic waves. Availability is the most important security property of a consumer hard disk drive (HDD). A lack of availability prevents meaningful preservation of other security properties such as confidentiality and integrity. Specifically, this chapter explores to what extent an adversary can intentionally manipulate HDDs with acoustic waves (Figure 4.1) and mitigation designs for this specific attack.

Magnetic HDDs remain common [53] because of the long tail of legacy systems and the relatively inexpensive cost for high capacity storage. However, sudden movement can damage the hard drive or corrupt data because of the tight operating constraints on the read/write head(s) and disk(s). Thus, modern drives use shock sensors to detect such movement and safely park the read/write head. Previous research has indicated that loud audible sounds, such as shouting or fire alarms, can cause drive components to vibrate, disturbing throughput [119, 111, 131, 35]. Audible sounds can even cause HDDs to become unresponsive [98].

[1]Connor Bolton, Sara Rampazzi, Chaohao Li, Andrew Kwong, Wenyuan Xu, Kevin Fu. "Blue Note: How Intentional Acoustic Interference Damages Availability and Integrity in Hard Disk Drives and Operating Systems." In *Proceedings of the 39th IEEE Symposium on Security and Privacy (SP)*, 2018.

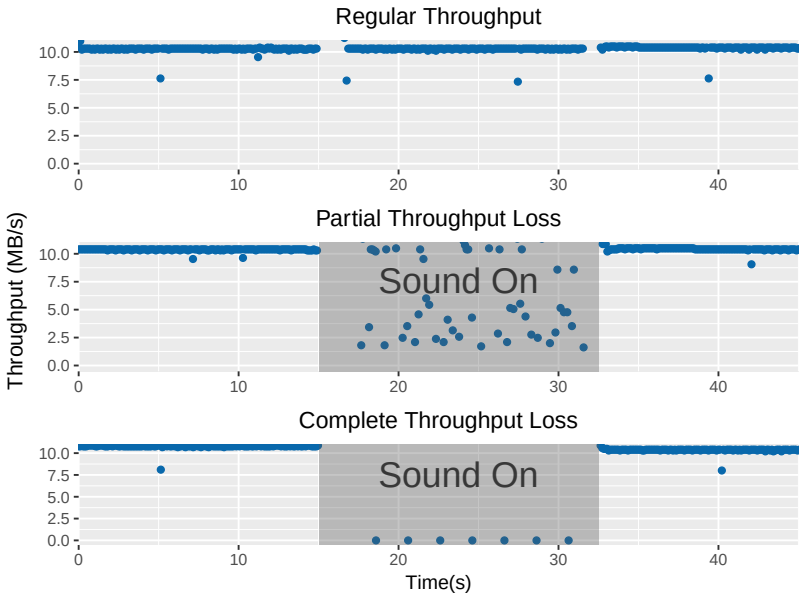



Figure 4.1: Vibration can interrupt disk I/O. Three plots show a Western Digital Blue WD5000LPVX drive under normal operation (top), partial throughput with vibration induced by a 5 kHz tone at 115.3 dB SPL (middle), and halting of writes with 5 kHz tone at 117.2 dB SPL (bottom).

This chapter explores how sustained, intentional vibration at resonant frequencies of various HDD sensor system components (*i.e.,* transduction vulnerabilities) can cause permanent data loss, program crashes, and unrecoverable physical loss in HDDs from three different vendors (Figure 4.2). This chapter also proposes, simulates, and implements several defenses against such attacks on HDDs that may translate to other transduction attacks. This chapter's contributions include:

- **Physical Causality:** How intentional audible and ultrasonic sounds cause errors in an HDD sensor system via two separate transduction vulnerabilities.
- **System Consequences and Attack Case Studies:** How these transduction vulnerabilities cause system-level errors such as hard disk drive non-responsiveness. How an adversary could exploit these errors for an end-to-end attack.
- **Defenses Design Patterns:** Simulate, implement, and propose mitigations for these vulnerabilities.

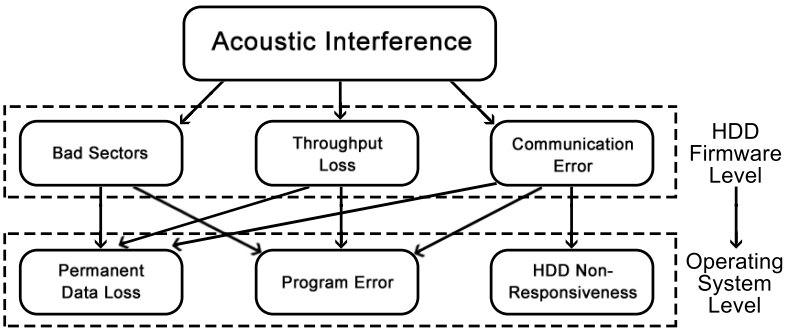



Figure 4.2: Intentional acoustic interference causes HDD firmware errors due to transduction vulnerabilities in the hard disk drive, which in turn cause system-level errors and other undefined application-level behavior. An arrow indicates a confirmed cause and effect relationship.

## 4.1 Overview

### 4.1.1 Hard Disk Mechanics and Acoustics

Acoustics vibrate the HDD head stack assembly and shock-sensor, leading to throughput loss and physical damage.

**Hard Disk Mechanics:** An HDD read/write head floats (∼10 nm) above the surface of each spinning disk. Data is organized in tracks that circle the disk. To read or write data, the head stack assembly (HSA) must position the head above the desired track. There is a narrow margin of error (on the scale of nm) within which the read/write head can operate. For writes, there is a narrower margin of 10% of the width of the track, while there is a 15% margin for reads [32].

Vibration poses problems for HDD designers. First, vibration may push the head away from the center of the track and render the drive temporarily unable to write. Second, the head may crash into the surface of the platter, physically damaging the disk and leading to possible data loss.

**Compensating for Vibration:** Two approaches can correct for positional error due to vibration (Figure 4.3): (1) a standard feedback controller that adjusts the head position using the current positional offset of the head from the center of a track and

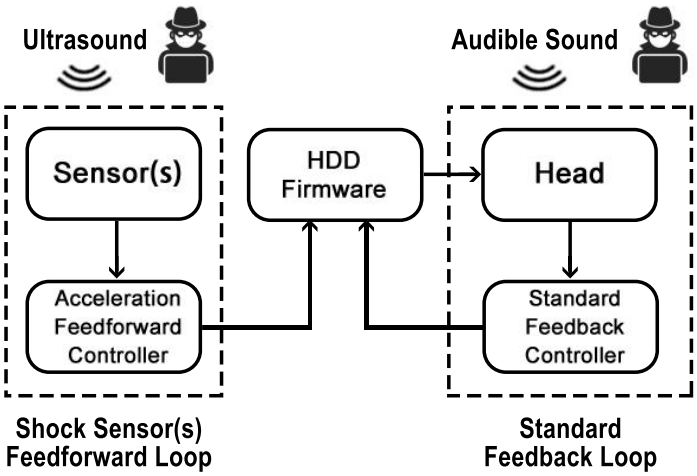



Figure 4.3: Acoustics disturb the HDD head stack assembly and shock sensor. Modern HDDs use sensor-driven feedforward controllers to adjust the head's position. Our work finds that ultrasonic vibration triggers false positives for head parking; audible tones vibrate the head—causing poor positioning.

(2) a feedforward controller where a shock sensor adjusts the head in anticipation of vibration. The HDD will park its head away from the track when the shock sensor senses extreme vibration, such as when a laptop falls.

### 4.1.2 Threat Model

The attacks in this chapter assume an adversary that uses vibration to interfere with an HDD on a target machine, typically induced through the use of a speaker. The adversary may catalog frequencies that are most effective for a given model of hard drive to speed up the attack. Two distinct attack models are investigated: a *self-stimulation attack* [144] and a *physical proximity attack*.

**Self-stimulated Attacks:** An adversary can attack an HDD by inducing vibration via acoustic emitters built into the victim system (or a nearby system). In this case, an adversary would temporarily control an emitter in the system through some means. The attack is more likely to succeed when the emitter is powerful and/or very close to the victim.

A self-stimulated attack may use a standard phishing attack, malicious email,

or malicious javascript to deliver audio to a laptop's speakers. Most laptops have speakers and the ability to browse the Internet. Modern browsers support JavaScript and HTML5, both of which are capable of playing audio without user permission. Therefore, should a victim visit a page owned by the attacker, the attacker would be able to play audio over the victim's speakers.

The frequency response of a built-in speaker may limit the ability of an adversary to deliver ultrasonic attacks, but some speakers may be able to deliver ultrasonic or near ultrasonic tones.

**Physical Proximity Attacks:** An attacker can induce vibration using a speaker near the victim system. The attacker must either control a speaker close to the victim's HDD or place a speaker in the proximity of the system. The case of controlling a speaker close to the victim's HDD is similar to that of the self-stimulated attack. An example of this would be the attacker controlling an AM or FM station of a radio playing sound near the victim's HDD with the desired signal.

When the attacker is able to physically place the speaker, the attacker can choose a speaker with the desired frequency range (audible, near ultrasound, or ultrasound). In addition, the attacker can choose non-traditional acoustic emitters that may beam-form signals to attack a drive from a long distance. A Long Range Acoustic Device (LRAD) can send audible acoustic waves above 95 dB SPL miles away in open air [30].

## 4.2 Experimental Method

There are three operational challenges to quantify the effects of acoustic interference on hard disk drives: (1) isolating the experiment from uncontrolled signals, (2) inducing precise vibration at the HDD, and (3) accurately measuring HDD errors due to acoustic interference. Unless noted otherwise, the experiments in this paper shared the same physical setup described in this section. Note that a setup with this

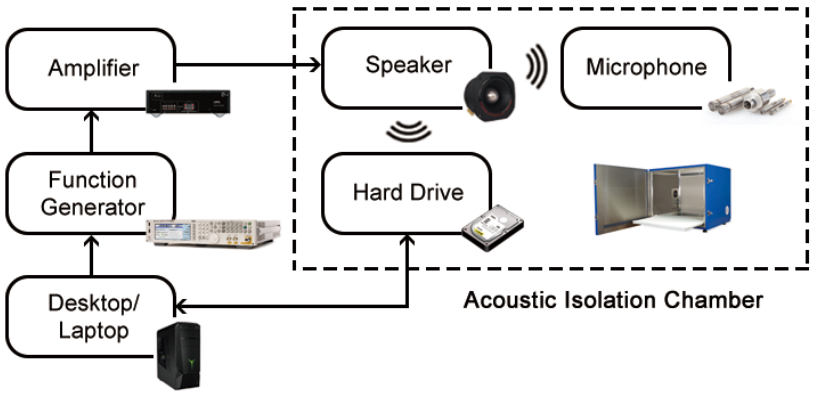



Figure 4.4: The physical setup for testing mechanically uncoupled acoustic interference. For mechanically coupled tests, the device containing both the HDD and speaker (such as a laptop) lay directly inside the chamber.

level of precision is only needed for scientific measurement to discover causality, but an attacker could use a simpler setup to cause the deleterious effects.

### 4.2.1 Isolating the Experiment

The setup must prevent environmental factors from significantly altering the results of the experiment. In the setup, the HDD lies in an acoustic isolation chamber, as shown in Figure 4.4, to prevent unintended noise from altering results. The setup also monitors the drive's temperature using SMART data to ensure the temperature stays within operational limits (below 50 °C [9]). The speaker hangs from the ceiling to mechanically uncouple it from the HDD in all tests.

### 4.2.2 Generating Sound

Accurately generating vibration is crucial in observing the effectiveness of this attack. Audible and ultrasonic frequencies use the same basic setup (Figure 4.4).

**Audible Frequencies:** This setup generates audible frequencies using a Tektronix AFG3251 function generator, a Yamaha R-S201 audio receiver, and a Pyramid Titanium Bullet Tweeter speaker. The setup verifies the emitter's output using a G.R.A.S. Type 26CB microphone, a G.R.A.S. 12AL preamplifier, and a PicoScope 5444B.

**Ultrasonic Frequencies:** This setup generates ultrasonic frequencies using a Keysight N5172B EXG X-Series RF Vector Signal Generator, a CRY584 Power Amplifier, and a NU C Series Ultrasonic Sensor. The setup measures the emitter's actual output using a CRY343 microphone and a RIGOL DS4022 oscilloscope.

### 4.2.3 Measuring the Effects of Vibration

The effects of vibration on HDDs during operation are typically: (1) throughput loss, (2) program crashing when using the HDD, and (3) writes or reads taking an indefinite amount of time to return (even if the acoustic interference subsides in the middle of the write).

The testing computer measures throughput using writes to the victim disk via the Linux `dd` utility with the `fdatasync` option. `dd` is a well-known and tested tool for basic throughput measurement. The testing computer writes 1MB of pseudorandom data directly to a pseudorandom location on the disk to avoid caching that may speed up the write process. The `fdatasync` option forces `dd` to wait for each block of data to be physically written to disk before writing the next block. Despite being well tested, `dd` often crashes or hangs indefinitely during use. By monitoring `dd` in a separate process, errors can be quickly intercepted and logged.

## 4.3 Transduction Vulnerabilities

In investigating this problem, two separate transduction vulnerabilities were found in (1) vibration of the head stack assembly and disk platters and (2) motion sensor spoofing.

### 4.3.1 Head and Disk Displacement

The first transduction vulnerability is the vibration of the read/write head and disk platters of the hard disk drive (HDD). This is considered a transduction attack

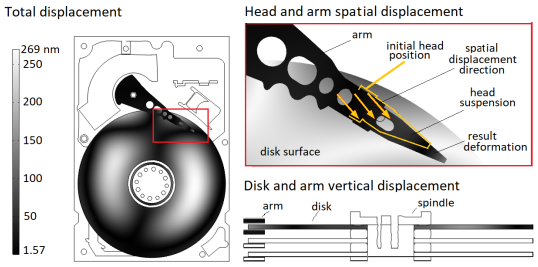
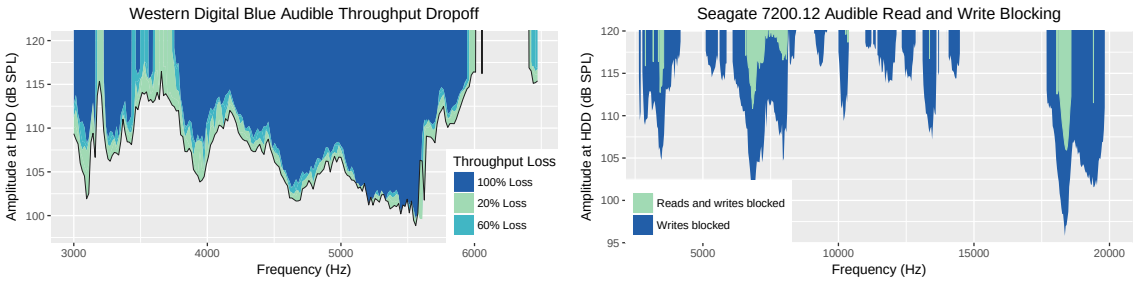



Figure 4.5: COMSOL simulation showing displacement of a HDD head assembly and disk during 5 KHz acoustic signal attack (left: top view; bottom right: lateral cross-section; top right: R/W head displacement). Note the displacement on the disk surface (∼156 nm of maximum vertical displacement across the central tracks), and the maximum horizontal displacement of the head suspension (∼8 nm, rectangle box). This exceeds the 7.5 nm read and 5 nm write fault thresholds, assuming a 50 nm width.

that does not fall into the scope of the Transduction Attack Model (Section 3.2), as these components are not part of the sensor itself, but are part of the feedback control in the HDD's sensor system (Figure 4.3). However, the attack functions similarly to many of the attacks discussed in the model. Prior work reports that audible acoustic waves cause throughput loss [119, 111, 131, 98]. A Finite Element Model and numerous experiments analyze how acoustic waves (and thus vibrations) displace the read/write head or disk platter outside of operational bounds, resulting in either partial throughput loss or complete loss of throughput (Figure 4.1).

#### 4.3.1.1 Vulnerable Hard Disk Drive Mechanics

A Finite Element Model (Figure 4.5, made using COMSOL, demonstrates the vibroacoustic response of the HDD's individual mechanical parts (a common use for Finite Element Models [34, 103]) with common manufacturer materials and parameters [78]. Figure 4.5 shows how acoustic waves can displace a read/write head or disk platter outside of operational bounds, inducing throughput loss. This model is simulating a 5 kHz acoustic wave striking the HDD chassis from above at 120 dB SPL. The model estimates maximum disk displacement of about 33 nm horizontally and




(a) Thresholds of write throughput loss due to audible signals

(b) Read and write blocking thresholds due to audible signals

Figure 4.6: Throughput loss under acoustic interference for a Western Digital Blue HDD and Seagate 7200.12 HDD. There is a measurable gradual degradation in throughput at each frequency for the audible range. Note that audible frequencies require less power to block writes than reads due higher operation tolerances.

156 nm vertically, while estimating maximum read/write head displacement of 9 nm horizontally and 112 nm vertically.

Given a track width of 50 nm [14], a 10% track width margin (i.e., a 5 nm margin) of error for writes and 15% margin for reads (i.e., a 7.5 nm margin) [32], and a vertical distance of 6 nm between the head and the disk [157], these displacements push the drive outside of its operational bounds for reading and writing. In addition, these numbers show the possibility of the read/write head crashing into the disk.

#### 4.3.1.2 Mechanical Throughput Loss Pathologies

Using the setup described in Section 4.2, we gathered data to show the two main qualities of throughput loss induced by head stack assembly and disk vibration: non-binary throughput loss and reads being significantly harder to block than writes.

**Non-Binary Throughput Loss:** One critical quality of throughput loss due to head stack assembly vibration is that it allows for partial throughput loss as shown in Figure 4.6a. A signal can be strong enough to vibrate the read/write head or disk sufficiently to hinder typical write throughput, but not strong enough to completely block the drive from reading or writing to disk. Figure 4.1 shows this behavior as the

lower amplitude signal vibrates the read/write head enough to hinder operation, but not enough to completely block reads and writes. Then, when the amplitude of the signal increases, the vibration of the read/write head also increases, leading to the drive being unable to read or write.

**Reads Require Higher Amplitudes to Block:** Another quality of throughput loss via head stack assembly vibration is that read blocking generally requires greater amplitudes than write blocking, shown in Figure 4.6b. This is because the operating margin of error is greater for reads than for writes. Thus, the head may vibrate within the read error margin but outside the write error margin.

### 4.3.2 Motion Sensor Spoofing

The second transduction vulnerability is spoofing MEMS vibration or accelerometer sensors. These sensors detect sudden disturbances (e.g., dropping the HDD) such that the HDD can park its head to prevent damage. Accelerometers were shown to be vulnerable to malicious sound waves and vibration [144] (Section 2.3). This section shows that piezo shock sensors are also subject to similar attacks with ultrasonic acoustic waves. The erroneous sensor output tricks the HDD into inadvertently parking its head, rendering the drive unable to read or write to disk. This vulnerability is shown in the Transduction Attack Model Systematization Table under the application of "Hard Disk" (Table 3.2).

#### 4.3.2.1 Vulnerable Sensor Mechanics

**Spoofing the Shock Sensor:** One can vibrate the shock sensor mass at its resonant frequency to induce false sensor output similar to prior work on spoofing MEMS accelerometers [144] and MEMS gyroscopes [135]. Shock sensors work similarly to MEMS accelerometers in that vibration of a sensing mass creates a voltage representative of the motion perceived by the sensor. By placing a shock sensor on an object,

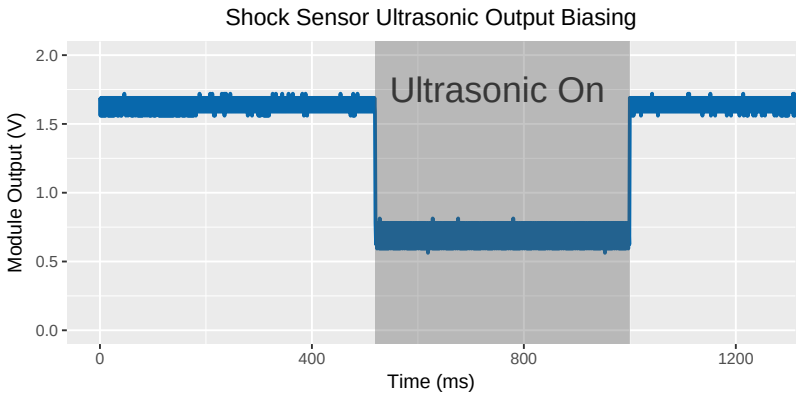


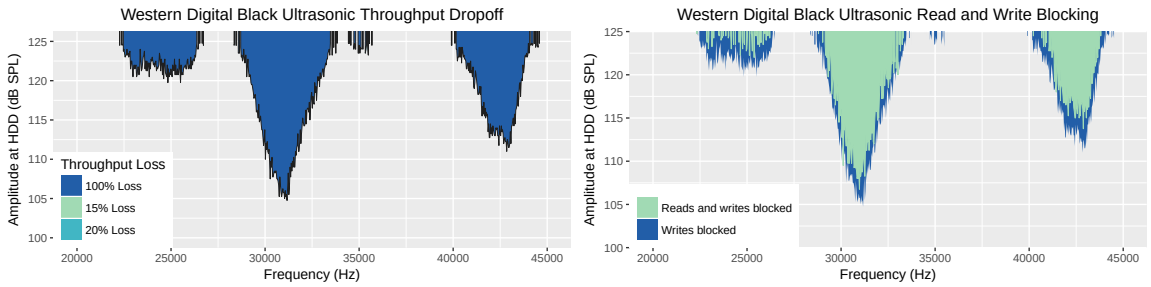

Figure 4.7: An ultrasonic wave alters the output of a piezo shock sensor in a PKGX-14-4010 shock sensor evaluation module.

the shock sensor can produce a voltage representative of the object's vibration. However, one can make the vibration of the mass of the piezo shock sensor different from the vibration of the object by exploiting resonant frequencies.

Figure 4.7 demonstrates altering output of a PKGX-14-4010 MEMS shock sensor evaluation module, which is believed is the same unit inside the Toshiba MQ01ABF050 HDD (Figure 4.7). The output of the shock sensor module under normal operation (with no intentional acoustic interference) is approximately 1.6 V. However, a 27 kHz tone at 130 dB SPL produces a 0.6 V output — incorrectly indicating over $10g$.

**Throughput Loss from Sensor Spoofing:** A spoofed sensor can lead to throughput loss by making the HDD inadvertently park its head. Under intentional acoustic interference, the shock sensor or accelerometer will report a false value to the HDD firmware. This false value implies that the HDD is moving violently, such as if it were dropped, and needs to park the read/write head. It follows that an attacker could continuously falsify the sensor's output to keep the head parked indefinitely, preventing the HDD from writing or reading.

Experiments confirm throughput loss from sensor spoofing. First, the setup plays inaudible sound at a resonant frequency of the shock sensor in the HDD (27 kHz at 125 dB SPL), which results in throughput loss (Figure 4.7). Second, to confirm that




(a) Thresholds of write throughput loss due to ultrasonic waves

(b) Read and write blocking thresholds due to audible signals

Figure 4.8: Ultrasonic throughput loss for a Western Digital Black WD1600BJKT HDD. In contrast to audible frequencies, ultrasonic frequencies cause full throughput loss (no partial) and block writes and reads using similar amplitudes.

it is indeed the shock sensor that causes the throughput loss, instead of read/write head or disk vibration, we removed the shock sensor from the drive and measured throughput with and without acoustic interference. This confirms that the sensor's erroneous output caused by acoustic interference leads to throughput loss.

#### 4.3.2.2 Sensor Throughput Loss Pathologies

**Binary Throughput Loss:** The throughput of the HDD is either unaffected or lost completely as shown in Figure 4.8a. This method cannot induce partial throughput loss as head parking is the root cause of throughput loss. The head can only be either parked or operate normally (assuming no other kind of interference).

**Similar Amplitudes to Block Reads and Writes:** Another observation is that write blocking and read blocking require similar amplitudes for sensor-induced throughput loss shown in Figure 4.8b. This observation may be because the firmware's threshold for head parking is similar, but not exactly the same for reads and writes.

### 4.3.3 Other Pathologies or Observations During Testing

#### 4.3.3.1 Consistent Resonance despite Manufacturing Variation

During testing, drives of the same model showed similar characteristics when subjected to acoustic interference. I attribute slight differences to process variation. These observations are consistent with previous research [35] that shows little frequency-dependent variation across drives of the same model. An adversary can predict effective frequencies for target drives by profiling a drive of the same model.

To test this characteristic, I profiled one Western Digital Blue WD5000LPVX HDD to discover the frequency that most affects drives of this model. Then I subjected 13 other drives of the same make and model to this frequency. The vibration denied each drive from being able to read or write. I also observed that ultrasonic interference exhibited consistent resonant frequencies across drives of the same model. In practice, experiments find that the most vulnerable frequencies remain similar from drive to drive of the same model.

#### 4.3.3.2 Bad Sectors

The vast majority of drives used in our tests developed several bad sectors or became nonoperational. While this work does not specifically conduct an experiment to test for abnormal levels of bad sectors, other indicate this trend.

**Gathering the Data:** Throughout our experiments, we collected the bad sector data presented in Table 4.1 through the Self-Monitoring, Analysis, and Reporting Technology (S.M.A.R.T.) system, a de-facto HDD monitoring standard that can measure bad sectors in HDDs [88, 106]. Our observations are anecdotal rather than controlled experiments. The drives were subjected to different frequencies, amplitudes, and durations of acoustic interference. All drives had between 15 and 500 power-on hours, except one drive that had 755 hours.

| Drive | # of Tested Drives | Avg # Bad Sectors |
|---|---|---|
| WD Blue WD5000LPVX | 7 | 705 |
| WD Enterprise WD1003FBYZ | 1 | 82 |
| WD Purple WD10PURX | 1 | 500 |
| Seagate 7200.12 | 3 | 961 |
| WD Black WD1600BJKT | 2 | 321 |
| Toshiba MQ01ABF050 | 1 | 14,448 |
| Total | 15 | 1,639 |

Table 4.1: The cumulative bad sector data for several drives used in various experiments. All drives had between 15 and 500 power on hours (except one that had 755 power on hours).

**Interpreting the Data:** As shown in Table 4.1, many of the drives tested showed high bad sector counts. In fact, every drive suffered at least one bad sector. As storage expert Erik Riedel [112] remarks "it would be highly unusual to regularly find bad sectors on hard disk drives under 500 power-on-hours." Analysis of bad sectors in consumer-grade drives from data center environments is consistent with the assertion that bad sectors are rare. Google found that only 9% of their consumer-grade hard disk drives developed any bad sectors [106] over eight continuous months of use.

I surmise that the alarming number of bad sectors is due to head crashes caused by the force that the sound exerts on the head stack assembly during experimentation (as outlined Section 4.3.1.1). For instance, scratches visible to the human eye were found on platters after disassembling some of the tested drives. However, there could be several other factors at play. For example, it is possible that the HDD firmware is incorrectly marking sectors as physically damaged after failing to write to them several times because of the interference.

Ultrasonic attacks are less likely to cause a head crash but could damage the drive in other ways such as causing the head to become unstable over time because of excessive parking. This instability could make the drive less reliable in its reads and writes, leading to sectors being marked as bad. For example, in a test that subjects the Toshiba HDD to an ultrasonic signal at the head parking amplitude threshold, one can hear head parking in rapid succession, possibly damaging the head controller.

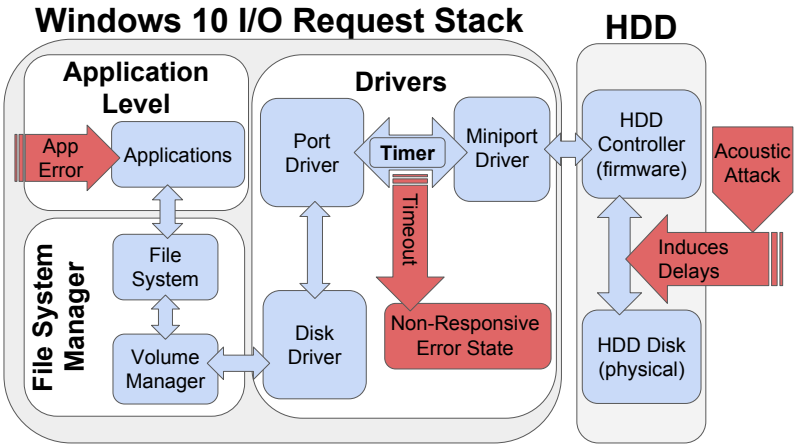



Figure 4.9: On Windows 10, prolonged acoustic interference induces delays in the HDD that cause a timer in I/O requests between the port driver and miniport driver to timeout, leading to the HDD entering a non-responsive state. Light blue indicates the normal path of operation while dark red shows what happens during an acoustic attack.

## 4.4 System Level Errors: HDD Non-responsiveness

During throughput testing under malicious acoustic interference (Sections 4.3.1 and 4.3.2), HDDs become non-responsive to the operating system (both Windows and Linux). Prior research by the IT security community [98] observed similar phenomena, yet the exact causality in the operating system remained a mystery.

### 4.4.1 Causes of Non-Responsiveness Errors

Evidence suggests that prolonged throughput loss may cause an HDD to enter a non-responsive state by causing timeouts in I/O requests, along with other errors in the I/O request stack. This non-responsive state lasts until the HDD is physically unplugged and reconnected or the operating system restarts. Examining the Windows 10 I/O request path, particularly the port and miniport drivers, reveals what practices cause these errors.

**I/O Request Path to an HDD:** The non-responsiveness error originates in the I/O request path (Figure 4.9). In Windows 10, several actors process each I/O storage

request (i.e., read, write, or control operations to the HDD) before delivering the request to the HDD [3]. When a typical file read/write request reaches the file system, the file system passes the file's location information to the volume manager as a partition offset. The volume manager converts this partition offset into an HDD block number and sends it to the disk driver. The disk driver converts the I/O storage request containing the HDD block number to a SCSI request block and sends the request block to the port driver, which interfaces with the HDD miniport driver. The miniport driver takes the request and sends it to the HDD.

**I/O Timeouts and Other Errors:** I/O timeouts and other errors in the I/O request path can lead to the drive entering a non-responsive state. In Windows 10, the timeout is specified in the port and miniport drivers. The port driver manages general data flow for a class of devices, in this case HDDs, whereas the hardware manufacturer designs the miniport driver to handle data flow specific to the device [5]. The pair work in conjunction to pass information from the disk driver to the HDD. When an I/O request packet is sent from the port driver to the miniport driver, the I/O request packet is put in a pending queue until the request is completed [7]. A timer monitors each unfulfilled request. The timer should never expire normally as expiration implies the device has stopped responding [8].

We find two types of errors in Windows 10. (1) The port driver may timeout, indicated by an error with Event ID code 129. When this happens, all outstanding I/O requests report an error to the programs that issued the request, and the port driver sends a reset request to the hard drive [6]. (2) Some miniport drivers also report a second error code with Event ID 153. Some miniport drivers may detect when port driver timeouts are about to occur and abort the request itself [4]. The miniport driver then returns an error code (ID 153) instead of the port driver returning an error code. The miniport driver may also return an error (also ID 153) if it detects HDD bus communication errors, unrecoverable read errors, or undocumented errors.

### 4.4.2 Observations

**Windows 10:** During an attack, we mainly observe errors originating from the port driver (ID code 129), but also some from the miniport driver (ID code 153), that affected numerous applications and could even crash the operating system. The numerous port driver errors indicate I/O requests frequently timing out, and also that numerous HDD reset commands are sent to the miniport driver. However, some of these reset commands remain incomplete, resulting in all outstanding requests remaining stuck, and causing some operating system applications to freeze. The miniport driver also returned errors, indicating possible bus or unrecoverable read errors. Sporadically, the Windows 10 OS would crash with a `CRITICAL_PROCESS_DIED` or `UNEXPECTED_STORE_EXCEPTION` error, likely because a critical process did not handle the port or miniport errors correctly.

**Ubuntu 16.04:** Expired timers in the I/O request chain lead to Ubuntu remounting all loaded files as a read-only file system, with any previously unaccessed files becoming inaccessible. Ubuntu 16.04 logging files (`dmesg`, `kern.log`, and `syslog`) confirm that the hard disk controller driver (in this case a generic ATA/SATA II controller driver) return errors to the operating system when under attack from acoustic interference. These errors are due to the expired timer of the outstanding I/O requests in the pending queue (e.g., `READ/WRITE FPDMA QUEUED` command failure) [1]. When the hard drive detects these conditions, it sends an error message to the controller driver, and waits to receive a reset command. Note that the controller driver tries a finite number of times (usually four) to send the reset request to the hard drive.

The file system disconnects and remounts as read-only if the attack persists after the last reset request failures. `dmesg` shows `COMRESET failure (errono=-16)` four times until finally showing `reset failed, giving up`. Then, the attack can also generate `delayed block allocation of inode` error followed by a `This should`

| Model | Freq (kHz) | Amp (dB SPL) | Time (s) |
|---|---|---|---|
| WD Blue WD5000LPVX | 4.6 | 118.1 | 100 |
| WD Purple WD10PURX | 6.9 | 118.9 | 130 |
| Seagate 7200.12 | 7.0 | 119.1 | 120 |
| WD Black WD1600BJKT | 21 | 120.0 | 5 |
| Toshiba MQ01ABF050 | 27 | 127.2 | 8 |
| WD Blue WD5000LPVX | 31 | 138.1 | 6 |
| Seagate 7200.12 | 31 | 139.5 | 6 |

Table 4.2: The frequency, amplitude, and the minimum required duration of acoustic signals used to induce vibration resulting in communication errors that persisted until system restart, HDD restart, or physical disconnection and reconnection of the HDD to the computer on Linux. Ultrasonic frequencies were able to induce errors in as few as 5 seconds while audible frequencies took as few as 100 seconds.

`not happen!! Data will be lost` message. In addition, the message `previous I/O error to superblock detected` might appear multiple times. These error messages indicate file system corruption and data loss.

### 4.4.3 Measuring Non-Responsiveness Errors

We measured how long it took to induce non-responsive errors on several HDDs.

**Setup:** We placed the drives in the experimental setup described in Section 4.2 and determined an effective frequency for acoustic interference. The test began throughput measurements as described in Section 4.2.3 for one minute without an acoustic signal present. Next, the experiment subjected the drive to intentional acoustic induced vibration, and afterward queried the drive to provide its identification information.

**Results:** Drives exhibited similar behavior when the error occurred (Table 4.2). After the acoustic signal subsided, the drive would still appear to the operating system as a block device. However, when queried for its basic info, the drive would typically not respond. In rare cases, it would send back nonsensical data, such as the WD Blue drive reporting non-displayable characters for its model number and that its capacity was 2,692 PB when its actual capacity was 500 GB. These problems persisted until either the computer was restarted, the HDD was power cycled, or the SATA cord was physically disconnected from the drive and reattached.

## 4.5 Attack Case Studies

We demonstrate two case studies where an attacker exploits the transduction vulnerabilities discussed previously (Section 4.3). In addition, we describe how an attacker might select a frequency to attack a drive.

### 4.5.1 Attack Frequency Selection

To maximize effectiveness, an adversary would select a frequency that requires the smallest acoustic amplitude to disturb a target HDD. To do so, an adversary may consider the frequency responses of the speaker and HDD, and whether or not an inaudible signal is possible or desirable. Note that because manufacturing variation has a low effect on drive characteristics (Section 4.3.3.1), an attacker can select a frequency using a different HDD of the same model as the victim drive.

**Speaker Profiling:** To profile a speaker's frequency response, one can simply record the loudness of the speaker at each desirable attack frequency. Alternatively, the frequency response of the speaker may be available online. Our tests indicate speakers of the same model share similar frequency responses, allowing an attacker to profile a speaker of the same make and model of a target speaker if the target speaker itself is unavailable.

**HDD Profiling:** The attacker can profile a drive as follows. The attacker sweeps the frequency range and finds the minimum amplitude that causes write blocking for each frequency. In addition, the program should periodically check the drive to ensure it is still working properly within operating margins. This includes checking the drive temperature (to see if it has overheated), the number of bad sectors, and that the throughput of the HDD is similar to normal operating parameters.

**Choosing a Frequency for Attack:** Choosing an attack frequency can be as simple as overlaying the speaker profile and HDD profile, then observing the cross-section

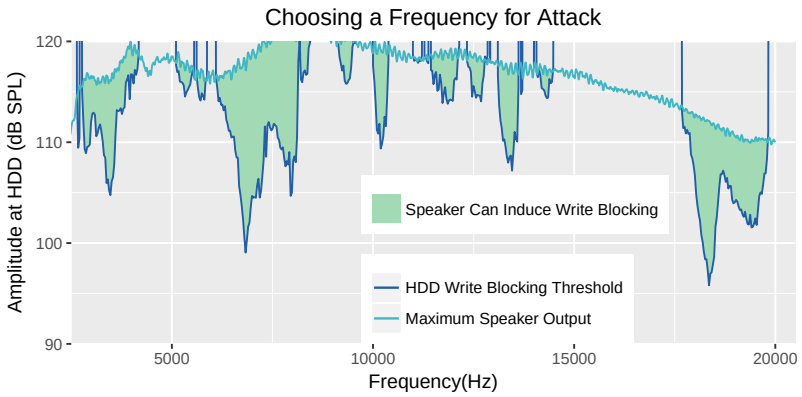



Figure 4.10: Profiles for a Seagate 7200.12 HDD and a Pyramid TW28 speaker are shown above. The areas where the profiles overlap (the shaded areas) are those where the speaker can block HDD writes.

(Figure 4.10). After doing so, one could choose a frequency in one of the largest areas of overlap for the best possibility of a successful attack. Alternatively, if ultrasound or near ultrasound (as some people cannot hear near ultrasonic frequencies because of high-frequency hearing loss) is an available frequency, then it may be desirable to select that frequency over others to make the attack harder to detect.

### 4.5.2 Case Study 1: Blue Note

Experiments demonstrate several proof of concept attacks that affect both Windows 10 and Ubuntu 16.04 systems in various scenarios. A webpage can launch a self-stimulated attack on a laptop using the laptop's own speakers while requiring no extra user permissions. An attacker can place a speaker near a victim's desktop computer to conduct an inaudible physical proximity attack on the desktop computer, even with the speaker and victim physically decoupled.

**Test Methodology:** This setup assumes that the attacker knows the model of the victim drive and determined the vulnerable frequencies via the method in Section 4.5.1. For each test, we installed a fresh operating system on the victim HDD, then placed the victim system in an acoustic isolation chamber.

For self-stimulation attacks, the victim accesses the adversary's website —perhaps

| Attack Type | Machine Description | Hard Disk Drive | Operating System | Freq (kHz) | Time Until (s) | Symptom Description |
|---|---|---|---|---|---|---|
| Self Stimulation Attack | Dell XPS 15 9550 Laptop | WD Blue WD5000LPVX | Windows 10 | 7.83 | 45 | Frozen |
| | | | | | 125 | System Crash |
| | | | Ubuntu 16.04.1 | 7.95 | 120 | HDD Non-Responsive (until OS restart) |
| | HP Elite Minitower w/ HP DC7600U Speaker | WD Blue WD5000LPVX | Windows 10 | 4.60 | 80 | Intermittent Freezing |
| Physical Proximity Attack | HP Elite Minitower | WD Blue WD5000LPVX | Windows 10 | 10.00 | 113 | System Crash |
| | | | Ubuntu 16.04.1 | 10.00 | 225 | HDD Non-Responsive (until OS restart) |
| | Intel NUC NUC5i5RYH | Seagate 7200.12 | Windows 10 | 5.60 | 180 | HDD Non-Responsive (until OS Restart) |
| | | | Ubuntu 16.04.3 | 5.60 | 120 | HDD Non-Responsive (until OS Restart) |
| | Sony PCG Laptop | Samsung HM321HI | Windows 10 | 40.00 | 120 | System Crash |

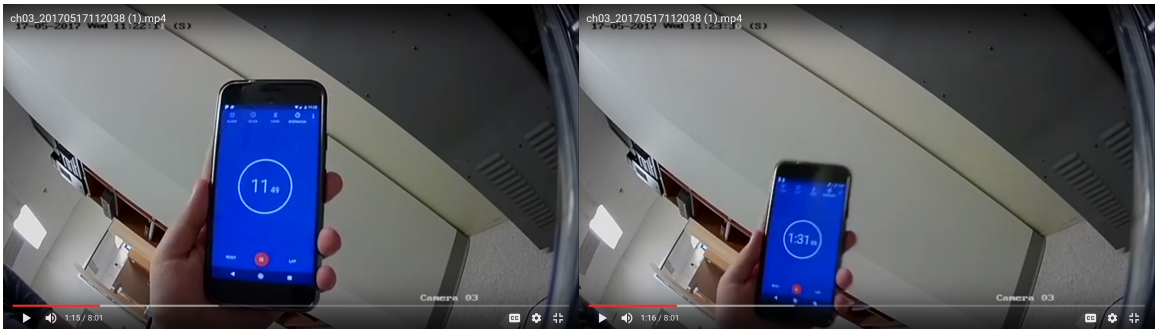
Table 4.3: A selection of attacks against operating systems using acoustically induced vibration. Windows 10 commonly froze, and would sometimes crash. On Ubuntu, the drive would often remount as read only.

through a phishing attack or a link within a malicious email. The site then plays malicious audio without permission over the system's built-in speaker to attack the HDD. The victim accesses the malicious site using the latest version of Google Chrome (58.0.3029.110).

For physical proximity attacks, the attacker places a chosen speaker near the HDD. Thus, the malicious acoustic waves may be audible or inaudible depending on the chosen speaker.

**Results:** Table 4.3 summarizes a selection of repeatable attacks on different laptops, operating systems, frequencies, and the minimum required interference duration before the reported symptom appears. For Windows and Linux, the average case across all tests (the majority of which are not shown) was that the HDD became non-responsive (described in Section 4.4) after playing audio for a prolonged period of time. This was the case for both ultrasonic and audible attacks. However, one notable outlier symptom was the Windows operating system crashing after freezing, displaying a `CRITICAL_PROCESS_DIED` or `UNEXPECTED_STORE_EXCEPTION` message.

**Possible Causes of System Crashing:** It is likely that the Windows 10 crash




(a) Frame Before Video Loss (b) Frame After Video Loss

Figure 4.11: Two frames from an unedited recording taken from a surveillance video system's HDD. During recording, the system was subjected to acoustic interference. The displayed images are roughly 5 frames apart (less than a second apart in video playback), including one frame that was only partially written because of acoustic interference. However, the timestamps indicate that roughly 80 seconds of video are missing due to the interference.

is closely related to the non-responsive error discussed in Section 4.4. The information extracted from the crash dumps generated by the operating system reveals information about the crashes. The crash dumps show the miniport driver returning a device error (`STATUS_IO_DEVICE_ERROR`), indicating there was an error in the HDD. The operating system does not seem to handle this error correctly, leading to `UNEXPECTED_STORE_EXCEPTION`. This indicates that the memory manager required data from the disk, but was unable to write into memory because of an in-page I/O error.

### 4.5.3 Case Study 2: Video Surveillance

An attacker can prevent a video surveillance system from writing to its HDD, resulting in recorded video loss. Video surveillance systems constantly store large quantities of video. These systems typically use HDDs rather than SSDs because of the need for a large storage capacity. For such systems, the integrity of the recorded data is vital to the usefulness of the system, which makes them susceptible to acoustic interference or vibration attacks.

| Interference Duration(s) | Delay Until Video Loss (s) | Video Loss Lasted Until |
|---|---|---|
| 60 | 12 | Interference Stoppage |
| 90 | 12 | Interference Stoppage |
| 100 | 12 | Interference Stoppage |
| 105 | 0 | System Restart |
| 120 | 0 | System Restart |
| 180 | 0 | System Restart |

Table 4.4: Acoustically induced video loss in recordings from a EZVIZ surveillance camera system.

**Video Surveillance System Setup:** The attacked system is an Ezviz 720p 4-channel video surveillance system using its stock Western Digital 3.5" Purple 1 TB, part of Western Digital's surveillance series of HDDs. The system stores its operating system on an onboard flash chip, and so the operating system is not directly affected by vibration. The system lies in an acoustic isolation chamber as described in Section 4.2.1. The speaker hangs from the ceiling, resting 10 cm directly above the video surveillance system's HDD. I did not tamper with the surveillance system, leaving its casing intact. Lastly, three (of the possible four) cameras were attached to the system, with one camera placed inside of the acoustic chamber and two cameras placed outside of the chamber.

**Attacking the System:** This test subjects the system to the malicious signal for increasing durations (Table 4.4) and records the results. I choose a 6,900 Hz sinusoidal signal at 120 dB SPL using the methods discussed in Section 4.5.1. During the course of the experiment, we monitored the system manually by looking at the live video feed from the system. After concluding the experiment, we examined the recordings from the HDD.

**Results:** For all tests, the observer did not notice any abnormalities in the live video stream, but attack durations longer than 12 seconds caused video loss in the video recorded on the HDD (Figure 4.11 and Table 4.4). There were two observed pathologies. (1) Recordings from periods of interference less than 105 seconds exhibited video

loss from about 12 seconds after being subjected to acoustic induced vibration until the vibration subsided. In contrast, (2) interference for periods of 105 seconds or longer resulted in video loss from the beginning of the vibration until the device was restarted.

These two pathologies coincide with the behavior exhibited by prior tests. The first pathology, with momentary video loss until interference subsides, is thought to be the write throughput blocking effect discussed in Sections 4.3.1 and 4.3.2. The system buffers video data until a certain limit, which in our configuration is about 12 seconds, after which subsequently recorded video is discarded until the drive becomes available once again. When the interference subsides, the system writes buffered data to disk and begins operation as usual.

The second pathology resembles non-responsiveness errors (Section 4.4). Unlike in the previous case, the HDD becomes non-responsive to the system until the system restart. The system is never able to write the buffered video before being restarted, explaining the immediate effect on the recorded video.

In the case that a victim user is not physically near the system being attacked, an adversary can use any frequency to attack the system. The system's live camera stream never displays an indication of an attack. Also, the system does not provide any method to learn audio in the environment. Thus, if a victim user were not physically near the system, an adversary can use audible signals while remaining undetected.

## 4.6 Defense Design

This section discusses, simulates, or implements several methods to detect or prevent the transduction vulnerabilities detailed in this chapter. With the Transduction Attack Model (Chapter III), one could see how these methods relate to those presented in other work (Table 3.3). Further, one could then compare these against other

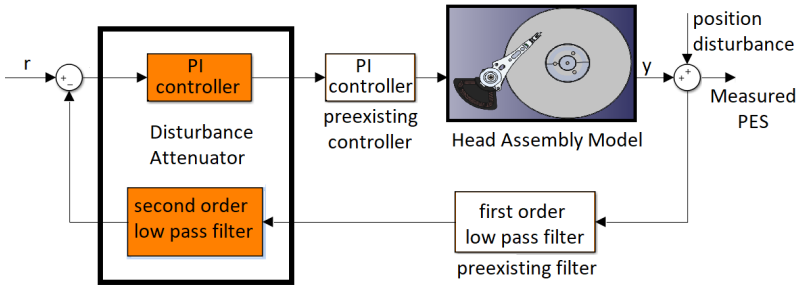



Figure 4.12: The block diagram of the servo control system with the disturbance attenuator composed of a Proportional-Integral (PI) controller and a second order low-pass filter.

transduction vulnerabilities (Table 3.2) to see how these mitigations could affect other attacks.

### 4.6.1 Augmented Feed-Back Controller

For the read/write head and disk platter vulnerability, manufacturers can use a software update to augment the firmware of the feedback controller with a disturbance attenuator to enable the HDD to operate even while under attack (Figure 4.12). This would be classified in the TAM (Section 3.3) as a prevention method using adaptive filtering in the digital backend.

**Position Error Signal:** This mitigation uses the position error signal (PES), the deviation of the R/W head from the center of the track, to judge attenuation effectiveness. The HDD actively uses the PES to control the read/write head under vibration [164]. The PES varies mainly because of repeatable runout and/or non-repeatable runout. Repeatable runout refers to vibration caused by repetitive operating factors, typically internal to the HDD, such as the oscillation of an imbalanced disk rotating. Non-repeatable runout refers to vibration caused by non-repetitive operating factors, typically external to the HDD, such as the acoustic attacks presented in this paper [78].

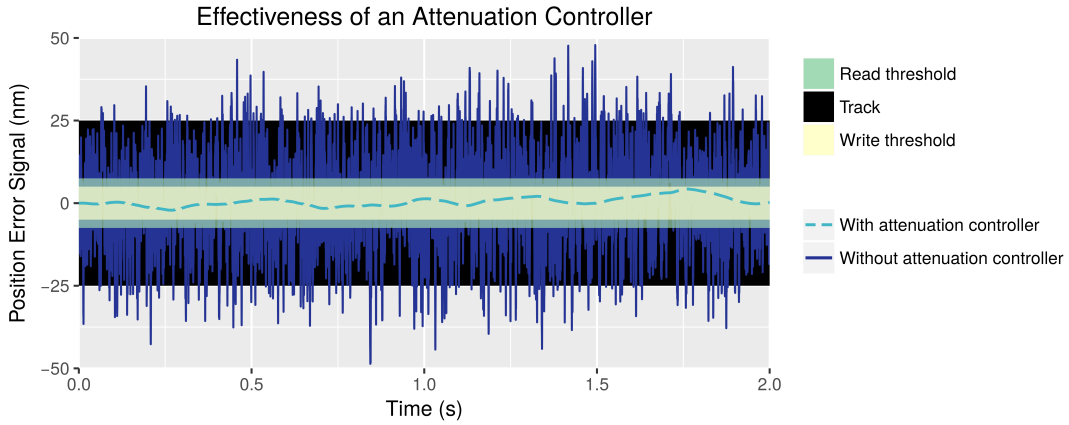



Figure 4.13: Simulated position error variation for a 7.5 kHz attack. Our proposed attenuator reduces position error to within the read/write fault thresholds (15% and 10% of the track respectively).

**Design of an Attenuator Controller:** Attenuator controllers typically compensate for precise, narrow-band peaks in mid-high frequency ranges [69, 139]. However, acoustic signals that affect the R/W head cover a wider frequency range than what is typical for an attenuator controller. Thus, we alter the controller to cover a wider frequency band than what is typical. This modification results in a controller that attenuates a wider frequency band, but with a lower attenuation strength.

**Simulation Model:** I design and simulate a feedback controller with an attenuator for a Seagate 7200.12 HDD that attenuates signals from 6 kHz to 8 kHz, the greatest range that affected the drive (Figure 4.6b). The simulation includes a 9th-order Matlab model of the head-disk assembly and a controller designed using Simulink [2]. The original Matlab model comprises a pre-existing control structure consisting of a first-order low-pass filter in the return path and a Proportional-Integral (PI) controller (Figure 4.12). PI controllers are a common type of feedback controller used in industrial control systems. The PI controller calculates the error value of the head position as the difference between the desired reference setpoint (in this case the center of the track) and the actual position and adds a correction.

Assuming that the pre-existing control sufficiently controls the HDD under normal

operation, fulfilling basic stability and trackseeking requirements, the augmented feedback controller defense adds an attenuator (i.e., another PI controller [P= 0.0079, I=0.1442]) plus a second order low pass filter (transfer function: $[s + 2800]/[s^2 + 128s + 2800]$) to mitigate the attack effect. Its goal is to keep the PES within the read/write fault margins.

The simulation models the disturbance $d$ induced by the attack as a sine wave with amplitude sampled from a uniform distribution, based on real PES data from a Seagate 7200.12 HDD measured during an attack at 7.5 kHz (Figure 4.13). On the non-attenuated controller, this signal induces a displacement up to 97.26% of a track width from the center of the track, well outside of the thresholds for reading and writing to disk (15% and 10% of track width respectively).

**Simulation Results:** The attenuator successfully keeps the PES within the read/write fault threshold within the range of the attenuator. For example, the maximum displacement for a 7.5 kHz disturbance using the non-attenuated controller is 97.26% of the track width, while the maximum displacement when using the attenuated controller is only 8.54% of the track width (Figure 4.13). Similarly, the maximum displacement for a 6.5 kHz disturbance with the non-attenuated controller is 58.36% of the track width, but only 5.12% of the track width with the attenuated controller.

### 4.6.2 Detecting Spoofing Attacks with Filtering or Sensor Fusion

Defenses in the previous sub-section would not prevent spoofing the vibration sensor, but HDDs could make use of filtering or redundant sensors to prevent or detect an ultrasonic attack. If the HDD were to detect such an attack, the drive could operate normally instead of parking the head as a malicious false positive. The key insight for the ultrasonic attacks is that they function effectively because they make use of certain *known* resonant frequencies specified by the physical construction of the component; countering this step can mitigate the attack.

**Notch Filter:** Described in Section 2.2, notch filters can filter a specific frequency while minimally attenuating other frequencies. The purpose of the motion sensors is to detect if the HDD is falling, which is a motion with a frequency much lower than the ultrasonic frequencies discussed in this section. Thus a notch filter that attenuates the resonant frequencies of the sensor while permitting other frequencies should mitigate the attack. This method was inspired by the design pattern in the TAM defense Section 3.3.

**Sensor Fusion:** Also in the TAM defense Section 3.3, sensor fusion could detect or prevent this vulnerability. For example, the HDD could have multiple vibration sensors, each with different resonant frequencies. The attacker would have to emit a tone at all of these resonant frequencies to successfully spoof the system, something made even more difficult with manufacturing variation for multiple sensors.

### 4.6.3 Acoustic Signal Reduction

Reducing the amplitude of acoustic signals is another way to defend against intentional acoustic interference (Section 3.3). Signal reduction approaches are either passive, such as using noise dampening material, or active, such as active noise cancellation. This section implements a passive noise dampening solution, finding it to be effective against higher frequencies but having the drawback of increasing drive temperature. We also discuss active noise cancellation, finding it to be infeasible.

**Passive Acoustic Attenuation:** Many applications use noise dampening materials to passively reduce incoming acoustic signals. To test the viability of noise dampening materials as a defense, we placed sound dampening foam molded into a 4 cm thick block on top of the HDD as described in Section 4.2. We developed acoustic vulnerability profiles with and without the foam block, as shown in Figure 4.14.

Our experiments showed that the foam significantly reduced an HDD's susceptibility to write blocking. However, it did not attenuate lower frequency signals to

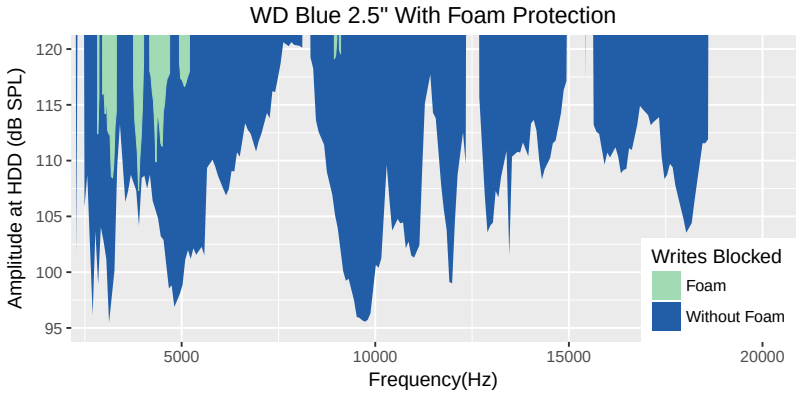



Figure 4.14: The effectiveness of mitigating acoustic interference by simply placing a 4 cm thick piece of foam on top of a HDD.

the same degree as higher frequency signals. This result is likely because of the physics behind how acoustic waves diffract. One could simply encapsulate an HDD with noise reduction materials, but this has one major drawback. Noise dampening material typically acts as a thermal insulator, leading to increases in operating temperature (10 C in our tests). Increased temperature has been linked to increases in drive failure, and thus makes this solution impractical. In addition, this solution can be costly. Depending on the quality of the sound dampening material, this can cost between $10 to $100 per drive.

**Active Acoustic Attenuation:** Noise cancellation may seem like a natural defense against acoustic attacks. However, several difficulties arise when faced with implementing such a defense that would likely make it impractical. It is simple enough to cancel noise along a single plane of points orthogonal to an oncoming wave. However, because of the high frequency of our injected waves, it is more difficult to cancel over an area large enough such that the read/write head is completely enveloped as it moves across the disk [67]. This is not accounting for canceling over the portions of the PCB where the sensors are mounted. In addition, without a high-end microphone, the machine under attack cannot easily determine which direction the sound is coming from without the use of multiple receivers. Lastly, a noise-canceling defense

requires a sound wave equal in amplitude to the attacking wave to completely cancel it, which could be difficult to generate without affecting the hard drive's operation. In combination, these difficulties make us believe that sound cancellation is not a practical defense for a hard disk drive.

### 4.6.4 Other Simple Defenses

There are a variety of other simple techniques that manufacturers or users could apply to defend against acoustic interference on HDDs. The most obvious defense is to use solid-state drives (SSDs) instead of HDDs. However, SSDs remain significantly more expensive per gigabyte than HDDs. Another defense would be to write data to multiple disks spatially spread out in a RAID configuration such that if an attacker simultaneously attacks drives, the system could later reconstruct the lost data from the other drives. If the drives are spatially distant in separately secure areas, denial of service would be significantly harder. Another defense is to simply disable all nearby unused emitters, but this may be situationally impossible or impractical.

## 4.7 Discussion

**Feasibility of Acoustic Attacks:** There are two hurdles for an adversary to cross: the acoustic signal must be strong enough to cause errors and the attack must be difficult to detect or stop. For instance, the attack in Cuba that allegedly used inaudible ultrasonic waves to damage US diplomats' hearing would be an example of being difficult to detect. The attack would also be difficult to stop; no one has claimed to have found any ultrasonic emitters.

Ultrasound may remain unnoticed by those in the vicinity of the attack despite the strength of the signal, as ultrasonic waves are inaudible to humans. Near ultrasonic attacks may remain unnoticed because of high-frequency hearing loss occurring in

human beings, caused by factors including age and poor choice in music.

An adversary may attempt an attack when a victim steps away from a computer. A malicious program or webpage might only play audio when people are likely to be present. If the program or webpage is targeting a specific person or group of people, it could utilize specific knowledge of that group to target times they are not around. Our tests have measured a Dell XPS 15 9550 laptop's output to be as high as 103 dB SPL from 1 cm away from the laptop. We have observed write blocking using signals as low as 95.6 dB SPL. This demonstrates the possibility of using the laptop's own speakers to attack its own hard disk drive.

Beamforming or concealing a speaker can make the speaker harder to locate and harder to stop. For example, a beamforming Long Range Acoustic Device could target a device from a distance greater than 1 mile and may cause malicious effects before the victim would be able to find the emitter.

**Acoustic Attacks in Data Centers and Medical Devices:** In a private data center, the environment is controlled by a single entity and the systems often have no co-located speakers to mount a self-stimulation attack. Companies or individuals can rent a rack, cabinet, cage, or room in a co-located data center. Thus, in a co-located data center, an adversary could pay to place a speaker next to other targeted machines. However, the speaker would need to produce inaudible ultrasonic waves because of constant datacenter monitoring.

Medical devices require high availability. However, in most hospitals and other medical buildings, there is typically an abundance of people, making it difficult to attack with audible frequencies. In the chaos of a hospital or other such building, it may be possible to conceal a device on one's person, but it may also be just as easy to cause a denial of service in other ways without the need of such equipment, such as by unplugging cables. However, acoustic attacks could cause a denial of service through more sophisticated means that leave little traceability back to the adversary.

## 4.8 Conclusion

Adversaries without special-purpose equipment can cause errors in the hard disk drive using either audible or ultrasonic acoustic waves. Audible waves vibrate the read/write head and platters; ultrasonic waves alter the output of the HDD's shock sensor, intentionally causing the head to park. These errors can lead to operating system level or application level consequences including persistent corruption and reboots. Defenses include mitigating attacks in vulnerable frequency bands with attenuation controllers, using sensor fusion to detect attacks, and noise dampening materials to attenuate the signal.

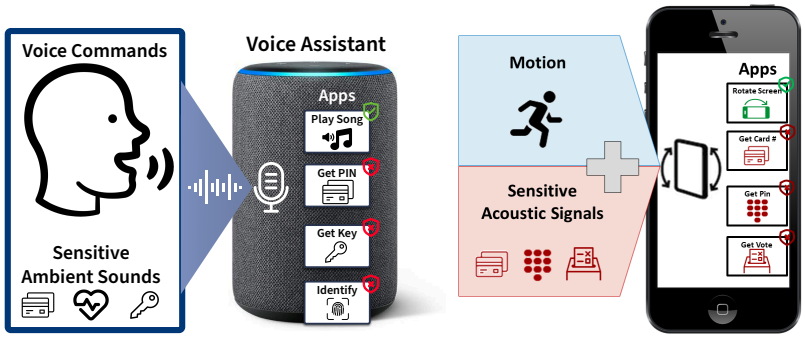
# CHAPTER V

# Oversensing Anti-system: a Smartphone Permission System to Mitigate Oversensing

This chapter, extending ideas in my previous work[1], introduces the Oversensing Anti-system, OA-Sys, to detail common models of oversensing in smartphones and provide a smartphone permission system that provides mitigation patterns for these common models. The fundamental challenge for mitigating oversensing is determining exactly what information sensor data contains and then separating desired from unnecessary information within the sensor data.

Sensor data often has subtle traces of sensitive information hidden within benign information due to the limitations of physical sensor construction. For example, a voice assistant application with microphone access also gathers the sounds of a user inserting a physical key into a nearby door. An adversary could use these sounds to reconstruct the physical key and access the door [108] (Figure 5.1a). While still sound, there is a clear mismatch between the intention of granted permission (speech) and given information (physical door access). Even if the hidden information in sensor data is determined, separating this sensitive information from applications' desired information can also prove challenging. Furthermore, combining data from

[1]Connor Bolton, Kevin Fu, Josiah Hester, Jun Han. "How to Curtail Oversensing in the Home," In *Communications of the ACM*, 63(6): 20-24, May 2020.




(a) Oversensing with Microphones (b) Oversensing with Accel.

Figure 5.1: Granting sensor permissions may provide hidden, unrequired, sensitive information to applications.

multiple sensors can result in emergent sensitive information that would be even more difficult to detect and eliminate, and might enhance an existing vulnerability. In short, current permission systems have no way of knowing or monitoring what information applications receive from sensor data, reminiscent to how until recently iOS had no way of knowing some applications were in fact closely monitoring clipboard data [47].

OA-Sys's solution to the above challenges is to better implement the well-established Principle of Least Privilege [118] for sensor-derived data via sensor permission additions generally applicable to sensor-driven devices, specifically applied to the Android OS. Additions are focused on providing highly defined sensor-derived information via a permission (e.g., transcribed speech), rather than raw sensor data with a higher chance of containing unnecessary, sensitive information (e.g., audio containing physical key insertion). In doing so, the use of these augmented permissions significantly lessens the risk of oversensing without reducing application functionality.

This chapter investigates three common models of sensor data usage then design a permission and demonstrative application for each model. These models include: (1) the user initiated event model with a QR code reader example application, (2) the event recognition model with a hotword recognizer example, and (3) continuous sensor data streams model with a live compass example. We find that oversensing risks are easier to reduce by design for more well-defined, sensor-derived information

such as with the QR code (a website link) or a hotword recognizer (an interrupt or flag indicating the hotword was recognized). However, other oversensing risks where the sensor-derived information is difficult to define are much more difficult to mitigate, such as the live compass application. In these cases, we rely on established signal-processing design meant to mitigate specific types of oversensing (i.e., Dolphin Attack [165]), which can be related to the transduction vulnerabilities in Chapter III. The signal-processing design analysis is part of OA-Sys but separated into a dedicated chapter in Chapter VI. Importantly, we describe how these permission capabilities can be added to current permission systems without removing existing functionality, supporting legacy applications.

We describe, implement, and evaluate anti-oversensing permission designs via OA-Sys, custom Android OS implementation that builds upon recent work [54]. In this implementation, we build our three demonstrative applications and their necessary permissions. The QR code reader application can function without ever receiving camera data, instead of receiving the encoded QR string (e.g., a website URL) from the permission system. The hotword recognition application can always detect when a certain phrase is uttered, without any microphone permissions or receiving audio in any form. Our contributions include:

- **Defining Oversensing Foundations:** This chapter defines and categorizes sources of oversensing, why these vulnerabilities can be difficult to mitigate, how advanced adversaries can improve existing vulnerabilities and the goals for mitigating oversensing.

- **Designing Anti-oversensing Privileging Systems:** We describe how to augment existing privileging systems and design anti-oversensing permissions for three common sensor uses — (1) user initiated events, (2) event recognition, or (3) continuous sensor data streams. New permissions are designed to

limit raw sensor access and unneeded sensitive information while maintaining functionality.

- **System Implementation and Evaluation:** We build an implementation of the permissioning system with OA-Sys and evaluate this Android OS implementation along with our three applications (1) a QR code reader, (2) a hotword recognizer, and (3) a compass. We also evaluate the system against a novel multi-modal attack system we design to test the permissioning and demonstrate the configurability of OA-Sys.

## 5.1 Oversensing Vulnerabilities

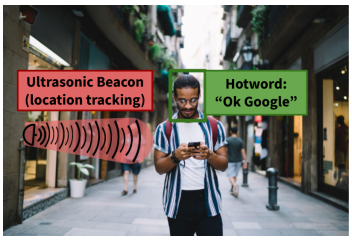
In the context of smart devices, we define *oversensing* as when an application has access to sensor data that contains unnecessary information to complete the application's task and an *oversensing vulnerability* as when that information contains potentially sensitive information. In some cases even if a signal remains unrecoverable to an attacker, the mere presence of the signal serves as sensitive information. The definition for oversensing is left purposely broad as knowing the potentially sensitive information contained in raw sensor data is an extremely difficult and vague task. In this section, we discuss categories for how sensitive information can be introduced in sensor data and a demonstration of mitigation difficulties via a dial-tone keylogger vulnerability.

### 5.1.1 Origins of Sensitive Information in Sensor Data

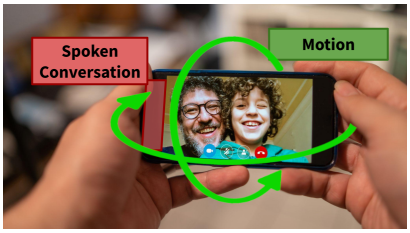
We divide sensitive information in sensor data into three types (Figure 5.2):

1. **In-scope:** information with the sensor's intended specifications (stimulus, frequency, amplitude, etc).


(a) In-scope: QR Codes




(b) Out-of-scope: Hotwords




(c) Unintended Stimuli: Motion Sensing

Figure 5.2: Categories of oversensing vulnerabilities by how sensitive information can be embedded in sensor data. These are: (a) in-scope, (b) out-of-scope, and (c) unintended stimuli oversensing vulnerabilities demonstrated through (a) QR code, (b) hotword, and (c) motion sensing example applications.

2. **Out-of-scope:** information originating from the sensor's intended stimulus, but outside the sensor's intended scope (e.g., outside the frequency range).

3. **Unintended Stimuli:** information originating from a stimulus the sensor should not transduce.

These types can impact what mitigations can remain effective for them. For example, some of the mitigations discussed in Chapter III are relevant for out-of-scope and unintended stimuli categories, but cannot mitigate in-scope vulnerabilities.

#### 5.1.1.1 In-scope

As shown in Figure 5.2a, sensitive information with the same characteristics (e.g., stimulus, frequency, and amplitude range, etc) as benign, intended information can be contained in sensor data given to an application (e.g., a credit card in camera data). While providing both sensitive and benign data is within specifications, an ideal system provides only benign information to applications.

Consider the QR code example. A QR code is a 2D grid of black and white squares used to encode some message visually, typically a link to a website or to give local access to some device or network. The code itself contains a simple link or local password but camera access is required to acquire this data. This permission may

unnecessarily provide the identities of nearby people, the user's location, sensitive data from local documents such as credit cards. However, the system cannot eliminate the sensitive data as benign applications may require it to function correctly. Currently, permissioning systems do not have a way to rectify the clear mismatch between the needed data and the given data as seen with QR codes.

#### 5.1.1.2 Out-of-scope

As seen in Figure 5.2b, sensor output may also contain traces of sensitive signals from intended stimuli that are outside the sensor's intended scope (i.e., outside the intended frequency range). As sensors are only designed to accurately capture signals within their range, these signals are often distorted or significantly altered in an unrecoverable manner. However, an attacker needs to only discern the presence of these signals to acquire sensitive information.

Consider a hotword application. A hotword is a designated phrase such as "Ok Google" that triggers an action when spoken. An application that desires this functionality would need undisturbed microphone access — clearly a dangerous permission. Any spoken information (which is in-scope), including passwords, credit cards, and more, would be readily available to a malicious application. However, this may also lead to out-of-scope vulnerabilities. For example, a microphone can often receive human-imperceptible ultrasonic acoustic signals, outside typical microphone frequency range specifications. Exploiting this out-of-scope interaction between microphones and ultrasound can lead to attacks such as location tracking [13].

#### 5.1.1.3 Unintended Stimuli

Sensitive signals from stimuli a sensor is not designed to measure may still affect the sensor's output, stealthily placing sensitive information in sensor output such as seen in Figure 5.2c. Furthermore, within sensor circuitry the information may

appear to be either an in-scope or out-of-scope signal in terms of characteristics such as frequency, amplitude, etc. These stealthy relationships enable an adversarial application to request one permission to sense something that should normally require separate permissions.

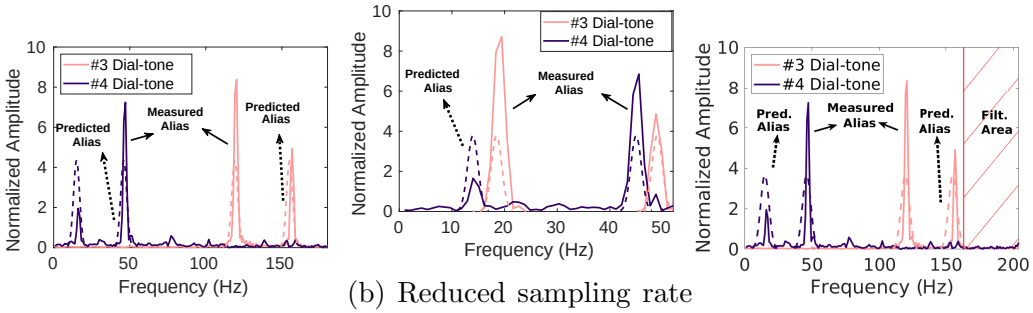
Consider a motion-sensing application such as a compass. This common feature uses multiple sensors including a magnetometer, accelerometer, and gyroscope to calculate true north. However, these sensors have been shown to perceive nearby speech. This would be a clear mismatch between intended use (motion and direction) and given information (nearby speech). Other applications with motion sensor data, such as motion-reactive user interfaces, could similarly use this attack.

### 5.1.2 Why Oversensing Mitigations are Difficult to Build: Touchtone Keylogger Example

Oversensing vulnerabilities are difficult to mitigate as hidden, sensitive information can be difficult to detect, discern, and differentiate from benign information; furthermore, designing methods to eliminate this sensitive information while preserving the benign information in its entirety is also a challenge in itself. Furthermore, any remaining discernible byproducts are sufficient for an attack. To highlight some of these mitigation difficulties we provide a short feasibility study using a touchtone keylogger (Expanded upon in Chapter VI), then discuss how an advanced attacker could improve oversensing attacks including the keylogger.

#### 5.1.2.1 Dial-tone Keylogger

This motivational oversensing example infers a victim's dial-pad input when navigating through an automated call system on a smartphone at high accuracy using motion-sensor access. Many automated phone systems require the user to input sensitive information such as a credit card number (for activation), a personal pin, an




(a) No defense (b) Reduced sampling rate defense (c) Digital low-pass filter

Figure 5.3: A short study on why oversensing may be difficult to mitigate. Two dial-tones are emitted from a Google Pixel 2's loudspeakers. The phone's accelerometer's x-axis is shown (a) with no defense, (b) with a reduced sampling rate, and (c) a digital low-pass filter. The aliases for the dial-tones remain distinguishable despite the defenses, meaning the vulnerability is not mitigated.

account number, a social security number, and in some cases votes in a federal election (done by phone) [150]. This keylogger is based on how dial-tones, also known as dual-tone multi-frequency (DTMF) signals, produced by a phone's speaker affect the phone's motion sensor output as discussed in Section 2.1. The dial-tones produce distinguishable, deterministic byproducts in the motion sensor output (Figure 5.3a) called aliases. The alias distinguishability can be used as the basis of an oversensing attack as detecting an alias indicates a press of the corresponding key. Detailed further in Section 5.4, this idea was used to create an attack with accuracy greater than 99%.

#### 5.1.2.2 Aliasing

As noted, aliases and aliasing are one of the core reasons the dial-tone keylogger works correctly and is a core component for many out-of-scope and unintended stimuli oversensing vulnerabilities. Aliasing can create discernible, deterministic byproducts called aliases for sensitive signals with frequencies higher than the sensor's normal frequency range. To be correctly sampled, a sensor must sample a signal with frequency $f_S$ at or above $2f_S$, which is called the Nyquist frequency $f_N$. An attacker may use

the presence of these byproducts to indicate the presence of the original signal.

Aliasing can make signal processing-based oversensing mitigation significantly more difficult. Take for example two mitigations suggested in previous literature motion-sensor eavesdropping attacks with reduced sampling rates and digital low-pass filtering. As shown in Figures 5.3b and 5.3c, the aliases for the dial-tones remain distinct despite the mitigations. Therefore the vulnerability remains unmitigated (detailed in Section 5.4). In this specific mitigation, the filter should attenuate all frequencies from 150-200 Hz, however, the aliases were produced below the 150 Hz threshold (Figure 5.3c). One could lower the threshold, but doing so attenuates even more benign information. A 150 Hz threshold attenuates 1/4 of the available range in this specific example, yet has little to no effect on these specific dial-tones.

#### 5.1.2.3 Improving Oversensing Attacks via Adversarial Sensor Fusion

We propose that advanced adversaries can improve existing attacks or create new oversensing vulnerabilities by selectively integrating data from multiple sensors for a single oversensing attack. This idea is based on the intuition that more sensors should yield more total information that can be used to create emergent oversensing vulnerabilities. This same idea, using multiple sensors to reveal emergent information, has been used by researchers for benign purposes in several fields including on drones [73], body-sensor networks [50], and much more. It follows that attackers with sufficient permissions could achieve similar results, though for adversarial purposes.

We find that even adding data from separate axes of the same sensor may carry some measure of unique information. Take for example the response of a phone's motion sensors (accelerometer and gyroscope) to the phone emitting different tones via its loudspeaker (5.4). Each axis of each sensor responds differently to acoustic signals of 470 Hz and 520 Hz, and each axis is more receptive to certain frequencies independently of the other axes. This difference means each axis may be a better

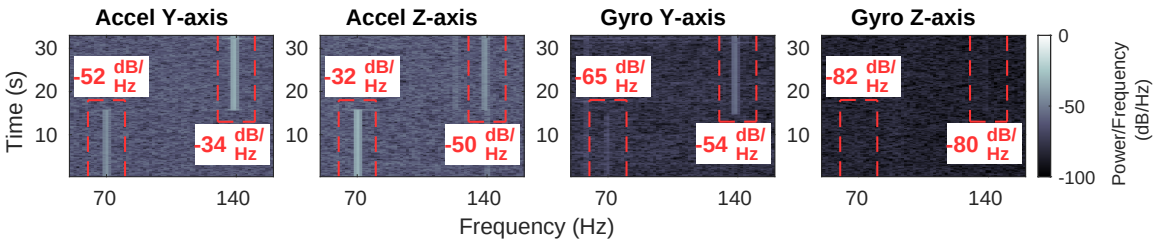



Figure 5.4: Figure depicting how the y and z axes for a Pixel 2's accelerometer and gyroscope may contribute different frequency information to an oversensing vulnerability. A Pixel 2's loudspeaker plays a 470 Hz and then 520 Hz tone each for 15 seconds while the phone records accelerometer and gyroscope readings. Each depicted axis responds to the tones differently due to different frequency responses.

receptor of certain frequencies, which in theory could lead to the axis being a better predictor for something based on those frequencies.

## 5.2 OA-Sys: Designing Anti-Oversensing Permission Models

We present OA-Sys (Oversensing Anti-System) as a framework and real-world implementation (in Android OS) to address oversensing attacks via a permissioning system and configurable signal processing framework. First, as a foundation for OA-Sys, we provide a detailed look at the necessary changes to a permissioning system to mitigate oversensing. We provide high-level permission designs for three common models of sensor usage. The primary goal of these designs is to bring the Principle of Least Privilege to sensing devices by mitigating oversensing while still providing full sensor functionality to applications. Oversensing vulnerabilities in certain sensor use-cases are easier to design out (such as with QR-codes) but are more difficult for others (such as motion sensor data streams for a compass application). We first outline the design goals, constraints, and considerations our design must address. Second, we detail the requirements of a permissioning system to be able to support our specific permission models. Then, we detail the design of specific permissions for the three common models of sensor usage. Finally, we investigate the privacy vs. performance trade-offs of using signal processing in the design of permissions for the

more difficult-to-mitigate oversensing vulnerabilities, which may not have a perfect ideal solution.

### 5.2.1 Defining the Goal: Principle of Least Privilege

The goal for system designers should be to deliver the minimum amount of information to an application while preserving that application's functionality, an idea known as the *Principle of Least Privilege (PoLP)* [118]. In the context of security, this principle's goal is to limit adversary capabilities by limiting the information available to all applications. PoLP is a widely employed principle in software systems, especially operating systems of all kinds. For example, privilege sets, Linux groups, Window's administrator mode, sandboxes, and more all implement this principle to some degree. Smartphones and other smart devices already prevent blatant violations of PoLP via permissioning systems, but distinguishable sensitive information may still subtly leak through to applications.

However, implementing PoLP for sensor data brings several context-specific challenges. For example, how does one define exactly what is the minimum amount of information that an application needs to function? Current permission systems are implementing PoLP to some degree, but it is increasingly hard to know what a sensor's information contains, which makes the current permission systems insufficient. For example, compare audio data to a phone contacts list. Audio data might contain speech, but might also be used to track user location or gain access to a user's home by deciphering their physical door key [108]. The information in the contact list is far more defined than the information in sensor data, but modern systems have similar designs for each type of permission. These challenges make implementing PoLP for sensors non-trivial.

While PoLP is a noble aspiration, real-world constraints make this difficult for sensor-based systems. Implementing real-life mitigation must balance several con-

straints while minimizing sensitive information and preserving application functionality.

- **Effectiveness:** The system must limit sensitive information to applications.
- **Unobtrusiveness:** Mitigations should minimally alter benign sensor signals and require minimal application code changes.
- **Overhead:** The system should require minimal overhead including power loss, signal delay, physical space, additional computational burden, and cost.
- **Ease of Deployment:** The mitigation should be easy for manufacturers to deploy.

### 5.2.2 Augmenting Permissioning Systems

OA-Sys permissioning changes aim to bring the Principle of Least Privilege to sensor permissions by providing applications with only the specific information they need from sensor data while still enabling full functionality (Figure 5.5). These changes *add* functionality to current permission systems without removing existing functionality; legacy applications would need to make no changes to continue their existence (but would not be any more or less secure than before). We detail key requirements for a system to support our augmented permissions.

#### 5.2.2.1 The Key: Limiting Raw Sensor Access

Future permission systems must support permissions that limit raw sensor access to mitigate oversensing. Full access to a sensor's data typically enables far more information than an application needs to complete their task (Figure 5.2). Oversensing attacks can exploit this information gap to create a malicious application that appears to use a sensor's data for one task but utilizes the extra information given by

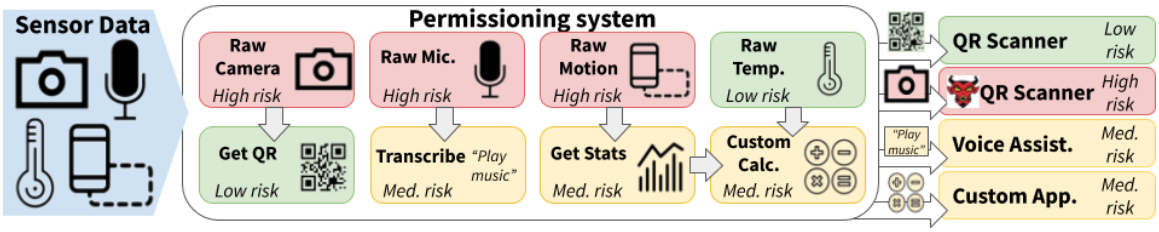



Figure 5.5: Our permission system provides permissions that provide well defined sensor-derived information such as transcription and qr codes. Permissions should have a notion of risk and raw sensor data should be labeled as more risky.

the required permission to accomplish some malicious goal. Once granted information access, they can save this data either on the device or send it to the server (if granted internet access, which is commonly given). Then the adversary can use signal processing or machine learning to search for any latent sensitive information in the given data. In such a scenario, few present-day systems can detect such a misuse of the given data, especially in an offline attack.

The key for new permissions is to better define the information and associated risk of sensor-derived information given to benign programs; systems must support such permissions. Most benign applications use only a fraction of the total data given by a permission, usually calculating a few specific quantities from the given data. Changing the permission systems to enable permissions that can give specific quantities instead of the totality of the data (i.e., raw sensor data) reduces the amount of data given to each application and reduces overall risks.

.

#### 5.2.2.2 System Design Requirements

We note a few design requirements, which we capture in our Android OS implementation of the permission system.

**Clear Risk Labels:** Each permission and its associated data must be assigned an appropriate level of risk and the system itself should indicate to the end-user

each time moderate or a dangerous permission is used. Use of dangerous privileges (i.e., raw sensor data access), especially in those of background services, should be discouraged but enabled and labeled. A visual, foreground sensor use (e.g., camera data actively displayed on the screen), should be labeled less dangerous than its background alternative. Alerting the user *each* time a dangerous privilege is used is essential.

**Modular, Extendable Framework:** The sensor permissioning system must be able to support custom calculations on sensor data to support future, unseen needs and have the ability to easily add or subtract the code for these permissions in a modular fashion. To do so, application designers could design a permission and submit it to whoever maintains the permission service. As a default to ensure privacy, each new permission could initially be given a risk level of its most dangerous sensor data source. Signed permission modules could be downloaded when an application is installed that needs the new permission.

**Support for Sophisticated Permission Services:** The permission system must support sophisticated permissions that may require active services or activities as part of giving data to an application. Most modern permissions merely give sensor data to applications, but do not do much else. Fundamentally, anti-oversensing permissions would take sensor data and reduce it in some manner. How it reduces the data may be complex, and there may need to be interaction with the user (such as in the case of QR codes). Additionally, a permission may need to periodically send data to one or multiple applications using the permission. To complete these tasks the permission itself may wish to run a brief foreground activity or background service that is always on. These permissions may need expressive API functionality, with applications having the ability to send or receive a wide variety of parameters to support feature-specific permissions.

#### 5.2.2.3 Performance Benefits

As a brief note, a permissioning system that limits raw sensor access and promotes processed data instead may have several performance benefits in addition to security. The first is that this may ease application programmer burden, where instead of having to manually transform sensor data, they can simply request said data from validated, optimized, core, or OS level APIs. This shared API will reduce application size if multiple applications use the same API, will enable hardware acceleration support, and could reduce energy consumption.

### 5.2.3 OA-Sys Specific Permission Designs

In this section, we describe three common models of sensor data usage and provide high-level designs for permissions to prevent oversensing for each model (Figure 5.6). These designs are specific to OA-Sys, but project to other permission systems. These designs include:

1. **User initiated events:** upon a user's request sensor data is acquired and consumed.

2. **Event recognition:** an application consistently acquires sensor data to recognize a specific event(s).

3. **Continuous sensor data stream:** an application consistently acquires and consumes data.

#### 5.2.3.1 User initiated events

Applications commonly acquire sensor data upon user request to perform some action. In this model, the application acquires sensor data only upon the user's request, which could be the user pressing a specific button or updating data upon the

```
initialization()
eventStatus =
 perms.myEvent.register()
 contStatus =
 perms.myContData.register()
 while applicationRunning do
 |  if userPerformsAction then
 |  |  userData =
 |  |   perms.myUser.request()
 |  |   han-
 |  |   dleUserEvent(userData);
 |  end
 |  if eventStatus.get() then
 |  |  handleEvent()
 |  end
 |  contData =
 |   contStatus.getData() up-
 |   dateSensorData(contData);
end
perms.eventRecognition.unregister()
 perms.contSensorData.unregister()
```

(a) Example of an application using our three privilege models.

```
initialization()
while permRunning do
 |  registeredApps.updateApps()
 |   while registeredApps.num > 0
 |   do
 |   |  registeredApps.updateApps()
 |   |   sensorData =
 |   |   sensors.getData()
 |   |   processed =
 |   |   sensorData.process() for
 |   |   app: registeredApps do
 |   |   |  cond =
 |   |   |   app.eventConditions if
 |   |   |   processed.isEvent(cond)
 |   |   |   then
 |   |   |   |  app.sendEventNotification()
 |   |   |  end
 |   |  end
 |  end
end
```

(b) Privilege model for event recognition.

```
initialization()
appInfo = getRequestingApp()
 sensorData = sensors.getData()
 processedData =
 sensorData.process()
 appInfo.sendData()
```

(c) Privilege model for user initiated events.

```
initialization()
while permRunning do
 |  registeredApps.updateApps()
 |   while registeredApps.num > 0
 |   do
 |   |  registeredApps.updateApps()
 |   |   sensorData =
 |   |   sensors.getData()
 |   |   processedData =
 |   |   sensorData.process() for
 |   |   app: registeredApps do
 |   |   |  app.sendData(processedData)
 |   |  end
 |  end
end
```

(d) Privilege model for continuous sensor data.

Figure 5.6: This shows (a) an example of using the privilege model to collect data, and privilege models for (b) user initiated events, (c) event recognition, and (d) continuous sensor data.

user entering or leaving an activity. This can include a user requesting camera access to read a QR code, such as in our demonstrative application, or requesting a current GPS location.

Permissions for the user-initiated one-time events act as a simple function at a high level, providing a result when called (Figure 5.6c). When the application receives user input that requires the sensor-derived information, it calls the required permission with necessary parameters. This permission, outside of the application, starts its process and accesses the needed sensor data via the OS. Getting the sensor data may require user interaction. The permission then performs the operation needed to reduce the sensor information into its minimal form, which should be clearly defined (such as providing a QR code string) information. This information is then returned to the application and the permission's process is ended. Essentially, the permission system (OA-Sys) sandboxes the sensitive function and is a mediator between the raw sensor data and the application.

#### 5.2.3.2 Event Recognition

A second common model in which applications use sensor data is intermittent event recognition. In this model, an application gathers data periodically and checks the data in some manner for particular events. This periodic checking can be started or stopped by the user. This can include functionality such as hotword recognition, such as in our demonstrative application, or gesture recognition.

Permissions for continuous event recognition act as a service that sends an interrupt to a registered application when the event is sensed (Figure 5.6b). When an application wishes to be notified of an event that is recognized via sensor data, they register with the event permission and give any associated parameters. If an application has registered with the permission service, it runs a separate process outside of application space, in the OS, that periodically gathers the necessary sensor data to

check for the event. The process checks to see if any of the registered applications' events would be triggered (each could have separate parameters such as different phrases for hotword recognition). If any events are sensed, it sends the corresponding registered application a notification that the event was sensed. The service will continue notifying all registered applications each time the event is sensed until all applications have unregistered, during which the service turns off its process loop.

#### 5.2.3.3 Continuous Sensor Data Stream

In addition to the two previous models, it is common for applications to display, consume, or log constant streams of data, for example in activity tracking or step counting. In this case, the application receives a similar or the same amount of data as in the continuous event recognition model; however, the intent of the application is different in that it intends to use all the data, more than simple event recognition. Thus most applications that log some form of sensor data, display sensor data in real-time, or feed sensor data to a machine learning model in real-time would fit in this category.

Similar to the event recognition scheme, applications register with the privilege model for the continuous sensor data stream through OA-Sys (Figure 5.6d). However, because specific events are not being captured, OA-Sys must do a general set of signal processing to reduce the privacy sensitivity of the data. Straightforward approaches could include averaging and filtering out voice bands to reduce eavesdropping. However, this can be difficult to design, which we discuss in more detail in the next section. This processed sensor data is sent to the application periodically, and the application de-registers the permission when it is done collecting. OA-Sys manages these background services and kills them when they are no longer needed, and also shares this data to applications that request it, to reduce duplication of work.

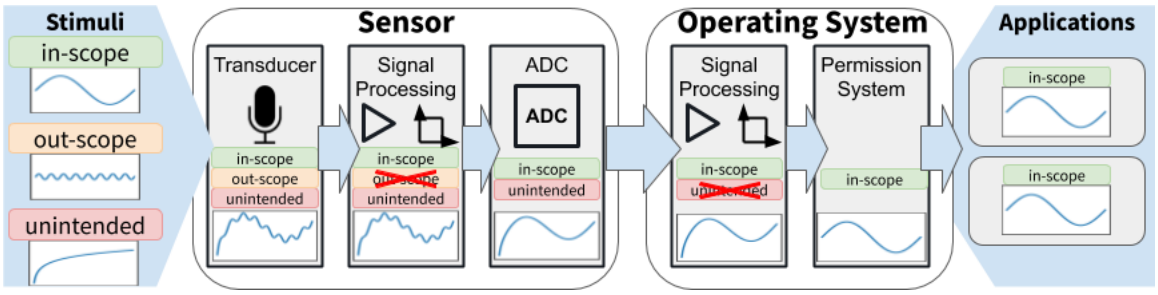



Figure 5.7: An example showing how signal processing should attempt to deliver the in-scope signal to the permissioning system. Software and hardware mitigation should eliminate signals produced out-of-scope and by unintended stimuli but may be unable to mitigate in-scope oversensing without eliminating benign data. Specific permissions may be able to further reduce the in-scope information.

### 5.2.4 OA-Sys Signal Processing Permission Design

Signal processing is a likely candidate to address many oversensing vulnerabilities. Many of these mitigation techniques can be shared from those presented in Chapter III. In this section, we discuss crucial considerations in designing specific mitigations such they can be easily deployed into existing systems without affecting benign application behavior. Chapter VI focuses and expands upon signal processing mitigation design for oversensing.

As there are many different sensors and applications, and new attacks are uncovered frequently, a general framework is required versus a single solution. Signal processing solutions can have different impacts based on their location (Figure 5.7), each with its benefit. I examine the most important features and pitfalls of oversensing signal processing mitigation design to aid manufacturers and operating system designers in building a specific defense for their scenario including where the signal is processed, distortion, and hardware variables[2].

[2]For signal processing, we consider hardware and system changes that affect sensor output to be part of the permission design as those changes can alter the way certain permissions are designed.

#### 5.2.4.1 Signal Processing Placement

Signal processing can happen in hardware blocks, OS, permission systems, or individual permissions.

**Inside or Before Individual Permissions:** Signal processing placement alters its design constraints and possible effectiveness (Figure 5.7). If a mitigation is placed in individual permissions the solution can be tailored to each permission. Individual permissions are more aware of what sensor information is important to its provided functionality and thus can better tailor the signal processing mitigation to remove specific oversensing vulnerabilities without removing the crucial information. This means that signal processing mitigations may be able to eliminate in-scope, out-of-scope, and unintended stimuli sensor information without functional penalty.

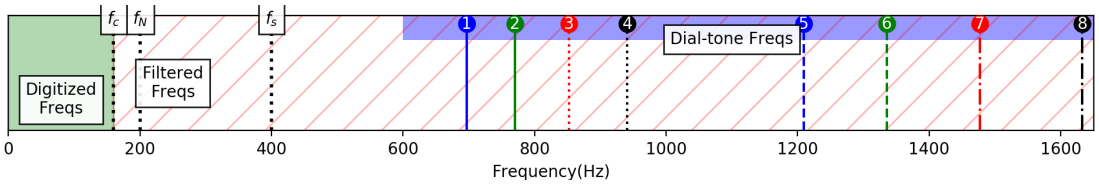
Conversely, placing the mitigation before the individual permissions, in the system or hardware, applies the signal processing mitigation to all permissions that use that sensor data. This has the advantage that permission designers need not worry about mitigating that particular vulnerability, and reduces the capability of malicious actors. However, designing such solutions is more difficult as little is known about the intended use of the data. Thus, these mitigations must preserve all in-scope information to not affect benign application performance. They should otherwise attempt to limit the sensor data to the sensor's intended use (e.g., an accelerometer should measure only acceleration) by eliminating out-of-scope and unintended stimuli information.

**Analog vs Digital:** One crucial consideration in a filter implementation is whether to implement a digital or analog filter. Generally, digital filters are easier to implement and deploy as digital software filters can be given as a software update. However, analog filters circumvent some of the crucial issues faced by digital filters (e.g., aliasing) while perhaps needing extra hardware support (Figure 5.8a).

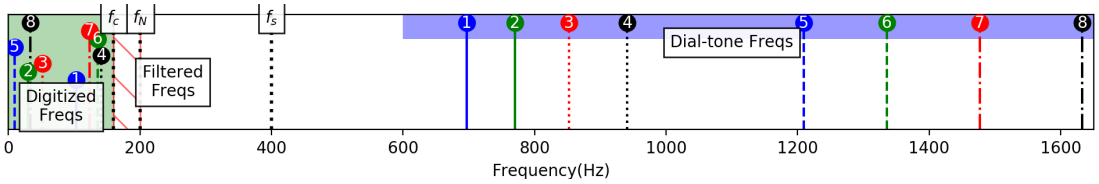
Software digital filters are largely desired due to the ability to easily deploy the mitigation as an update, such as a permission update in OA-Sys. These mitigations can be configured and further updated over time should other vulnerabilities become known. Additionally, they require no extra hardware overhead such as additional physical space or latent power draw (when unused). Thus, effective software mitigations are of great value when available. For this reason, we only consider mitigations that may be implemented as a software update in our evaluation.

**Mitigating Aliasing Based Oversensing:** Aliasing is a major source of oversensing vulnerabilities (Section 5.1.2.2), and something all filters must consider to eliminate oversensing. Aliasing is not a large problem for analog filters (Figure 5.8a) as aliasing takes place after the signal is digitized and analog filters work on pre-digitized signals. However, digital filters are often desirable over analog filters due to the aforementioned deployability benefits.

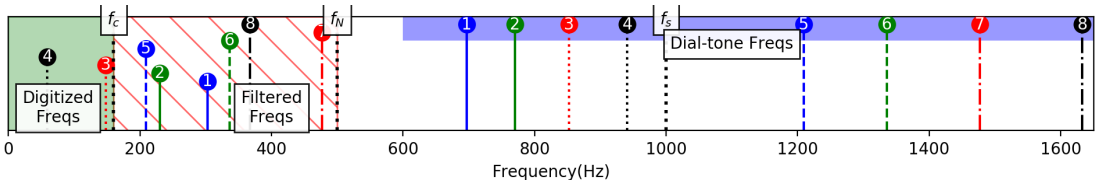
To enable effective digital filters, the sensor must sample at a rate greater than what the system intends to give to applications. This act of sampling at a rate faster than what will eventually be given is often called oversampling. As shown in Figure 5.8b, aliasing will produce byproducts in sensor output for any signal with a frequency above $f_N$, which is $f_S/2$. Mitigations should aim to provide applications with minimally altered benign information in the sensor signal. Filtering large parts of the expected bandwidth, or range of frequency information, would violate this principle. However, without oversampling low-pass filters must filter large parts of the expected bandwidth to attenuate sensitive signals such as dial-tones used in our example (Section 5.1.2). Yet, if the signal is oversampled (Figure 5.8c), this problem will be alleviated to a certain degree. This is due to how the oversampling will alter the aliases of these sensitive signals, possibly placing them above the desired bandwidth to provide to applications. This is a basic idea behind anti-aliasing filters, which are two of our tested mitigations.




(a) LPF analog defense




(b) No Oversampling




(c) Oversampling

Figure 5.8: How analog and digital filters handle the 8 dialtone frequencies used in the keylogger (Section 5.1.2). (a) Analog filters work on pre-digitized data, and thus circumvent the problem of aliasing. (b) Digital filters without oversampling cannot filter sensitive digitized signals (in this case dial-tones) without attenuating a significant region of benign signals (green area). (c) With oversampling there is a greater chance sensitive aliases may be outside of the range of benign signals (green area) and can be filtered digitally.

Oversampling should be possible in most smartphone hardware, as the sampling rate of the sensor is limited by the operating system due to power constraints. An operating system update could enable the oversampling and then filter the signal. Then to provide a data rate (number of samples per second) the same as before, the system can down-sample the signal to the original data rate.

However, oversampling does have a few drawbacks. The downsampling step may introduce minor distortion in the signal unless the oversampled rate is a multiple of the original rate. Additionally, oversampling will likely slightly increase the power consumption of the sensor. Last, implementing oversampling in software will have memory and computational overhead may burden resource constrained systems.

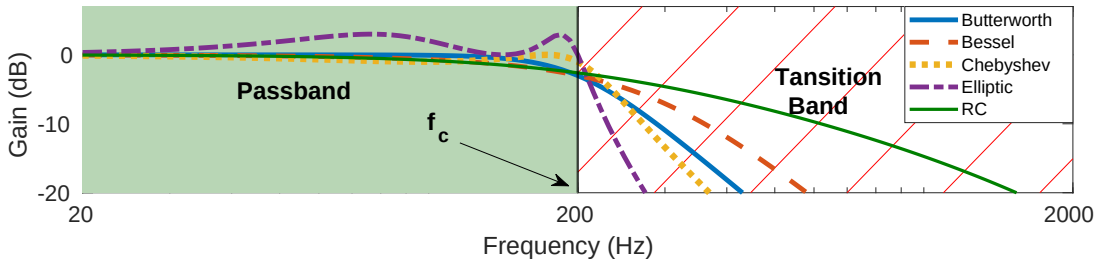



Figure 5.9: Filters trade performance vs. distortion. Filters with a flat passband minimally distort benign application data, but filters with sharper cutoffs may be more secure against oversensing vulnerabilities.

#### 5.2.4.2 Distortion

Most filters in hardware or software will cause some distortion of the benign information in the sensor data. Depending on the use case, more or less output distortion may be permissible. The filter's design will have a large effect on the distortion of the signal. There are several established digital and analog filters. We identify five common choices for various applications (Figure 5.9). Our implementations use a Butterworth filter for having a good balance of minimal distortion and sharp cutoff to attenuate a larger range of potentially sensitive data. however, other applications may wish to use a different design.

Two other factors that will affect the distortion of benign data are the filter order and cutoff frequency. Both parameters can be fairly application-specific. Cutoff frequency limits bandwidth, and thus should likely never be much less than the system's original sampling frequency. Otherwise, this risks misleading and harming benign applications. However, the frequency should likely be close to the Nyquist frequency in some way to maximize the attenuation for stopband frequencies.

Filter order balances how sharp the filter roll-off is with cost, size, and real-time delaying detriments. The higher the order, the sharper the transition band is, and thus the more of the intended range of frequencies to filter will be filtered. However, in hardware a higher-order requires more components, which may not be permissible given cost and space limitations.

#### 5.2.4.3 Hardware Based Signal Handling.

Other miscellaneous hardware changes may also significantly reduce oversensing vulnerabilities. In particular, recent work has investigated how hardware changes may mitigate malicious signals from unintended stimuli [161]. Many of these changes could include sensor redesign. For example, there are microphones designed specifically to be resistant to light affecting the audio output. There are also thermal accelerometers designed to be resistant against high-frequency vibration that may significantly alter the output. Unfortunately, hardware changes cannot be implemented via a software update and will require a significant redesign of the system. However, they may enable long-term solutions to mitigate many oversensing vulnerabilities.

## 5.3 OA-Sys Implementation

We built OA-Sys as a proof-of-concept of a permission system for oversensing, targeting Android OS 9 and built on Privacy Enhancements for Android (PE for Android)[3] This section describes two implementation efforts towards building and evaluating OA-Sys (detailed in Table 5.1):

1. **OA-Sys: the permission system.** This includes the additions to PE for Android to provide a framework for adding new permissions related to oversensing, and new signal processing techniques at the OS level.

2. **OA-Permissions: the individual permissions.** The individual permissions and associated services are designed for a QR code, hotword, and compass application, fulfilling outlines given in Section 5.2.3.

The PE for Android implementation was chosen because of its adoption and support by industry partners and the research community, including multiple efforts

[3]To learn more about Privacy Enhancements for Android, see `https://android-privacy.org/`

| Name | Description |
|---|---|
| OA-Sys | Android OS designed to enable permissions to mitigate oversensing. |
| OA-Permissions | Short running functions that provide reduced sensor data to applications |
| OA-Services | Trusted services that assist OA-Permissions with complex functionality. |
| QR Code Application | Demonstrative application for user initiated events. |
| Hotword Application | Demonstrative application for event recognition. |
| Compass Application | Demonstrative application for continuous sensor data streams. |

Table 5.1: Systems, permissions, applications, and case studies used to implement or evaluate OA-Sys.

by researchers to enhance Android privacy building on the platform [74, 93, 59]. Evaluation for signal processing mitigations for the compass application appears in Chapter VI.

### 5.3.1 Permission System

Our permission system extends the implementation of the Privacy Enhancements for Android [54, 59] to support the requirements listed in Section 5.2.2. Figure 5.10 displays how the permission service provides applications data. Each non-standard component is described below. OA-Services are the core hierarchy addition from the original PE for Android implementation. OA-Services which are trusted userspace services that permissions can interact with for complex operations. This system is deployed as a custom Android OS image.

#### 5.3.1.1 Private Data Request

A private data request is a request for (sensor-derived) data through our implementation system or a signal to communicate with a permission service that would provide such data. These requests come in three types: (1) a request for a singular instance of data, (2) a registration request for permission service, and (3) an unregistration request for a permission service. Each request specifies the data it would like, the purpose for requesting the data, a result listener to receive the resulting sensor

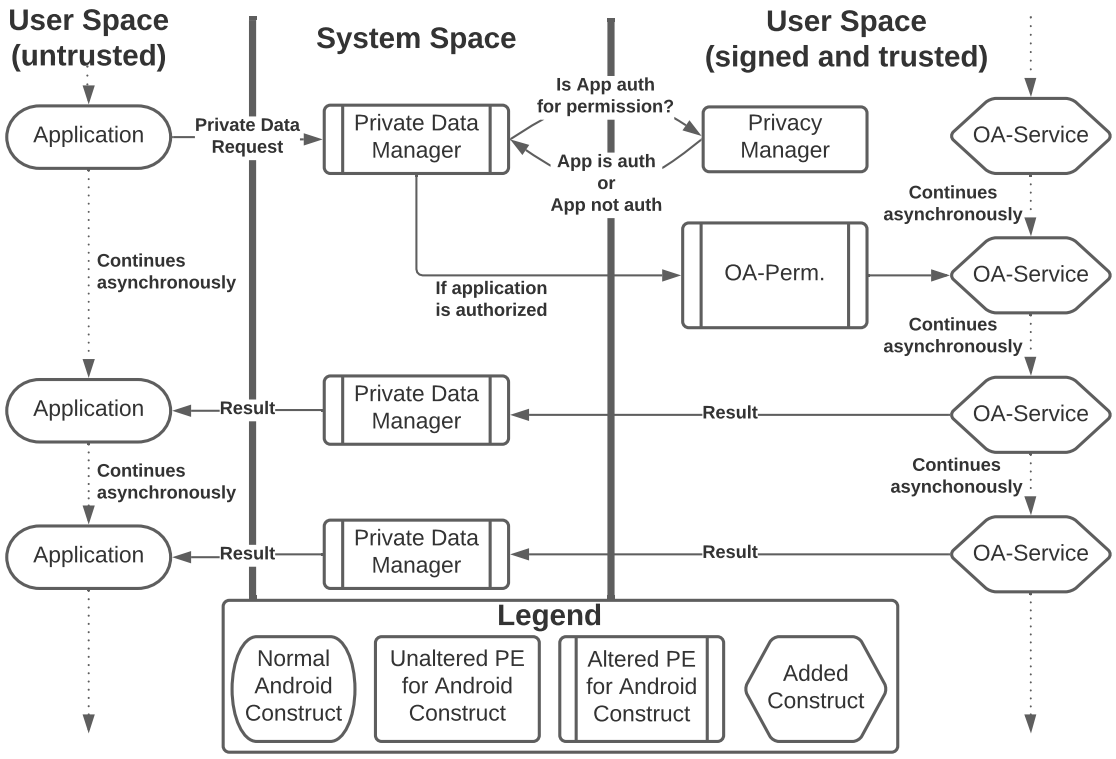



Figure 5.10: How a permission with an OA-Service request resolves in the Android implementation, highlighting alterations and additions from Privacy Enhancements (PE) for Android. Private data requests can include a request for a singular instance of data, registering for a permission service, or un-registering for a permission service.

data, the specific OA-Permission to transform the data, and any permission-specific parameters. The private data request remains relatively unchanged from the PE for Android version; however, regular PE for Android did not have OA-Services and thus no communication with those services.

#### 5.3.1.2 Private Data Manager

The private data manager is a system API in PE for Android that facilitates communication between applications and OA-Permissions. This service registers and manages all OA-Permissions while also listing the available permissions to applications. When an application makes a private data request, the private data manager gives the request to the privacy manager and receives a decision on whether to al-

low or deny the request. With a denial, nothing happens except the application is notified. When a request is allowed, the private data manager calls the appropriate OA-Permission with any needed parameters or data.

In our implementation, the Privacy Manager also provides the permission with the request's result listener. This is the primary and only code change to PE for Android's private data manager implementation. This allows OA-Permissions and services more flexibility in fulfilling requests, such as an OA-Service sending results continuously to any registered applications. This is the crucial change that enables OA-Services.

#### 5.3.1.3 Privacy Manager

The privacy manager is a trusted userspace application that permits and denies requests for sensor-derived data. Through the Android System Settings, the user can select the active privacy manager. Only one privacy manager can be active at a time and the privacy manager must be signed for the system to permit its use. Our implementation does not change the purpose or underlying code of the "Privacy Checkup" policy manager from its original implementation in PE for Android. The manager allows users to check which applications have which permissions.

#### 5.3.1.4 OA-Permission

An OA-Permission is the core of OA-Sys, either fulfilling a request for sensor data directly (if the request is simple) or communicating with an associated OA-Service if the permission requires complex management (e.g., event recognition). This code is meant to be single-threaded and run quickly to interface well with the private data manager. The OA-Permission can hand off complex functionality to an OA-Service which runs asynchronously.

OA-Permissions, being in userspace, could be easily and modularly installed as

separate applications from the app store. In the future, when a new application is downloaded, a manager service could identify, download, and install all new necessary OA-Permissions. However, these permissions must be signed and trusted to be used in our system. As said before, each permission is arbitrarily given a risk level. Our implementation of OA-Permissions is very similar to the original PE for Android implementation for uPALs. uPALs are micro-Privacy Abstraction Layer modules that reduce a standard Android permission's information before applications receive said information.. However, we found uPALs, built to simple and quickly-terminating modules, on their own to be insufficient for our complex sensor usage. We modified the existing uPAL implementation by allowing OA-Permissions to facilitate the handling of result receivers (such as to OA-Services). These result receivers allow the OA-Permission (and the OA-Service) to send results back to the application.

#### 5.3.1.5 OA-Service

OA-Permissions that require more complex or continuous sensor data management require an OA-Service. These services are given the requesting application's result receiver, which they can use to send data back to the requesting application at any time. This enables the service to interface with the user through foreground activities, get a result, and send it back. Another example is how multiple applications can register with a single ongoing service process, which can perform an operation on sensor data and send the result to selected registered applications. This is a new addition to the PE for Android implementation. Like OA-Permissions, they can be easily installed separately. Ideally, these services should be signed and registered with the Private Data Manager to be valid; however, our implementation forgoes this for an easy proof-of-concept system and instead implements these services as basic Android foreground services that notify the user when run.

### 5.3.2 Demonstrative Applications and Permissions

Each of our applications is written in Java for OA-Sys. Each application uses a specific OA-Permission for sensor data and requests sensor data without using the more dangerous Android permission. Each OA-Permission has an associated OA-Service which is implemented via a foreground service and implements the corresponding permission model pseudocode found in Figure 5.6. Additionally, each application logs power data via the BatteryManager API at a rate close to 20 Hz for power evaluation. The QR and Hotword OA-Permissions provide near theoretical minimal information to applications to support functionality. The Motion OA-Permission provides more information to the Compass application than required for serving as a basis to evaluate the attack system and signal processing mitigations.

#### 5.3.2.1 QR Code Application and Permission

This application gets a QR code with no camera permission and instead only receives the QR encoding itself from the QR OA-Permission. This permission is an example of an user-initiated event (Section 5.2.3.1).

**Application:** This application's user interface is simple: one button to initiate capturing of a QR code, and one text box to display the result of the last QR code. The user pushes the button, the activity requests the QR code OA-Permission in our system, and then asynchronously receives a result. The application itself never receives any camera data, and only receives a QR code string (a website URL in our tests).

**OA-Permission:** This OA-Permission is very simple: when receiving an authorized request it sends the request to the QR code OA-Service along with the requesting application's result receiver.

**OA-Service:** When the OA-Permission calls this service, written as an activity as it has UI elements, it starts and promptly launches a foreground activity that captures

and displays the camera using the androidx camera API. When the service detects a QR code in the camera using the Google zxing API, it sends the result back to the application that called this OA-Permission via the result receiver. To provide this functionality, this service needs to request the Android camera permission and declare it uses camera hardware in its AndroidManifest file.

#### 5.3.2.2 Hotword Recognizer Application and Service

This application performs hotword recognition without having microphone data or permissions in any form. Instead, it only receives a notification from the Hotword OA-Permission and service. This provides an example of an event recognition permission model.

**Application:** This application has a user interface with two buttons and one text box for display. The phrase "oversensing" is hardcoded into the application to serve as the hotword. The hotword phrase is displayed on the text box to remind the user. The first button, the start button, registers the application with the hotword recognition OA-Permission when pressed. Until the stop button is pressed the application will receive a notification any time the hotword is spoken in the vicinity of the phone. The stop button unregistered the application with the OA-Permission.

**OA-Permission:** Upon an authorized request, this OA-Permission simply sends the Hotword OA-Service (1) the requesting application, (2) the result receiver for that application, and (3) the desired hotword phrase.

**OA-Service:** When called by the permission, the service receives (1) the requesting application, (2) the result receiver, and (3) the desired phrase from the OA-Service. It stores each triplet a list. When the list has any entries, the service listens using the microphone for any nearby speech. It uses Google Speech to Text API to transcribe the phrase, then compares the top 5 results against the list of all registered hotwords.

If a hotword is recognized, the service sends a simple message indicating the hotword was found to the corresponding application through its result receiver. In this fashion, one application can register multiple hotwords and be notified when a specific one is said and multiple applications can register different (or the same) hotwords. Notice, in a case where many hotwords or applications are registered the recognition code is still only run once, saving power versus each application performing this separately.

#### 5.3.2.3 Compass Application and Motion Service

This application receives motion sensor data without motion sensor permissions in any form. Information in this provided data could be reduced via signal processing; however, in this specific implementation, we provide unfiltered motion sensor data such that the data can be recorded for several software-based signal processing mitigations to be tested on the same data. Additionally, one could make a simple "compass heading" OA-Permission and service to further reduce data. We instead provide the motion sensor data itself. This permission and service is an example of a continuous sensor data stream permission.

**Application:** This application has a user interface with a start and stop button, a text box that displays the current bearing, and a compass image that rotates to display the current north heading. When the start button is pressed, the application registers with the motion sensor data permission that supplies accelerometer, gyroscope, and magnetometer data. When the application receives this data, it uses the SensorManager API to calculate its current orientation and then updates the current bearing text and the compass image's rotation angle.

**OA-Permission:** Similar to the Hotword and QR code permissions, this permission is kept simple and forwards the authorized requester and its receiver to the Motion Service to handle properly.

**OA-Service:** This service receives and stores the requester and receiver pair into a list. If the list of pairs has any members (if any application is registered with this permission), the service registers a listener with the Sensor Manager service to receive accelerometer, gyroscope, and magnetometer data. If the list loses all members, the service unregisters these listeners. When a sensor manager listener receives data, the service forwards this data and its timestamp to all registered apps. Note that this service purposely does not use signal processing to reduce the information in this data, but easily could before sending the data to the registered applications. It does not reduce the data so that the data can be logged unaltered, and so that our software signal processing mitigations can be tested on the same data.

## 5.4 OA-Sys and OA-Permissions Overhead

We find the apps that use our permissions incur little to no power overhead and slightly smaller APK, android application package, sizes. We installed our system and permission implementations (Section 5.3) on an unlocked Google Pixel 3 to measure power consumption and measure APK size. We created non-OA-Sys versions of each demonstrative application (which we refer to as *normal applications*, versus *OA-Apps*) to use as comparisons in this subsection.

### 5.4.1 Power Overhead

We collected power consumption using the built-in power logger described in Section 5.3.2 for the system and each application. A blank activity, an application with a blank screen and no calculations, served as a power consumption baseline. For each case study, we collected power readings from the OA-App and normal versions of the code in OA-Sys for 20-21 minutes. The collections were staggered (OA-App then normal, repeat) to reduce temporal bias from outside factors. This was done three times each, with each version having a total of 60 minutes of logged power readings.

| Configuration | | Application Power Draw (mA) | | | |
|---|---|---|---|---|---|
| **OS** | **App Type** | **Blank** | **QR Code** | **Hotword** | **Compass** |
| OA-Sys | OA-Apps | 144.87 | 594.69 | 193.93 | 160.06 |
| OA-Sys | Normal | 143.79 | 599.83 | 193.96 | 160.63 |

Table 5.2: Average power draw (mA) over 20-21 minutes of a combination of OA-Apps and normal Android apps on OA-Sys and Android 9.

| | Applications | | | OA-Sys | |
|---|---|---|---|---|---|
| **App Type** | **QR Code** | **Hotword** | **Compass** | **Permissions** | **Service** |
| OA-Apps | 3.212 | 3.300 | 3.470 | 3.331 | 3.800 |
| Normal | 3.895 | 3.303 | 3.471 | n/a | n/a |

Table 5.3: APK sizes (MB) of normal applications, OA-Apps, OA-Permissions, and OA-Services.

The 60 minutes total was averaged for our final numbers. During all collections, the screen brightness was adjusted to its lowest setting; wifi and Bluetooth were turned off to reduce outside effects on power consumption. For the hotword applications specifically, the smartphone was placed in a quiet room. The applications consume non-trivial power when noise is detected to classify the noise. Thus reducing ambient noise reduces the chances of one test attempting more classifications and biasing the power results. Last, each application was manually checked at the start and end of its power recording to ensure Android OS had not killed reliant processes, affecting power consumption.

Table 5.2 shows that the power difference between the OA-Apps and normal apps is fairly minimal and likely under the measurement noise threshold. This is to be expected, as much of the code between the two implementations are very similar, except in the normal version the code is in the application itself, whereas in the OA-Apps version the code is in the OA-Permissions.

### 5.4.2 Application and Permission File Sizes

We also provide the sizes of the applications. As one can see in Table 5.3, the sizes for OA-Apps are slightly smaller than the sizes of the normal apps. This is

to be expected as the OA-Apps rely on the OA-Services to perform much of the calculations. Thus they have less code in their applications while having similar functionality. However, OA-Sys also requires the overhead of the OA-Permissions and OA-Services to be installed. This does incur some overhead. Yet, if there are server applications that use these permissions it may save overall storage size.

## 5.5 Discussion

### 5.5.1 Security of QR and Hotword OA-Permissions

The QR and Hotword OA-Permissions mitigate respective oversensing vulnerabilities by design. The extent to which they improve security is difficult to evaluate quantitatively as they provide very different forms of data. However, both permissions reach near theoretical minimums for information given to applications while providing full original functionality. The QR OA-Permission reduces video data to a website link or character string, the encoded data, and nothing more. The hotword OA-Permission reduces ongoing microphone data to an event indicating the time that the hotword is said, essentially a timestamp for the event and nothing more. One potential security flaw for the hotword OA-Permission would be if an application registers for many hotwords simultaneously. However, the operating system could limit the number of registered hotwords or require the application to declare registered hotwords on install. While difficult to evaluate quantitatively, both reduce information to applications and reduce oversensing vulnerabilities.

### 5.5.2 Enabling Future Applications

Future applications, such as those in mobile augmented reality systems, may heavily rely on sensor data to function correctly. Current permissioning models may not sufficiently support secure models of sensor usage for these future applications, such

as provided by our hotword OA-Permission. Modern permissioning systems consider cameras, microphones, and other sensors that are always available to applications to be quite dangerous, and with good reason. However, future application functionality will hinge on sensor use. Eventually, permissioning systems must support functionality while preserving user privacy such as in OA-Sys.

### 5.5.3 Security Analog Filtering

Analog filters are a likely mitigation for oversensing vulnerabilities that rely on aliasing, such as for the attack system described in Section 5.3, as these filters operate on signals before aliasing occurs in analog-to-digital converters. For our case study, we chose to instead build signal processing mitigations that could be deployed via a software update such as in a OA-Permission as these can be most readily adopted by the largest number of people. However, in the future manufacturers will consider building analog filters into their sensor chips instead of digital filters (as seen with the digital anti-aliasing filter) to better mitigate such vulnerabilities.

### 5.5.4 Operating Systems Support and Compatibility

Our proposed changes to permission systems should be more expressive and enable more sensor functionality; however, they should also be completely compatible with modern systems and applications as they add functionality into systems without removing existing functionality. These more expressive and secure permissions could be deployed into current systems, with older permissions and capabilities left untouched. This would completely support all existing applications. However, with these more secure permissions available, the older permissions should be marked as significantly more dangerous than they currently are. This danger should then be conveyed to users explicitly through something like a pop-up each time the dangerous permission is used. This may encourage a gradual switch to the newer permissions. Significant

potential future research remains in this area.

## 5.6 Conclusion

Future applications, such as those in augmented reality or the Internet-of-Things, will need fine-grain sensor data to properly function. However, current permission systems must undesirably choose between providing privacy-sensitive information embedded in raw sensor data — introducing oversensing vulnerabilities — to applications or preventing those applications from functioning correctly by withholding raw sensor data. We design OA-Sys to bridge this gap and mitigate oversensing vulnerabilities, providing applications with necessary sensor-derived data while eliminating access to other information from that sensor. OA-Sys builds on previous work [54] to produce an open-source custom Android 9 implementation in which we build three permissions to provide QR codes, hotwords, and compass functionality to demonstrative applications despite these applications receiving significantly reduced sensor information. Two of the permissions reduce given information near to its theoretical minimum, while the last demonstrates how to rely on signal processing to reduce sensitive information for difficult-to-eliminate oversensing vulnerabilities (reducing our dial-tone keylogger accuracy by >50%). We provide OA-Sys to the community to provide anti-oversensing concepts, designs, and implementations that may be built upon or built into other permission systems as an initial step to mitigating oversensing as smart devices, mobile devices, and sensing reliant applications become ubiquitous.

# CHAPTER VI

# Touchtone Eavesdropping: An Oversensing Example on Smartphone Motion Sensors

This chapter, based on my previous work[1], presents an example of an oversensing vulnerability in touchtone eavesdropping on smartphone motion sensors and signal processing mitigations to mitigate this vulnerability. This chapter's oversensing vulnerability is an out-of-scope vulnerability. The signal processing-based mitigations could be included in permissions for continuous sensor data streams such as for the Compass application discussed in Chapter V.

Touchtones, the sounds produced by a smartphone when a numerical key is pressed, are an established communication standard [149] often used to encode user feedback in telephony channels. In modern telephony systems, touchtones often represent important information such as credit card numbers (during activation), bank pins, various account numbers, social security numbers, selections for various options in automated services and even votes in a federal election (done by phone) [150]. Recent research has shown that sound produced by a smartphone's speaker may leak into the same phone's motion sensors, particularly speech. This chapter's experiments show that *touchtone leakage*, touchtone information leaking into motion sensor data,

[1]Connor Bolton, Yan Long, Jun Han, Josiah Hester, Kevin Fu. "Touchtone Leakage Attacks via Smartphone Sensors: Mitigation Without Hardware Modification." 2021 arXiv. `https://arxiv.org/abs/2109.13834`

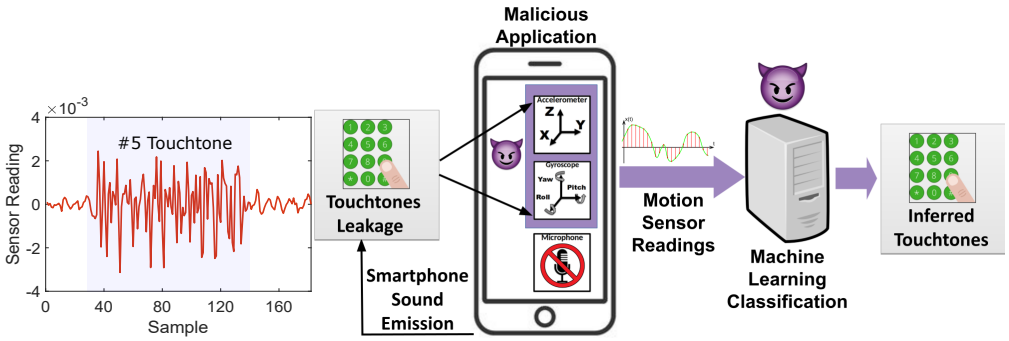



Figure 6.1: (a) A touchtone, indicating a "5" on a smartphone number pad, leaks into accelerometer data. (b) A malicious smartphone application can classify this leakage to discern that a "5" touchtone was emitted, infering user input of a "5" for purposes such as dialing a phone number or inputing information into automated services.

enables malicious smartphone applications with motion sensor access (e.g., a seemingly benign smartphone game) to ascertain any numerical user input that produces touchtones (Fig 6.1), an attack we term *touchtone eavesdropping*. This chapter's primary goal is to open discussion on how to reduce this oversensing vulnerability.

To understand how to mitigate touchtone eavesdropping, it is necessary to understand how acoustic information is hidden in motion sensor data and how signal processing and physical phenomenon, such as aliasing or varying frequency responses, aid adversarial recovery of the original user keypress. These phenomena cause artifacts of touchtone information to manifest in several ways, but an adversary only needs the capability to ascertain user input through one of those manifestations. More advanced techniques such as selective integration of multiple sensors and sensor axes via machine learning can utilize several of these manifestations simultaneously for a more proficient attack. Our experiments show user input can be recovered by an adversary at over 99 % accuracy with such methods.

Mitigations can reduce touchtone leakage by reducing the total information in sensor output, but this also affects benign applications relying on such data. It is thus important to keep functionality in mind when designing mitigations. Additionally, we focus on solutions that may be implemented as a software update to support existing

devices and designs where hardware changes may not be viable.

Using these criteria we analyze both effective and ineffective solutions to demonstrate ideas to emulate or avoid. For example, we analyze and evaluate how some apparent mitigations briefly suggested in related work, such as sampling rate reduction or digital low-pass filtering alone, are ineffective at reducing touchtone leakage; sampling rate reduction can reduce available information to *all* applications by more than 80% yet our classifier maintains accuracy over 95% for three of the four tested phones. Other designs, such as a software anti-aliasing filter that uses oversampling, do not change the amount of information available to applications while reducing accuracy by over 50.1%. Our contributions include:

**Touchtone eavesdropping assessment:** We discuss and experimentally demonstrate the relevant physics and signal processing theory of touchtone leakage to reveal challenges mitigations must consider, such as aliasing and non-linear frequency responses. We detail both simple and advanced adversaries, discussing how selective integration of multiple sensors' data with machine learning can further impede mitigation efforts.

**Defense design analysis:** We analyze the advantages and disadvantages of several signal processing-based leakage reduction solutions. We explicitly include the idea of not reducing functionality as design criteria. Some apparent approaches, such as reduced sampling rates, are unintuitively ineffective mitigations. Other filter designs, such as software Butterworth anti-aliasing filters, can reduce leakage in the signals as a software update. Hardware changes could serve as long-term solutions.

**Implementation and evaluation:** We implement and evaluate both simple and more advanced touchtone leakage attacks. These attacks serve as baseline metrics for mitigation evaluation. Baseline attacks can achieve accuracy higher than 99%. We evaluate several signal processing mitigation designs to demonstrate both effective and ineffective designs. Apparent mitigations, such as digital low-pass filters and reduced

sampling rates, may remain ineffective (less than a 1% difference in accuracy from baseline) even while reducing benign information crucial to application functionality. Our anti-aliasing filter reduces accuracy by over 50.1% with no information loss and can reduce further with minor benign information loss.

## 6.1 Overview

### 6.1.1 Touchtones

Touchtones, also known as dual-tone multi-frequency (DTMF) signals, are a standardized [149] code of two-tone audible acoustic signals that play upon a numerical keypress, often used in telecommunications or other applications with a numerical touchpad [27, 72, 125, 51]. The sound produced by a phone when you press an individual key to dial a phone number, answer an automated telephony question (e.g., "press 1 to...."), register credit card numbers or bank pins over the phone, or other such actions are examples of touchtones. There are 16 unique touchtones (Fig 6.2), each consisting of two frequencies taken from two separate frequency sets, used for the numbers 0-9, the symbols * and #, and an additional four tones reserved for special services. As they are unique, hearing one touchtone is indicative of a particular number input. These dual-tone combinations are designed for reliability in noisy channels for reliable communication.

### 6.1.2 Threat Model

This paper considers an adversary whose goal is to determine a user's numerical keypresses on a smartphone using access to a smartphone's motion sensor data and the knowledge of touchtone leakage, an attack we term *touchtone eavesdropping*. We assume the adversary can obtain and save motion sensor data through means such as a malicious application with motion sensor access. The adversary may have access


Figure 6.2: Touchtones are comprised of two single-frequency tones emitted simultaneously to convey numerical input.

to the same model as the victim's phone(s); a phone's model can be determined by an application using fingerprinting techniques [18, 107, 167]. The adversary can use their duplicate phone(s) to collect training data to build a classification system. Last, the adversary has unlimited time to classify victim data; data is easily saved for offline processing and sensitive information (*e.g.,* credit card numbers, bank pins, social security numbers) is unlikely to often change.

## 6.2 Touchtone eavesdropping assessment

Developing an understanding of how attacks utilizing touchtone leakage occur (Fig 6.1) and why they can be difficult to mitigate (Fig 6.3) is crucial for defense design. At a high level, touchtone leakage is classified as an out-of-band oversensing vulnerability using the relationship between acoustic waves and motion sensor output. This attack is possible in continuous data stream sensor use cases as defined in Chapter V. We assess (1) how touchtones produced by a phone's speaker leak distinguishable, deterministic byproducts to the smartphone's motion sensors (e.g., the accelerometer and gyroscope), and (2) how adversaries can use leakage to determine user input.

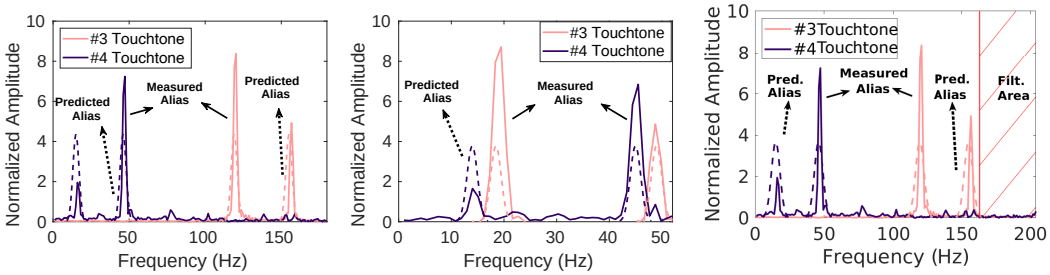



Figure 6.3: Touchtone leakage for #3 and #4 touchtones in a Google Pixel 2's accelerometer's x-axis. These signals remain discernable and predictable in the frequency domain with (a) a normal, unaltered signal, and also despite previously suggested mitigations in (b) reduced sampling rates and (c) digital low-pass filtering.

### 6.2.1 Touchtone information in motion sensor data

Artifacts of touchtone leakage manifest in motion sensor data in a multitude of forms due to various physical and signal processing phenomenon. Each of these manifestations can contain redundant or complementary information regarding the original touchtone. However, an attacker only needs one recovery method for one touchtone artifact to achieve an eavesdropping attack. Defenders must then consider how to block as many touchtone artifacts and recovery methods as possible.

#### 6.2.1.1 Touchtone aliasing

Aliasing (Section 2.2) is a key factor in both making touchtone leakage occur and for making it difficult to mitigate. Touchtones have frequencies higher than the Nyquist sampling rate for most smartphone motion sensors, and thus have aliases. However, the frequencies of these aliases can be predicted as the touchtone frequency and sampling rate are both known (Fig 6.3). An attacker can use these known aliases to indicate the presence of the missing original touchtone frequencies.

Furthermore, the non-linear placement of these aliases — how all touchtone frequencies can lie somewhere in the sampled signal's frequency band — can make touchtone eavesdropping resistant to suggested mitigations. For example, reducing the sampling rate will not eliminate aliases; changing the sampling rate only alters

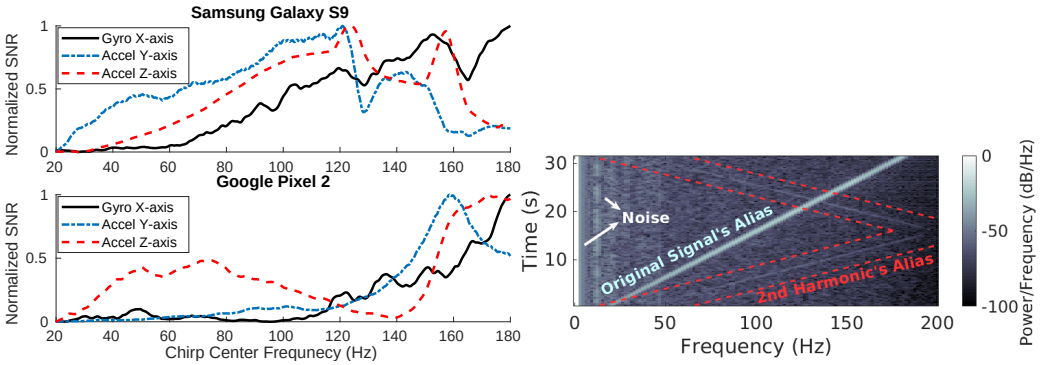



Figure 6.4: Touchtone information can be embedded in a variety of forms or to varying extents in motion sensor data. In (a), two axes have distinct non-linear frequency responses to a 420 Hz to 580 Hz chirp from the loudspeaker. Different axes may be better predictors for certain tones. (b) shows how there may be many subtle artifacts in touchtone data. An attacker could use any of these artifacts to launch a touchtone eavesdropping attack.

aliased frequencies (Fig 6.3b). Low-pass filters remain ineffective unless the cutoff frequency is placed low, as touchtone aliases could be close to 0 Hz (Fig 6.3c).

#### 6.2.1.2 A cacophony of sensitive information

The above factors enable touchtone leakage when combined with how acoustic waves interact with motion sensors (Section 2.3). However, various physical and signal processing phenomenon can cause the same target information (i.e., a touchtone) to uniquely manifest in different sensors' data (Figure 6.4a) or manifest in a multitude of forms simultaneously in the same sensors' data (Figure 6.4b). An attacker may only need one of these manifestations for a successful attack. An attacker could also make use of all information simultaneously for a more effective attack.

For example, different sensors or sensor axes can contain complementary or different frequency information about the same set of touchtones (Figure 6.4a). This difference stems from varying frequency responses in inherent to phone construction, speakers, sensors, or even individual axes of sensors. With access to multiple axes or sensors, an adversary may be able to exploit how each axis or sensor could be a better indicator of certain frequencies due different signal-to-noise ratios.

Additionally, in the same signal (i.e., sensor axis) information about the same touchtone can manifest in different manners (Figure 6.4b). For example, an axis will have information on an alias of the touchtone frequency, but could also have information on the harmonics of the same touchtone. A touchtone eavesdropping attack would only need to recognize one of a touchtone's alias, harmonic, or even an alias of the harmonic to be successful.

### 6.2.2 Adversarial touchtone recovery

The goal of the attacker is to recognize when a touchtone is pressed by discerning the presence of the eight individual touchtone frequencies (Fig 6.2) in motion sensor data. The adversary can benefit by trying to use all possible touchtone information in motion sensor data, as discussed in Section 6.2.1.2. The most straightforward approach to do this is by making a machine learning-based classifier as the attacker does not particularly care which information the classifier uses, just that it can classify motion sensor data into touchtones. The advent of easily usable machine learning tools makes this task easy in the modern-day.

Furthermore, adversaries can make use of the varying information in different sensors and sensor axes by selectively integrating data from multiple sensors (in our case the accelerometer and gyroscope). For example, one sensor axis may be more apt at discerning the presence of a particular touchtone but a separate sensor axis could be a better indicator of a separate touchtone. This same idea, using multiple sensors to reveal emergent information, has been used by researchers for benign purposes in several fields including on drones [73], body-sensor networks [50], and much more. Building a classification model to specifically use this fact should lead to more efficient attacks.

## 6.3 Functionality-aware software mitigation design

Touchtone eavesdropping mitigations require careful forethought and consideration of leakage mechanics (Section 6.2.1) to effectively reduce leakage while not hampering benign application behavior, a problem shared by all signal processing oversensing mitigations. To accomplish this task, mitigations should reduce touchtone information in motion sensor data while minimally altering or reducing any other information. This chapter adds an additional criterion that mitigations should be able to deploy as a software update to support current devices. This section examines several mitigation designs to show how some apparent designs such as sampling rate reduction can only reduce touchtone leakage by reducing the total information in a signal, hampering application functionality, while other designs such as anti-aliasing filters can reduce touchtone leakage while minimally harming application functionality.

### 6.3.1 Designing for both privacy and functionality

While protecting the privacy of smartphone users from touchtone eavesdropping attacks is an urgent issue, I consider ensuring functionality to be a second — but no less critical — criterion for mitigation design. The reason is that to be adopted into mainstream systems the mitigation must also support the expected functionality of motion sensor dependent applications. It is widely accepted that security and privacy must support some level of functionality and usability [160, 153, 81] as these features drive device markets and development.

Touchtone eavesdropping attackers and benign smartphone applications using the motion sensors use the same signals — the motion sensor readings. As a result, limiting the attackers' capability might also inadvertently limit benign applications' performance. Thus for practical deployment, designing such mitigation for privacy protection requires the designers to be functionality-aware and guarantee minimal

degradation of functionality by carefully optimizing the implementation of their mitigation.

As discussed in Section 2.2 two significant factors for reducing information in a signal — and thereby reducing application functionality — are (1) bandwidth reduction and (2) signal distortion; conversely, minimizing bandwidth reduction and signal distortion can better support application functionality. There is a nearly unanimous trend of higher sensor bandwidth leading to higher performance in previous research in various activities such as human activity recognition [68], animal health monitoring, [105], road quality assessment [20], etc. In addition, more commercialized techniques such as using motion sensor readings for smartphone image stabilization [62] and rolling shutter correction [65] rely on sample rates higher than 100 Hz, correlating to a bandwidth of 50 HZ. Thus reducing bandwidth below those ranges risks causing these applications to malfunction and may stifle future application performance. A significantly distorted signal could also impact application behavior and thus should also be minimized when possible, but it is more difficult to ascertain how much distortion is permissible. Ideal mitigations should support the original bandwidth with minimal distortion, only removing traces of touchtone byproducts.

### 6.3.2 Apparent mitigations that sacrifice functionality

Mitigation strategies predicated on reducing available sensor bandwidth may not only hinder application functionality but also may ineffectively attenuate sensitive touchtone information. This section analyzes apparent mitigations of sampling rate reduction and digital low-pass filtering to show how touchtone information may persist despite significant bandwidth reduction.

#### 6.3.2.1 Sampling rate reduction

Lowering sampling rates directly lowers available bandwidth (Section 2.2) in an attempt to also lessen the threat of acoustic eavesdropping; however, it is ineffective at attenuating leakage (Fig 6.3b) primarily due to how aliasing will place touchtone information in a digital signal no matter the sampling rate (Section 6.2.1.1). Even at very low frequencies, the eight touchtone frequencies still provide discernible and potentially recoverable aliases. However, such low sampling rates may prevent an attacker from differentiating between different touchtone aliases at high fidelity. But the low rates similarly affect dependent applications' functionality similarly. Our experiments back this intuition (Section 6.5.2.1), as reduced sampling rates achieve minimal accuracy reduction for our touchtone eavesdropping attack until having sampling rates under 100 Hz, a fourth of the original sampling rate.

#### 6.3.2.2 Digital Low-pass filter

Low pass filters may at first seem like a natural mitigation for touchtone leakage, which relies on aliasing, but a software digital low-pass filter alone cannot increase privacy while preserving functionality (Fig 6.3c). To note, previous papers seldom specify which low-pass filter design they suggest, and hardware changes to include analog low pass filters may be a sufficient future defense as later discussed. However, when discussing software-updatable mitigations, digital low-pass filters alone also do not address the problem of aliasing. Referring back to Section 6.2.1.1, many of the resulting touchtone aliases could be under the low-pass filter cutoff frequency that is chosen due to the non-linear placement of alias frequencies. A lower cutoff frequency is more likely to attenuate more aliases, but only because it is reducing the available bandwidth for all motion sensor data. Thus it also suffers from needing to reduce available bandwidth to provide better privacy. Our experiments demonstrate this pathology (Section 6.5.2.2).

### 6.3.3 Designing functionality-aware signal processing mitigations

A functionality-aware signal processing mitigation should minimally reduce available bandwidth and distortion while attenuating touchtone leakage. Our approach is to rely on established digital signal processing techniques designed to eliminate specific leakage contributors, particularly aliasing. We propose a software update enabling oversampling and digital anti-aliasing filters as a primary means of defense against acoustic general acoustic leakage. Additionally, we describe how one can utilize the predictable nature of touchtone aliases in defense design.

#### 6.3.3.1 Oversampling and digital anti-aliasing filters

Oversampling is the act of sampling at a faster rate than the bandwidth that you wish to eventually provide, and can be used to create anti-aliasing filters that reduce touchtone leakage while still providing the original bandwidth to current applications (Fig 6.5). Oversampling can be implemented as a software update on most phones as often the sampling rate is limited not by the sensing hardware, but by the operating system and sensor drivers to preserve power. Thus, a software update could change these driver values to provide a faster sampling rate to the operating system. The operating system can then perform some operation on the oversampled signal and subsequently downsample the signal to the original sampling frequency. If the oversampled frequency is a multiple of the original by a factor of $n$, downsampling is trivially performed by selecting every $n$th sample. The oversampling frequency being a non-multiple can introduce distortion into the digitized signal. This method provides the same signal sampling rate and bandwidth as current designs.

Digital anti-aliasing filters can employ oversampling to attenuate touchtone aliases while minimally altering other information applications may desire. The key is that the filters can remove any information above the original sampling rate's Nyquist frequency without affecting legitimate (i.e., not touchtone alias) data as seen in Fig 6.5.

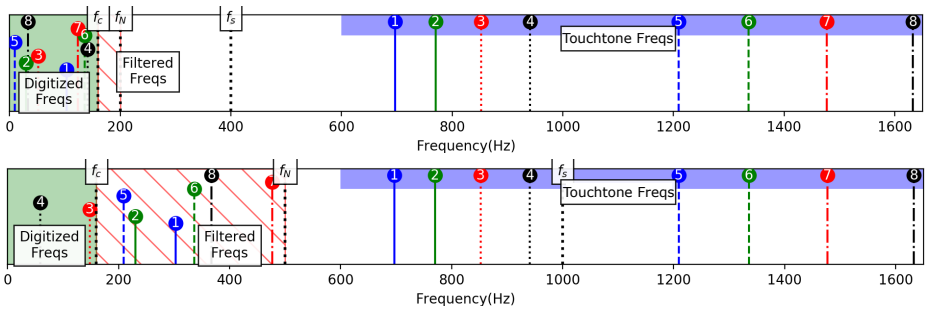



Figure 6.5: Digital anti-aliasing filters can attenuate more touchtone aliases than a low pass filter without reducing available bandwidth due to the use of oversampling. In the example with a sampling rate of $f_S = 400Hz$, $f_N = 200\ Hz$, and $f_c = 180\ Hz$, touchtone aliases (numbered 1 to 8 to correlate with the eight touchtone frequencies in the blue box) are attenuated if filtered (diagonal red-lined area) and otherwise unattenuated (green area). (a) A digital low-pass filter may be unable to attenuate many touchtone frequency aliases without also eliminating significant frequency information benign applications may rely on (Section 6.3.2.2). (b) A digital anti-aliasing filter with the same $f_c$ can filter more frequencies due to the use of oversampling (Section 6.3.3.1).

Due to the non-linear nature of aliased frequencies, with oversampling the touchtone aliases may fall into range and can be attenuated without affecting benign information. This is not a panacea, however, as aliases of sensitive information may still be in the original sampling range, but such a design can attenuate touchtone aliases without attenuating information that applications may expect.

#### 6.3.3.2 Mitigations for targeted sensitive frequencies

When there is a case of known sensitive signals with specific frequencies, such as in the case of mitigating touchtones, one can use frequency-specific mitigation designs such as notch filters and selective sampling frequencies in combination with anti-aliasing or other filtering techniques. A notch filter is a digital or analog filter design, similar to the high and low pass filters in Section 2.2, that attenuates information with frequencies between two cutoff frequencies. One could design multiple notch filters to attenuate targeted sensitive frequencies such as the eight touchtone frequencies.

Another approach is to use an anti-aliasing design while carefully selecting the

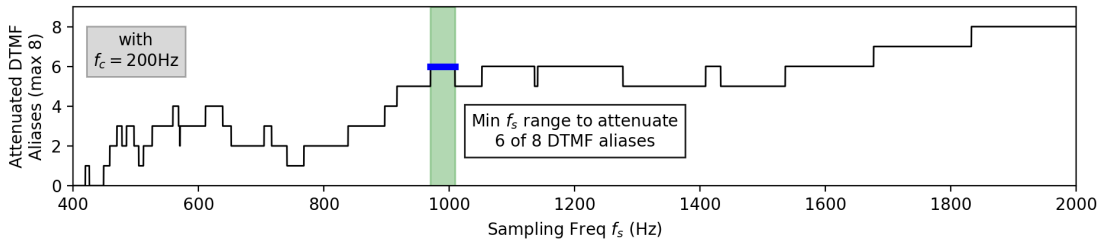



Figure 6.6: A mitigation designer desiring to attenuate the most touchtone aliases using the lowest sampling rate $f_S$ when given a bandwidth $f_c$ to support can make use of the non-linear but predictable nature of aliased frequencies. As the oversampled rate increases, the number of aliased frequencies above $f_c$ will change. The designer can calculate this number of attenuated aliases then select an appropriate sampling rate to meet design constraints.

sampling frequency to maximize the number of targeted sensitive frequencies above $f_N$ (Fig 6.6). The basis for this lies in the non-linear relationship between signal frequency, sampling frequency, and the alias frequencies as seen in Section 2.2. One can set the filter cutoff frequency $f_c$ to design for the desired bandwidth. Then, the designer could change the sampling rate until a desired number of aliases fall above $f_c$ so they can be eliminated. This could allow a mitigation designer to select lower sampling frequencies that result in greater protection from touchtone leakage.

## 6.4 Experimental Method

We recorded and used a custom machine learning classifier to recognize touchtone samples on multiple phones with and without mitigations in place. Our machine learning classifier mimics advanced adversaries using a variety of time and frequency features along with selective axis integration as explained in Section 6.2.2. We implemented and tested four software-based signal processing mitigations.

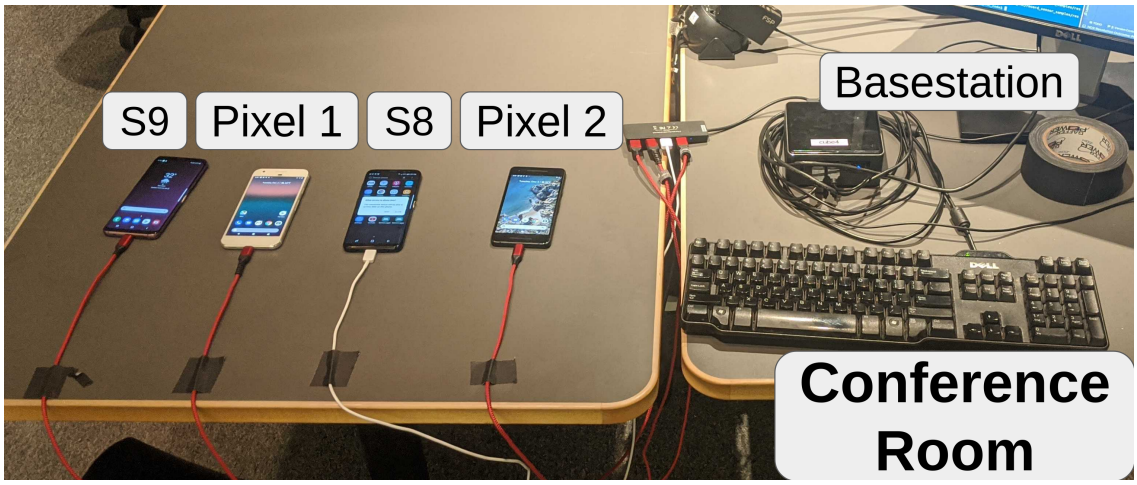



Figure 6.7: Data collection setup in conference room

### 6.4.1 Data Collection

#### 6.4.1.1 Hardware

We have three different hardware setups for motion sensor data collection. The first two setups collect data from the four Android phones listed in 6.1 for baseline and software-only mitigation evaluation; these two setups differ only in physical locations: a quieter conference room versus a noisy server room. The conference room was next to a busy atrium with the door closed to mimic a conference call setting, while the server room was chosen to mimic a noisy environment measured at an average of 67 dB SPL as measured by a General DSM403SD sound level meter [142]. Each setup used an Intel NUC running Ubuntu 18.04 [63] as a base station, smart-phones (Table 6.1), cables, and base station peripherals on a table (Fig 6.7). In this setup, the acoustic speaker was a phone's loudspeaker and the motion sensors (accelerometer and gyroscope) were the same phone's sensors. The base station used a python API for the Android Debug Bridge [33] to upload a custom Android data collection program to each phone and for other communication or file transfer.

The third hardware setup collects data at faster sampling rates for anti-aliasing software filters and for testing on-board sensor anti-aliasing filtering. Phone hardware

| Manufacturer and Model | Release Date | Reported IMU Model | Sampling Rate (Hz) | |
|---|---|---|---|---|
| | | | Reported | Measured |
| Google Pixel 1 | Oct 2016 | BMI160 | 400.00 | 401.69 |
| Google Pixel 2 | Oct 2017 | LSM6DSM | 400.00 | 409.96 |
| Samsung Galaxy S8 | Mar 2017 | LSM6DSL | 400.00 | 429.27 |
| Samsung Galaxy S9 | Mar 2018 | LSM6DSL | 415.97 | 413.61 |

Table 6.1: Reported inertial measurement unit (IMU) model, which contains both an accelerometer and gyroscope, and sampling rates are found via the Android Debug Bridge tool. Note that the sampling rates are limited by the operating system, and not sensing hardware.

can collect at rates faster than what is made available to applications in smartphones to limit power consumption. Although current phones do not support it, we test it with external sensors to emulate possible future mitigation. To that end, our setup uses LSM9DS1 breakout boards, a very similar chip to the ones in three of the phones (Table 6.1), a Teensy 3.6 microcontroller, the same Intel NUC base station as in the previous setup, and an external speaker connected to the NUC to produce audio. The speaker was placed 10cm away from the LSM9DS1 breakout board. A python program was used to produce audio on the speaker and interface with a custom sensor collection program on the Teensy micro-controller.

#### 6.4.1.2 Recording

To reduce temporally correlated biases from data collected over a long period of time, the python3 program running on the base station first determines a randomized order for all audio samples to record. The program then ensures the proper setup of all devices for the experiment. It then has the speaker for the experiment play each touchtone audio clip in succession while recording motion sensor data. In the event with multiple devices connected to the base station, only one speaker and sensor were used simultaneously. Motion sensor data was collected at the fastest available sampling rate and saved and sent back to the base station to save the recording to disk.

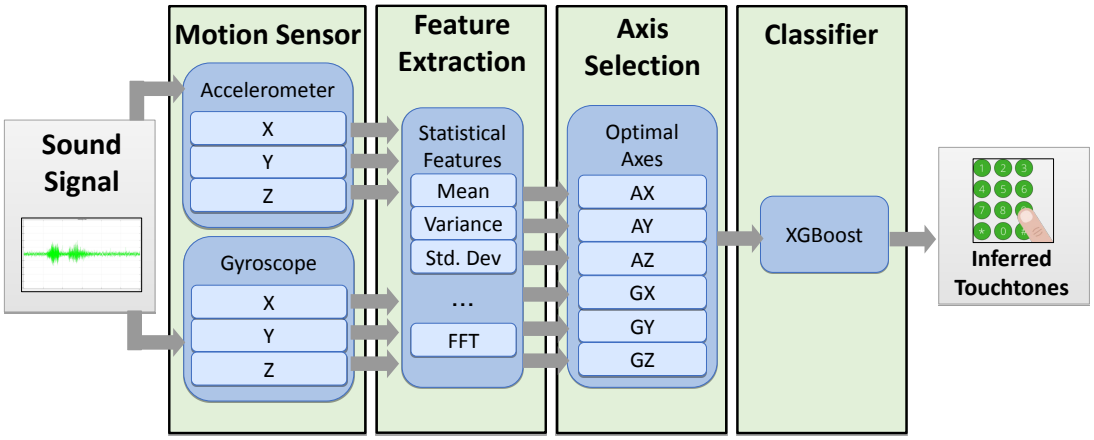



Figure 6.8: Our system extract signals features and selectively combine useful motion sensor data from multiple sensors and axes to better classify touchtones.

For each individual setup, we recorded the motion sensor data as each individual dial-tones was played for 0.5 s, with each tone being recorded 250 times per setting for a total of 4,000 recordings. The data set was divided into training and test sets at 80% and 20% respectively. It was ensured that touchtones were divided equally during the split (e.g., in the test set there were 50 samples of each of 16 touchtones).

### 6.4.2 Touchtone classifier

#### 6.4.2.1 Selective integration of sensor data

To emulate a more advanced adversary, we build classifiers that selectively integrate feature data from multiple sensors into a single attack model based on the intuition that each sensor axis can be a better or worse predictor for a given touchtone (Section 6.2.2). Previous work has demonstrated classifiers for acoustic leakage onto motion sensors [83, 166, 12]; however, to our knowledge no previous work has combined data from both sensors simultaneously or selectively integrated axes into a single model. This improvement works as each axis from each sensor carries some measure of unique information. Selectively combining these sources of unique information should yield the best results.

Our method to selectively integrate axes is as follows. First, the system imperially ranks the axes in order of best predictor by building a model for each individual axis and tests its accuracy on validation data. Then the system builds a model with the most accurate two axes, then top three, etc., until a model with all axes has been tested. Then the system selects the best performing model among the single-axis and multi-axis models to use in actual testing. Once the best combination of axes has been chosen, the axes will be selected in the Axis Selection step shown in Fig 6.8.

#### 6.4.2.2 Features and Classifier Design

We briefly detail the feature extraction and classifier of our touchtone classifier in this section.

**Time-alignment and Windowing:** For feature extraction of a sample, our model first time-aligns signals from different sensors (i.e., sample 1 from one signal correlates with sample 1 of the others). Subsequently, it divides each time-series signal into a series of windows. Each window should correlate with windows of other signals.

**Extract Statistical Features:** The system calculates a series of statistics per window per selected sensor axis and concatenates these metrics to produce a single feature vector. The set of statistical measurements (Table 6.2) is similar to previous work [12].

| | | | | |
|---|---|---|---|---|
| Mean | Median | Kurtosis | Absolute Area | % Mean Crossings |
| Minimum | Variance | Skew | Standard Deviation | Interquartile Range |
| Range | Maximum | Variation | Spectral Entropy | Signal Power |
| Fast Fourier Transform | | | First, Second, Third Quantiles | |

Table 6.2: A list of features used in classification. The signal would be split into windows where the above features were calculated.

**Zero-padding:** The classifier requires feature vectors of equal duration and subsequently an equal number of time windows. However, recordings will often have slightly different duration due to factors such as experimental error. This imprecision can result in varying numbers of time windows. The system corrects this by zero

padding each feature vector to the same length. To perform zero-padding, the system adds zeros or windows of zeros to shorter samples until all samples have the same length.

**XGBoost Classifier:** Our system uses *xgboost* to classify the extracted features from the selected axes. Xgboost is a common classifier that uses gradient boosting and has been shown to effective in several different applications [25].

#### 6.4.2.3 Implementation details and hyper-parameter tuning

The system uses a python3 program to process the sensor recordings and subsequently train and/or test recognition models. We utilize numpy, scipy, and other standard python3 libraries to perform feature extraction as described previously. The system then uses python3 XGBoost implementation with support libraries from Scikit-learn to perform any training, validation, or testing of machine learning models. To select the optimal combination of axes as described previously, the system would first train separate models for each individual axis. These axes would be then be ranked by individual accuracy performance. Axes would be added in order of highest accuracy and evaluated. Last, for these eleven combinations (6 individual and 5 multi-axis), the system would choose the best performing axis combination and use that for its model.

To choose specific features and model hyper-parameters, we performed a randomized grid search using data collected from a Pixel 2 phone in a conference room to imperially pick parameters. The randomized grid search did not test every possible combination of parameters in the interest of time, and thus it is possible more optimal parameters could be chosen. The possible parameters for features and classifiers are shown in Tables 6.3 and 6.4 respectively with selected parameters shown in bold. We tested these settings against a commonly used feature set for audio classification with Mel Frequency Cepstral Coefficients (MFCCs) [83] and another common

| Feature | Setting | Possible Choices |
|---|---|---|
| **Statistic Features** | Frame Size (#vals) | 10, 20, **50**, 100 |
| | Frame Step (#vals) | **5**, 10, 20 |
| MFCC | Window Length (s) | 0.025, 0.05, 0.1, **0.2**, 0.3, 0.5 |
| | Window Step (s) | **0.01**, 0.05, 0.01 |

Table 6.3: Feature settings. Settings used in the final model are in bold.

| Classifier | Setting | Possible Choices |
|---|---|---|
| **xgboost** | learning rate | 0.05, 0.10, 0.15, **0.20**, 0.25, 0.30 |
| | max depth | 3, 4, **5**, 6, 8, 10, 12, 15 |
| | min child weight | 1, **3**, 5, 7 |
| | gamma | 0.0, **0.1**, 0.2 , 0.3, 0.4 |
| | colsample bytree | 0.3, 0.4, **0.5** , 0.7 |
| Random Forest | bootstrap | True, **False** |
| | max depth | 10, 20, 30, 40, 50, 60, 70, **80**, 90, 100, None |
| | min samples leaf | 1, **2**, 4 |
| | min samples split | **2**, 5, 10 |
| | $n$-estimators | 200, 400, 600, 800, 1000, 1200, 1400 **1600**, 1800, 2000 |

Table 6.4: Classifier Settings. Settings used in the final model are in bold.

classifier with Random Forest [12] to provide a comparison against other commonly used selections. We took the highest accuracy result to select feature and classifier settings. These settings stayed the same throughout all testing.

### 6.4.3 Signal Processing Mitigations

#### 6.4.3.1 Selection

We selected a total of four signal processing mitigation designs to evaluate. Two designs were chosen due to their mention in previous research as possible mitigations: a software-only low pass filter and reduced sampling rates. However, our analysis (Section 6.3.2) projects that both mitigations should have minimal effect on touchtone leakage without significantly reducing the available information to all applications.

The low-pass filter used a Butterworth filter design with an order of 5.

The third and fourth mitigations, software and hardware digital anti-aliasing filters, were chosen to better support application functionality by not reducing bandwidth while still attenuating touchtone aliases. The tested software anti-aliasing filter design is essentially oversampling combined with filtering as shown in Fig 6.5. Specifically, it uses the oversampled data with a Butterworth low-pass filter and we test various filter orders. The Butterworth filter provides a good balance with only slight signal distortion and a sharper cutoff. The slight signal distortion means it should minimally affect applications relying on sensor data while the sharper cutoff means it should theoretically attenuate aliased signals further. Furthermore, this filter can be implemented as a software update to any phone but will require some computational burden and cause some signal delay, which may be unacceptable for some applications.

The hardware digital anti-aliasing filter refers to the on-board anti-aliasing filter on the LSD9DS1 breakout board, which should also be included on the LSM6DS(L/M) sensors on three of the tested phones. This filter should work similarly in theory to software anti-aliasing filters as they are both digital anti-aliasing filters, but the exact filter details are unfortunately black-box. The hardware filter benefits by requiring no computational burden and requiring less signal delay, but has the drawback that is less configurable and may not be available on some devices. To note, it *can* be implemented as a software update should the hardware be available by changing values in the sensor driver.

#### 6.4.3.2 Implementation Details

Implementation details for our four tested mitigations include:

1. **Reduced sampling rate.** The reduced sampling rate mitigation uses the original motion sensor data from the conference room hardware setup, but takes

1 sample of every $n$ samples to emulate downsampling. We vary $n$ to test sampling rates from 400 Hz to as low as 50 Hz, with Nyquist frequency and bandwidth equal to half the sampling rate.

2. **Software low-pass filtering.** The low-pass filter uses the original motion sensor data from the conference room hardware setup with an unaltered sample rate but applied the Python scipy Butterworth filter with an order of 5 for low-pass filtering. The signal cut-off frequency was varied from 200, 150, 100 to 50 Hz. This cut-off frequency effectively becomes the bandwidth of unattenuated information in the signal.

3. **Software (digital) anti-aliasing filtering.** The software anti-aliasing filter uses the oversampled data from the sensor breakout board setup and then applies a scipy Butterworth filter (the same as from the original low-pass filter) with a cutoff frequency equal to the eventual desired bandwidth. We vary this desired bandwidth to use as a comparison against other mitigations. The filtered signal is then downsampled (similar to the reduced sampling rate mitigation) to the desired bandwidth. We also vary filter order in this evaluation.

4. **Hardware (analog) anti-aliasing filtering.** The hardware anti-aliasing filter collects data from the sensor breakout board, changing the sensor's onboard filtering settings. There are four bandwidth configurations. The data from the four configurations are then processed by the classifier.

## 6.5 Evaluation Results and Analysis

In this section, we report the attack and mitigation results with the setups described in Section 6.4. We analyze and summarize the findings of our assessment of the eavesdropping attack and different mitigations. Software low-pass filtering and

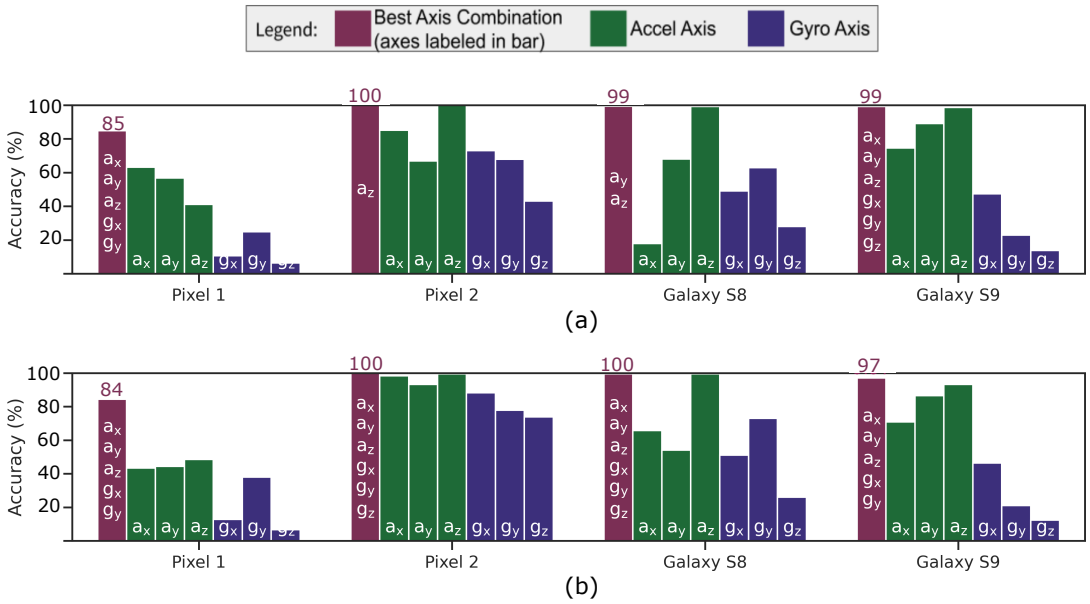



Figure 6.9: (a) Conference room and (b) Server room hardware setups. For each phone, we show the accuracy a classification model trained on individual axes alone, then show the accuracy for the model trained on the optimal combination of axes.

reducing the sensor sampling rate can only moderately mitigate the attack while significantly hindering data bandwidth (and thereby application functionality). Software and hardware digital anti-aliasing filters cannot eliminate touchtone eavesdropping, but can more significantly mitigate the threat while also preserving more data bandwidth.

### 6.5.1 Baseline evaluation metrics: attack effectiveness

We find that the unmitigated touchtone classifier achieves accuracy exceeding 99% for three of the four phones as shown in Fig 6.9, demonstrating that malicious applications can effectively recover user input.

#### 6.5.1.1 Differences between phone models

One of the phones, the Pixel 1, performs poorest in nearly every test despite similar sampling rates as the other phones. The highest touchtone inference accuracy for Pixel 1 does not exceed 85% while other phones can all achieve over 99%. The

most obvious explanation would be that it has a different inertial measurement unit produced by a different manufacturer than all other phones (Table 6.1).

This result demonstrates that factors other than sampling rates can vary recognition rates. These factors could include a signal propagation path that attenuates the acoustic signal, less sensitive sensors, different frequency responses, or different sensor configurations. This result also suggests that some motion sensors may be more resistant to touchtone leakage than others. An examination on which motion sensors are less susceptible could provide insight into future hardware-based mitigations.

#### 6.5.1.2 Accelerometer vs. gyroscope axis accuracies

Classification based on data from an accelerometer axis achieved higher average accuracy gyroscope axis data. While the exact reasons remain unclear, we provide a possible assumption. Accelerometers measure linear acceleration while gyroscopes measure angular acceleration. The phone's speakers produce audio through vibration, and then vibration travels through the phone body to affect both the accelerometers and gyroscopes. Vibration acts as linear acceleration in this case, which the accelerometer is designed to measure. While the gyroscope is not designed to measure linear acceleration, its sensing mass(es) still vibrates and these vibrations are quantized. Thus, the intent of each sensor changes the effectiveness of this particular scenario.

#### 6.5.1.3 Selective integration of sensor axes.

Selective integration of axis data only achieved significantly higher results for one phone model, the Google Pixel 1, but it did improve accuracy versus a single axis for all but one case. This case was the test for the Google Pixel 2 in the conference room, and it could not improve accuracy as accuracy was already 100%. For all phones but the Pixel 1, the improvement was limited because the results were already near 100%

accuracy. However, for the Pixel 1 the selective-axis integration improved as much as 40% over single-axis accuracies. This indicates that in cases with noisier data, the selective axis integration could help a classifier model utilize the various touchtone information in each axis to achieve higher accuracies.

### 6.5.2 Mitigation strategy evaluations

#### 6.5.2.1 Reduced Sampling Rates

The results for the reduced sampling rates mitigation support the theoretical analysis in Section 6.3.2.1 to show that this approach does not greatly affect touchtone eavesdropping until sampling rates are reduced significantly (Fig 6.10a). Once again, this is because touchtone aliases will remain in the digitized signal no matter the sampling rate, and this mitigation affects eavesdropping accuracy by reducing the total information available (affected functionality of benign applications). More concretely in our results, the reduced sampling rate does not seem to have much of an effect until the sampling frequency is roughly 100 Hz, and thus bandwidth reaches around 50 Hz.

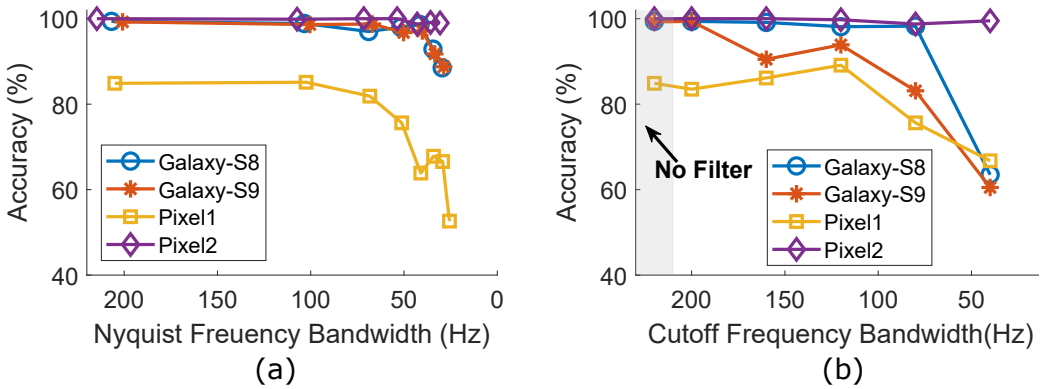



Figure 6.10: (a) Results for the downsampling mitigation with listed bandwidth equivalent to half the used sampling rate. (b) Results for the low-pass filter mitigation with listed bandwidth equivalent to the cutoff frequency used. Both mitigations do not greatly reduce touchtone eavesdropping accuracy until bandwidth is under 50 Hz, which could hinder functionality for benign applications.

#### 6.5.2.2 Software Low-pass Filter

Our evaluation supports the intuition (Section 6.3.2.2) that software low-pass filters do not reduce classifier accuracy without significantly reducing available signal bandwidth. As Fig 6.10b demonstrates, in our tests the touchtone accuracy results were only minimally affected until a very low 40 Hz cutoff frequency was reached. In fact in some cases, such as with the Pixel 1's accuracy results, the average accuracy improved when using cutoffs of 160 Hz and 120 Hz, which could indicate a large amount of noise in the 160 Hz to 200 Hz range for that particular phone. However, for the Pixel 2 the accuracy remained unaffected even at a 40 Hz cutoff frequency.

#### 6.5.2.3 Software and Hardware Anti-aliasing Filter

Our experiments show that anti-aliasing filters are more effective than either pure low-pass filters or sampling rate reduction. For example, with a cutoff frequency/bandwidth of 100 Hz, the order 5 and 8 software anti-aliasing filters reduce the touchtone eavesdropping accuracy from over 99% to below 40% (see Fig 6.11), while neither the pure low-pass filter nor the reducing sampling rate mitigation could reduce the accuracy to below 80%.

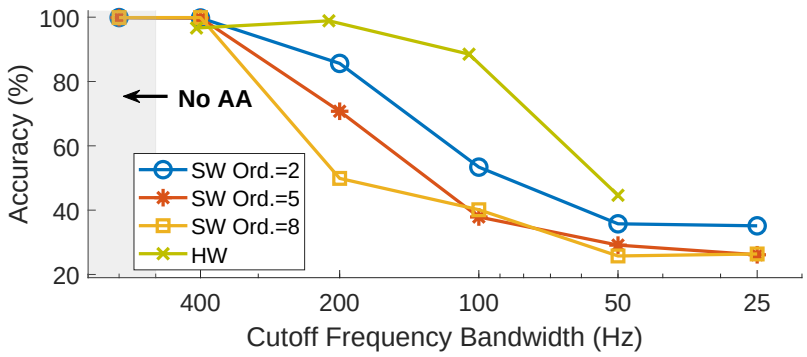



Figure 6.11: While not completely eliminating the attack, the software anti-aliasing filter is able to significantly reduce the accuracy of the touchtone eavesdropper and speech snooper attacks. Additionally, with better hardware implementations (higher sampling rates), this mitigation could be even more effective while preserving bandwidth.

Both the software and hardware anti-aliasing filter work to some degree, but neither are perfect mitigations. Most manufacturers may expect a built-in, hardware-based "anti-aliasing filter" to fully filter aliases without looking into the details. While somewhat effective, it was the software-based solution we found to work best. We do not currently know the exact filter parameters of the hardware filter due to the black-box nature of the design. However, the hardware implementation would likely increase in effectiveness by increasing filter order, as seen in our software implementation. The 8th order, 200 Hz bandwidth software anti-aliasing filter is likely the best solution from our experiments as it preserves the 200 Hz bandwidth used by most of our phones while still having a significant effect on the attacks.

## 6.6 Discussion

### 6.6.1 Hardware solutions

Some mitigations embedded into circuitry could serve as long-term solutions. Analog filtering mitigations schemes should work well against touchtone leakage as they can directly attenuate the original touchtone frequencies before sampling, and therefore before aliasing. Additionally, randomized sampling could mitigate some of the aliasing effects.

### 6.6.2 Application to other acoustic leakages

This chapter focuses on touchtones, but lessons are largely applicable to other forms of acoustic leakage. I focus on touchtone leakage as it provides a high-impact, yet simple signal for attack and mitigation analysis when compared to other potential targets such as speech. Touchtones are difficult to mitigate due to being intentionally designed to be easily recognizable in the presence of noise. Other targets, such as speech, are likely more affected by noise due to signal complexity. Thus, mitigation

strategies that are effective against touchtones should remain effective against speech.

## 6.7 Conclusion

This chapter examines how to mitigate recognizable artifacts of touchtones, representing sensitive user input such as credit card number or selections in an automated telephony service, from leaking into smartphone motion sensor data. Our experiments show that an adversary with understanding of relevant physics and signal processing concepts can use this motion sensor data to recover user input at >99% accuracy. Some of the more obvious mitigations, such as software low-pass filters or reduced sampling rates, are ineffective without significantly reducing information available to all applications. I instead propose software and hardware digital anti-aliasing filtering designs which achieve moderate success and can be implemented as a software update.

# CHAPTER VII

# Discussion

## 7.1 Transduction Mitigations Applied to Oversensing

Many of the ideas for transduction vulnerability mitigations shown in Chapter III in Table 3.3 may also apply to oversensing vulnerabilities. Specifically, the transduction mitigations often apply to the out-of-scope and unintended stimuli categories of oversensing (Chapter V).

Furthermore, the Transduction Attack Model (TAM) could be used to describe some vulnerabilities in sensor circuitry for oversensing. For example, take Walnut [143] and the touchtone eavesdropping attack shown in Chapter VI. Both vulnerabilities could be described in TAM as using:

$$\text{[pre-transducer injection]} \longrightarrow \text{insufficient filtering} \longrightarrow \text{aliasing}$$

The difference is that while in Walnut the interaction of the acoustic waves and the transducer is done intentionally, *i.e.,* injection, this same physical interaction occurs unintentionally for the touchtone eavesdropping attack.

Oversensing and transduction vulnerabilities that share the same or similar physical causes, *i.e.,* a signal interacting with sensor output, could rely on similar defense methods. For example, Dolphin Attack [165] describes a transduction vulnerability

where ultrasonic acoustic waves affect the output of a microphone intended for audible sound. The sound first injects a signal into the circuit via the microphone's transducer. Then to complete the attack the adversary makes use of two measurement shaping steps: (1) intermodulation distortion in the transducer and amplifier, and then (2) filtering to remove traces of the original signal. This transduction vulnerability has similarities with user location tracking via ultrasonic beacons [13], an oversensing vulnerability. The exact signal path in the sensor circuitry for the oversensing vulnerability is yet unknown, but one could use (as described by TAM) either of the below methods to receive ultrasonic information used for tracking:

- pre-transducer injection $\longrightarrow$ IMD $\longrightarrow$ filtering
- pre-transducer injection $\longrightarrow$ aliasing

Thus, using TAM mitigations for IMD, filtering, or aliasing could result in this oversensing vulnerability being mitigated.

## 7.2 Attacks Using Both Transduction and Oversensing Simultaneously

Transduction and oversensing vulnerabilities are not mutually exclusive, and particular attacks could be transduction and oversensing simultaneously. For example, the oversensing attack listed in the previous section, user location via ultrasonic beacons [13], could be thought of as a combined attack. The emitted ultrasonic waves are essentially a signal crafted to inject a certain pattern in the sensor output, a transduction vulnerability; this pattern can then be received by the attacker via a microphone permission, an oversensing vulnerability.

Combining transduction and oversensing vulnerabilities could lead to many possible attacks, such as covert channels. A transduction vulnerability would allow for an

adversary to covertly send data. Then an oversensing vulnerability could allow the adversary to recover this data elsewhere.

# CHAPTER VIII

# Conclusion

This dissertation investigates the question of "How is systemic design for sensing vulnerabilities possible?" While there is still work remaining, the work in this document shows how this goal is reachable. It provides language and models to describe this wide threat space, showing the similarities between many of the vulnerabilities. Then it provides mitigations patterns, either from existing work or new, to defend against these classes of vulnerabilities. In summary, this dissertation's contributions include:

- *Categorizing and analyzing* the space of physical sensing vulnerabilities. The first "categorization" is transduction and oversensing vulnerabilities. Transduction vulnerabilities are then further described and systematized using the Transduction Attack Model (TAM). Oversensing vulnerabilities can call into one of several described categories, and adversaries can recover information in one of several described patterns described in the Oversensing Anti-System (OA-Sys).

- *Mitigation design patterns* are provided for both transduction and oversensing vulnerabilities. TAM was used to systematize past mitigation strategies and shows how each of these strategies could be applicable to other transduction

attacks, even across sensor designs. OA-Sys provides patterns to mitigate oversensing vulnerabilities for the most common sensing uses in smartphones.

- *Detailed case studies* are provided for transduction and oversensing vulnerabilities in (respectively) Blue Note and Touchtone Eavesdropping. These chapters provide intuition on how the high-level concepts and strategies provided in the other chapters map when applied to specific problems.

Looking forward, the work in this dissertation provides opportunities to advance the research community and provide practical solutions to manufacturers and operating system designers. The models and language provided in this dissertation provide methods to describe vulnerabilities for physical attacks on sensors and reveal the previously opaque problems preventing mitigation. This ability to communicate and describe both prevents confusion on similarities or differences between research and illuminates when a vulnerability investigated on one sensor could map to other sensors.

In addition, this dissertation provides concrete mitigation strategies for many known physical vulnerabilities on sensors, across both transduction and oversensing vulnerabilities, in an organized fashion. Manufacturers or operating system designers will still have to design their specific solution, but this work provides a beginning for design.

In summary, as sensor systems become ever more prevalent and impactful on everyday life – driving our entertainment, transportation, manufacturing capability, and more – I believe this work provides a foundation to address the problems presented by physical attacks on sensing systems.

# BIBLIOGRAPHY

# BIBLIOGRAPHY


[1] Serial ATA II Native Command Queuing Overview Application Note. Technical report, Intel, April 2003. `http://download.intel.com/support/chipsets/imsm/sb/sata2_ncq_overview.pdf`.

[2] Design Hard-Disk Read/Write Head Controller, 2017. Accessed: 2017-09-22.

[3] Driver stacks, 2017. Accessed: 2017-10-30.

[4] Interpreting Event 153 Errors, 2017. Accessed: 2017-10-30.

[5] Minidrivers, Miniport drivers, and driver pairs, 2017. Accessed: 2017-10-30.

[6] Multi-Tier Reset in Storport, 2017. Accessed: 2017-10-30.

[7] Queuing and Dequeuing IRPs, 2017. Accessed: 2017-10-30.

[8] Understanding Storage Timeouts and Event 129 Errors, 2017. Accessed: 2017-10-30.

[9] What is the normal operating temperature for Seagate disk drives?, 2017. Accessed: 2017-05-17.

[10] AirSafe.com. COPA airlines plane crashes. http://www.airsafe.com/events/airlines/copa.html, 1992.

[11] Mohammad Abdullah Al Faruque, Sujit Rokka Chhetri, Arquimedes Canedo, and Jiang Wan. Acoustic side-channel attacks on additive manufacturing systems. In *2016 ACM/IEEE 7th international conference on Cyber-Physical Systems (ICCPS)*, pages 1–10. IEEE, 2016.

[12] S Abhishek Anand, Chen Wang, Jian Liu, Nitesh Saxena, and Yingying Chen. Spearphone: A speech privacy exploit via accelerometer-sensed reverberations from smartphone loudspeakers. *arXiv preprint arXiv:1907.05972*, 2019.

[13] Daniel Arp, Erwin Quiring, Christian Wressnegger, and Konrad Rieck. Privacy threats through ultrasonic side channels on mobile devices. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 35–47. IEEE, 2017.

[14] K. O. Aung, C. Shankaran, R. Sbiaa, E. L. Tan, S. K. Wong, and S. N. Piramanayagam. Achieving High Aspect Ratio of Track Length to Width in Molds for Discrete Track Recording Media. *Research Letters in Nanotechnology*, 2008:1–4, 2008.

[15] Michael Backes, Markus Dürmuth, Sebastian Gerling, Manfred Pinkal, and Caroline Sporleder. Acoustic side-channel attacks on printers. In *Proceedings of the 19th USENIX Conference on Security*, USENIX Security'10, page 20, USA, 2010. USENIX Association.

[16] Sebastian Biedermann, Stefan Katzenbeisser, and Jakub Szefer. *Hard Drive Side-Channel Attacks Using Smartphone Magnetic Field Sensors*. Springer Berlin Heidelberg, 2015.

[17] Leif Bjørnø. Introduction to nonlinear acoustics. *Physics Procedia*, 3(1):5–16, 2010.

[18] Hristo Bojinov, Yan Michalevsky, Gabi Nakibly, and Dan Boneh. Mobile device identification via sensor fingerprinting. *arXiv preprint arXiv:1408.1416*, 2014.

[19] C. Bolton, S. Rampazzi, C. Li, A. Kwong, W. Xu, and K. Fu. Blue note: How intentional acoustic interference damages availability and integrity in hard disk drives and operating systems. In *Proceedings of the 39th IEEE Symposium on Security and Privacy (SP)*, pages 1048–1062, May 2018.

[20] Raj Bridgelall. Inertial sensor sample rate selection for ride quality measures. *Journal of Infrastructure Systems*, 21(2):04014039, 2015.

[21] Liang Cai and Hao Chen. Touchlogger: Inferring keystrokes on touch screen from smartphone motion. In *Proceedings of the 6th USENIX Workshop on Hot Topics in Security (HotSec)*, pages 1–9, 2011.

[22] Health Canada. Guidelines for the safe use of ultrasound: Part ii—industrial and commercial applications safety code 24, 1991.

[23] Simon Castro, Robert Dean, Grant Roth, George T Flowers, and Brian Grantham. Influence of acoustic noise on the dynamic performance of MEMS gyroscopes. In *ASME 2007 International Mechanical Engineering Congress and Exposition*, pages 1825–1831. American Society of Mechanical Engineers, 2007.

[24] Ruchir Chauhan. A platform for false data injection in frequency modulated continuous wave radar. Master's thesis, Utah State University, 2014.

[25] Tianqi Chen and Carlos Guestrin. Xgboost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug 2016.

[26] Yuxin Chen, Huiying Li, Steven Nagels, Zhijing Li, Pedro Lopes, Ben Y Zhao, and Haitao Zheng. Understanding the effectiveness of ultrasonic microphone jammer. *arXiv preprint arXiv:1904.08490*, 2019.

[27] Yun Chan Cho and Jae Wook Jeon. Remote robot control system based on dtmf of mobile phone. In *2008 6th IEEE International Conference on Industrial Informatics*, pages 1441–1446. IEEE, 2008.

[28] B Cochrun and Arvin Grabel. A method for the determination of the transfer function of electronic circuits. *IEEE Transactions on Circuit Theory*, 20(1):16–20, 1973.

[29] Bernard Cole. The design challenges of a trillion sensor world. https://www.embedded.com/electronics-blogs/cole-bin/4433743/The-design-challenges-of-a-trillion-sensor-world, 2014.

[30] LRAD Corporation. LRAD 2000X. `https://www.dropbox.com/s/4qth9beayjx5gxr/LRAD_Datasheet_2000X.pdf?dl=0`, 2017. Accessed: 2017-05-19.

[31] Drew Davidson, Hao Wu, Robert Jellinek, Thomas Ristenpart, Cornell Tech, and Vikas Singh. Controlling UAVs with sensor input spoofing attacks. In *10th USENIX Workshop on Offensive Technologies*, pages 221–231, 2016.

[32] Albert Dayes and John Treder. Drive Performance-TMR. `http://www.logicsmith.com/performance.html`, 2017. Accessed: 2017-05-15.

[33] Android Developers. *Android Debug Bridge*. `https://developer.android.com/studio/command-line/adb`.

[34] Harijono Djojodihardjo. Vibro-acoustic analysis of the acoustic-structure interaction of flexible structure due to acoustic excitation. *Acta Astronautica*, 108:129–145, 2015.

[35] Trinoy Dutta and Andrew R Barnard. Performance of hard disk drives in high noise environments. *Noise Control Engineering Journal*, 65(5):386–395, 2017.

[36] John Eargle. *The Microphone Book: From mono to stereo to surround-a guide to microphone design and application*. CRC Press, 2012.

[37] Joshua J Engelsma, Sunpreet S Arora, Anil K Jain, and Nicholas G Paulter. Universal 3D wearable fingerprint targets: advancing fingerprint reader evaluations. *IEEE Transactions on Information Forensics and Security (TIFS)*, 13(6):1564–1578, 2018.

[38] Jose Lopes Esteves and Chaouki Kasmi. Remote and silent voice command injection on a smartphone through conducted IEMI: Threats of smart IEMI for information security. Technical report, Wireless Security Lab, French Network and Information Security Agency (ANSSI), 2018.

[39] Martin A Fischler and Robert C Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[40] Denis Foo Kune, John Backes, Shane S Clark, Daniel Kramer, Matthew Reynolds, Kevin Fu, Yongdae Kim, and Wenyuan Xu. Ghost talk: Mitigating EMI signal injection attacks against analog sensors. In *Proceedings of the 34th IEEE Symposium on Security and Privacy (SP)*, pages 145–159. IEEE, 2013.

[41] Denis Foo Kune, John Backes, Shane S. Clark, Daniel B. Kramer, Matthew R. Reynolds, Kevin Fu, Yongdae Kim, and Wenyuan Xu. Ghost Talk: Mitigating EMI Signal Injection Attacks against Analog Sensors. In *Proceedings of the 34th Annual IEEE Symposium on Security and Privacy*, May 2013.

[42] Jacob Fraden. *Handbook of modern sensors: physics, designs, and applications*. Springer Science & Business Media, 2004.

[43] Todd C. Frankel. Sensor cited as potential factor in Boeing crashing draws scrutiny. `https://www.washingtonpost.com/business/economy/sensor-cited-as-potential-factor-in-boeing-crashes-draws-scrutiny/2019/03/17/5ecf0b0e-4682-11e9-aaf8-4512a6fe3439_story.html`, 2018.

[44] Kevin Fu and Wenyuan Xu. Risks of trusting the physics of sensors. *Communications of the ACM*, 61(2):20–23, 2018.

[45] Ilias Giechaskiel, Youqian Zhang, and Kasper B Rasmussen. A framework for evaluating security in the presence of signal injection attacks. *arXiv preprint arXiv:1901.03675*, 2019.

[46] DV Giri and FM Tesche. Classification of intentional electromagnetic environments (IEME). *IEEE Transactions on Electromagnetic Compatibility*, 46(3):322–328, 2004.

[47] Dan Goodin. Tiktok and 32 other ios apps still snoop your sensitive clipboard data. *ars Technica*, 2020.

[48] Google. *Android Developers: SensorEvent specification,*. `http://developer.android.com/reference/android/hardware/SensorEvent.html`.

[49] T. Grandke and W. H. Ko. *Sensors a Comprehensive Survey, Vol. 1: Fundamentals and General Aspects*. VCH, New York, 1989.

[50] Raffaele Gravina, Parastoo Alinia, Hassan Ghasemzadeh, and Giancarlo Fortino. Multi-sensor fusion in body sensor networks: State-of-the-art and research challenges. *Information Fusion*, 35:68–80, 2017.

[51] Aditi Sharma Grover, Madelaine Plauché, Etienne Barnard, and Christiaan Kuun. Hiv health information access using spoken dialogue systems: Touchtone vs. speech. In *2009 International Conference on Information and Communication Technologies and Development (ICTD)*, pages 95–107. IEEE, 2009.

[52] Mordechai Guri, Matan Monitz, Yisroel Mirski, and Yuval Elovici. BitWhisper: Covert Signaling Channel between Air-Gapped Computers Using Thermal Manipulations. pages 276–289, 2015.

[53] Mordechai Guri, Yosef Solewicz, Andrey Daidakulov, and Yuval Elovici. Disk-Filtration: Data Exfiltration from Speakerless Air-Gapped Computers via Covert Hard Drive Noise. *arXiv preprint arXiv:1608.03431*, 2016.

[54] Dan Hallenbeck, Michael Lack, Jason McPeak, and Irwin Reyes. Privacy enhancements for android. `https://github.com/twosixlabs/PE_for_Android/blob/master/PE_for_Android_whitepaper.pdf`.

[55] Mark F Hamilton, David T Blackstock, et al. *Nonlinear acoustics*, volume 1. Academic press San Diego, 1998.

[56] Jun Han, Albert Jin Chung, and Patrick Tague. Pitchin: Eavesdropping via intelligible speech reconstruction using non-acoustic sensor fusion. In *16th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2017.

[57] Jun Han, Emmanuel Owusu, Le T. Nguyen, Adrian Perrig, and Joy Zhang. Accomplice: Location inference using accelerometers on smartphones. In *Communication Systems and Networks (COMSNETS)*, 2012.

[58] Matt Hatton. The iot in 2030: Which applications account for the biggest chunk of the $1.5 trillion opportunity? 2020.

[59] Jason I Hong, Yuvraj Agarwal, Matt Fredrikson, Mike Czapik, Shawn Hanna, Swarup Sahoo, Judy Chun, Won-Woo Chung, Aniruddh Iyer, Ally Liu, et al. The design of the user interfaces for privacy enhancements for android. *arXiv preprint arXiv:2104.12032*, 2021.

[60] Paul Horowitz and Winfield Hill. *The art of electronics*. Cambridge Univ. Press, 1989.

[61] Yih-Chun Hu, Adrian Perrig, and David B Johnson. Wormhole attacks in wireless networks. *IEEE Journal on Selected Areas in Communications*, 24(2):370–380, 2006.

[62] Zhe Hu, Lu Yuan, Stephen Lin, and Ming-Hsuan Yang. Image deblurring using smartphone inertial sensors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1855–1864, 2016.

[63] Intel. *Intel NUC*. `https://www.intel.com/content/www/us/en/products/boards-kits/nuc.html`.

[64] Elya B Joffe and Kai-Sang Lock. *Grounds for grounding: A circuit to system handbook*. John Wiley & Sons, 2011.

[65] Alexandre Karpenko, David Jacobs, Jongmin Baek, and Marc Levoy. Digital video stabilization and rolling shutter correction using gyroscopes. *CSTR*, 1(2):13, 2011.

[66] Chaouki Kasmi and Jose Lopes Esteves. IEMI threats for information security: Remote command injection on modern smartphones. *IEEE Transactions on Electromagnetic Compatibility*, 57(6):1752–1755, 2015.

[67] Erkan Kaymak, Mark Atherton, K R G Rotter, and B Millar. Active Noise Control at High Frequencies. In *13th International Congress on Sound and Vibration*, 2006.

[68] Adil Mehmood Khan, Muhammad Hameed Siddiqi, and Seok-Won Lee. Exploratory data analysis of acceleration signals to select light-weight and accurate features for real-time activity recognition on smartphones. *Sensors*, 13(10):13099–13122, 2013.

[69] Kim, Y., C. Kang, and Masayoshi Tomizuka. Adaptive and optimal rejection of non-repeatable disturbance in hard disk drives. In *IEEE/ASME Int. Conf. Advanced Intelligent Mechatronics*, Monterey, California, August 2005.

[70] Ernst-Christian Koch. Review on pyrotechnic aerial infrared decoys. *Propellants, Explosives, Pyrotechnics*, 26(1):3–11, 2001.

[71] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Annual international cryptology conference*, pages 388–397. Springer, 1999.

[72] Tuljappa M Ladwa, Sanjay M Ladwa, R Sudharshan Kaarthik, Alok Ranjan Dhara, and Nayan Dalei. Control of remote domestic system using dtmf. In *International Conference on Instrumentation, Communication, Information Technology, and Biomedical Engineering 2009*, pages 1–6. IEEE, 2009.

[73] Seoungjun Lee, Dongsoo Har, and Dongsuk Kum. Drone-assisted disaster management: Finding victims via infrared camera and lidar sensor fusion. In *2016 3rd Asia-Pacific World Congress on Computer Science and Engineering (APWC on CSE)*, pages 84–89. IEEE, 2016.

[74] Tianshi Li, Yuvraj Agarwal, and Jason I Hong. Coconut: An ide plugin for developing privacy-friendly apps. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(4):1–35, 2018.

[75] Yuanchun Li, Fanglin Chen, Toby Jia-Jun Li, Yao Guo, Gang Huang, Matthew Fredrikson, Yuvraj Agarwal, and Jason I Hong. Privacystreams: Enabling transparency in personal data processing for mobile apps. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(3):1–26, 2017.

[76] Bruce D Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial intelligence*. Morgan Kaufmann Publishers Inc., 1981.

[77] Mark W Maciejewski, Harry Z Qui, Iulian Rujan, Mehdi Mobli, and Jeffrey C Hoch. Nonuniform sampling and spectral aliasing. *Journal of Magnetic Resonance*, 199(1):88–93, 2009.

[78] Abdullah Al Mamunm, GuoXiao Guo, and Chao Bi. *Hard Disk Drive: Mechatronics and Control*. CRC Press, 2006.

[79] Philip Marquardt, Arunabh Verma, Henry Carter, and Patrick Traynor. (sp)iphone: Decoding vibrations from nearby keyboards using mobile phone accelerometers. In *Conference on Computer and Communications Security (CCS)*, New York, NY, USA, 2011. ACM.

[80] S. Maruyama, S. Wakabayashi, and T. Mori. Tap 'n ghost: A compilation of novel attack techniques against smartphone touchscreens. In *Proceedings of the 40th IEEE Symposium on Security and Privacy (SP)*, pages 628–645. IEEE, 2019.

[81] Václav Matyáš and Zdeněk Říha. Biometric authentication—security and usability. In *Advanced communications and multimedia security*, pages 227–239. Springer, 2002.

[82] N. Matyunin, J. Szefer, S. Biedermann, and S. Katzenbeisser. Covert channels using mobile device's magnetic field sensors. In *Asia and South Pacific Design Automation Conference*, pages 525–532, 2016.

[83] Yan Michalevsky, Dan Boneh, and Gabi Nakibly. Gyrophone: Recognizing speech from gyroscope signals. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 1053–1067, San Diego, CA, August 2014. USENIX Association.

[84] Yan Michalevsky, Dan Boneh, and Gabi Nakibly. Gyrophone: Recognizing Speech from Gyroscope Signals. In *USENIX Security*, pages 1053–1067, 2014.

[85] Yan Michalevsky, Aaron Schulman, Gunaa Arumugam Veerapandian, Dan Boneh, and Gabi Nakibly. Powerspy: Location tracking using mobile device power analysis. In *24th {USENIX} Security Symposium ({USENIX} Security 15)*, pages 785–800, 2015.

[86] Emiliano Miluzzo, Alexander Varshavsky, Suhrid Balakrishnan, and Romit Roy Choudhury. Tapprints: Your finger taps have fingerprints. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, MobiSys '12, pages 323–336, New York, NY, USA, 2012. ACM.

[87] Devaprakash Muniraj and Mazen Farhood. Detection and mitigation of actuator attacks on small unmanned aircraft systems. *Control Engineering Practice*, 83:188–202, 2019.

[88] Joseph F Murray, Gordon F Hughes, and Kenneth Kreutz-Delgado. Hard drive failure prediction using non-parametric statistical methods. In *Proceedings of ICANN/ICONIP*, 2003.

[89] S. Narain, T. D. Vo-Huu, K. Block, and G. Noubir. Inferring user routes and locations using zero-permission mobile sensors. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 397–413, May 2016.

[90] Sashank Narain, Amirali Sanatinia, and Guevara Noubir. Single-stroke language-agnostic keylogging using stereo-microphones and domain specific machine learning. In *Proceedings of the 2014 ACM conference on Security and privacy in wireless & mobile networks*, pages 201–212, 2014.

[91] Shoei Nashimoto, Daisuke Suzuki, Takeshi Sugawara, and Kazuo Sakiyama. Sensor con-fusion: Defeating kalman filter in signal injection attack. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security (AsiaCCS)*, pages 511–524. ACM, 2018.

[92] NDT Resource Center. Attenuation of sound waves. `https://www.nde-ed.org/EducationResources/CommunityCollege/Ultrasonics/Physics/attenuation.htm`.

[93] Phillip Nguyen, Alex Silence, David Darais, and Joseph P Near. Duetsgx: Differential privacy with secure hardware. *arXiv preprint arXiv:2010.10664*, 2020.

[94] Norman S Nise. *Control Systems Engineering*. John Wiley & Sons, 2007.

[95] Katsuhiko Ogata and Yanjuan Yang. *Modern Control Engineering*. Prentice-Hall, 2002.

[96] Kanji Ono, Hideo Cho, and Takuma Matsuo. Transfer functions of acoustic emission sensors. *Journal of Acoustic Emission*, 26:72–91, 2008.

[97] Alan V. Oppenheim and Ronald W. Schafer. *Discrete-Time Signal Processing*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2009.

[98] Alfredo Ortega. Turning hard disk drives into accidental microphones. Ekoparty, October 2017.

[99] Emmanuel Owusu, Jun Han, Sauvik Das, Adrian Perrig, and Joy Zhang. Accessory: Password inference using accelerometers on smartphones. In *HotMobile*, HotMobile, New York, NY, USA, 2012. ACM.

[100] Phillip E Pace. *Detecting and classifying low probability of intercept radar*. Artech House, 2009.

[101] Youngseok Park, Yunmok Son, Hocheol Shin, Dohyun Kim, and Yongdae Kim. This ain't your dose: Sensor Spoofing Attack on Medical Infusion Pump. In *10th USENIX Workshop on Offensive Technologies*, 2016.

[102] E Andrew Parr. *Logic designer's handbook: circuits and systems*. Elsevier, 2013.

[103] Giorgio De Pasquale, Libor Rufer, Skandar Basrour, and Aurelio Soma. Modeling and validation of acoustic performances of micro-acoustic sources for hearing applications. *Sensors and Actuators A: Physical*, 247:614–628, 2016.

[104] Jonathan Petit, Bas Stottelaar, Michael Feiri, and Frank Kargl. Remote attacks on automated vehicles sensors: Experiments on camera and lidar. *Black Hat Europe*, 2015.

[105] Thilo Pfau and Patrick Reilly. How low can we go? influence of sample rate on equine pelvic displacement calculated from inertial sensor data. *Equine Veterinary Journal*, 53(5):1075–1081, 2021.

[106] Eduardo Pinheiro, Wolf-Dietrich Weber, and Luiz André Barroso. Failure Trends in a Large Disk Drive Population. In *USENIX FAST*, volume 7, pages 17–23, 2007.

[107] Rahul Potharaju, Andrew Newell, Cristina Nita-Rotaru, and Xiangyu Zhang. Plagiarizing smartphone applications: attack strategies and defense techniques. In *International symposium on engineering secure software and systems*, pages 106–120. Springer, 2012.

[108] Soundarya Ramesh, Harini Ramprasad, and Jun Han. Listen to your key: Towards acoustics-based physical key inference. In *Proceedings of the 21st International Workshop on Mobile Computing Systems and Applications*, pages 3–8, 2020.

[109] Soundarya Ramesh, Harini Ramprasad, and Jun Han. Listen to your key: Towards acoustics-based physical key inference. In *Proceedings of the 21st International Workshop on Mobile Computing Systems and Applications*, HotMobile '20, page 3–8, New York, NY, USA, 2020. Association for Computing Machinery.

[110] Kasper Bonne Rasmussen, Claude Castelluccia, Thomas S Heydt-Benjamin, and Srdjan Capkun. Proximity-based access control for implantable medical

devices. In *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS)*, pages 410–419. ACM, 2009.

[111] Brian Rawson and Kent Green. Inert Gas Data Center Fire Protection and Hard Disk Drive Damage. Technical report, The Datacenter Journal, August 2012. `http://www.datacenterjournal.com/inert-gas-data-center-fire-protection-and-hard-disk-drive-damage/`.

[112] Erik Riedel. Personal Communication, January 2018.

[113] Cyrous Rostamzadeh, Flavio Canavero, Feraydune Kashefi, and Mehdi Darbandi. Effectiveness of multilayer ceramic capacitors for electrostatic discharge protection. *In Compliance Magazine*, 2012.

[114] Nirupam Roy, Haitham Hassanieh, and Romit Roy Choudhury. Backdoor: Making microphones hear inaudible sounds. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 2–14. ACM, 2017.

[115] Nirupam Roy, Sheng Shen, Haitham Hassanieh, and Romit Roy Choudhury. Inaudible voice commands: The long-range attack and defense. In *Proceedings of the 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 547–560. USENIX Association, 2018.

[116] Seref Sagiroglu and Gurol Canbek. Keyloggers: Increasing threats to computer security and privacy. *IEEE technology and society magazine*, 28(3):10–17, 2009.

[117] Yasser Shoukry Sakr. *Security and Privacy in Cyber-Physical Systems: Physical Attacks and Countermeasures*. PhD thesis, UCLA, 2015.

[118] Jerome H Saltzer and Michael D Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975.

[119] Derek Sandahl, Alan Elder, and Andrew Barnard. The Impact of Sound on Computer Hard Disk Drives and Risk Mitigation Measures. Technical report, Tyco, Michigan Technical University, 2015. `https://www.ansul.com/en/us/DocMedia/T-2016367.PDF`.

[120] Jayaprakash Selvaraj. *Intentional Electromagnetic Interference Attack on Sensors and Actuators*. PhD thesis, Iowa State University, 2018.

[121] Jayaprakash Selvaraj, Gökçen Yılmaz Dayanıklı, Neelam Prabhu Gaunkar, David Ware, Ryan M. Gerdes, and Mani Mina. Electromagnetic induction attacks against embedded systems. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security (AsiaCCS)*, pages 499–510. ACM, 2018.

[122] A. Shabtai, Y. Fledel, U. Kanonov, Y. Elovici, S. Dolev, and C. Glezer. Google android: A comprehensive security assessment. *IEEE Security Privacy*, 8(2):35–44, March 2010.

[123] Claude Elwood Shannon. Communication in the presence of noise. *Proceedings of the Institute of Radio Engineers*, 86(2):447–457, 1998.

[124] Kirti Sharma and Manish Paradakar. The melamine adulteration scandal. *Food Security*, 2(1):97–107, Mar 2010.

[125] Rohit Sharma, Kushagra Kumar, and Shashank Vig. Dtmf based remote control system. In *2006 IEEE International Conference on Industrial Technology*, pages 2380–2383. IEEE, 2006.

[126] Hao Shen, Weiming Zhang, Han Fang, Zehua Ma, and Nenghai Yu. Jamsys: Coverage optimization of a microphone jamming system based on ultrasounds. *IEEE Access*, 7:67483–67496, 2019.

[127] Hocheol Shin, Dohyun Kim, Yujin Kwon, and Yongdae Kim. Illusion and dazzle: Adversarial optical channel exploits against lidars for automotive applications. In *Proceedings of the 19th International Conference on Cryptographic Hardware and Embedded Systems (CHES)*, pages 445–467. Springer, 2017.

[128] Hocheol Shin, Yunmok Son, Youngseok Park, Yujin Kwon, and Yongdae Kim. Sampling race: Bypassing timing-based analog active sensor spoofing detection on analog-digital systems. In *Proceedings of the 10th USENIX Workshop on Offensive Technologies (WOOT)*, 2016.

[129] Yasser Shoukry, Paul Martin, Paulo Tabuada, and Mani Srivastava. Non-invasive spoofing attacks for anti-lock braking systems. In *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, pages 55–72. Springer, 2013.

[130] Yasser Shoukry, Paul Martin, Yair Yona, Suhas Diggavi, and Mani Srivastava. PyCRA: Physical challenge-response authentication for active sensors under spoofing attacks. In *Proceedings of the 2015 ACM Conference on Computer and Communications Security (CCS)*, pages 1004–1015, 2015.

[131] Siemens. Silent Extinguishing. Technical report, Siemens, September 2015. `https://www.downloads.siemens.com/download-center/d/White-Paper---Silent-Extinguishing-EN-PDF_A6V10699087_hq-en.pdf?mandator=ic_bt&segment=HQ&fct=downloadasset&pos=download&id1=A6V10699087`.

[132] Laurent Simon and Ross Anderson. Pin skimmer: Inferring pins through the camera and microphone. In *Proceedings of the Third ACM workshop on Security and privacy in smartphones & mobile devices*, pages 67–78, 2013.

[133] Ian Sinclair and John Dunton. *Electronic and Electrical Servicing, Second Edition: Consumer and Commercial Electronics*. Newnes, Newton, MA, USA, 2nd edition, 2007.

[134] Grahame Smillie. 2 - analogue modulation principles. In Grahame Smillie, editor, *Analogue and Digital Communication Techniques*, pages 15 – 45. Butterworth-Heinemann, Oxford, 1999.

[135] Yunmok Son, Hocheol Shin, Dongkwan Kim, Youngseok Park, Juhwan Noh, Kibum Choi, Jungwoo Choi, and Yongdae Kim. Rocking Drones with Intentional Sound Noise on Gyroscopic Sensors. In *24th USENIX Security Symposium (USENIX Security)*, pages 881–896, 2015.

[136] Liwei Song and Prateek Mittal. Inaudible voice commands. *arXiv preprint arXiv:1708.07238*, 2017.

[137] Jack Stewart. Why Tesla's autopilot can't see a stopped firetruck. `https://www.wired.com/story/tesla-autopilot-why-crash-radar`, 2018.

[138] Takeshi Sugawara, Benjamin Cyr, Sara Rampazzi, Daniel Genkin, and Kevin Fu. Light commands: laser-based audio injection attacks on voice-controllable systems. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*, pages 2631–2648, 2020.

[139] J.N. Teoh, C. Du, G. Guo, and L. Xie. Rejecting high frequency disturbances with disturbance observer and phase stabilized control. *Mechatronics*, 18(1):53–60, 2008.

[140] Tesla. A tragic loss. https://www.tesla.com/blog/tragic-loss, 2016.

[141] Texas Instruments. System-level ESD protection guide. Technical report, Texas Instruments, 2018.

[142] General Tools. *DSM403SD*. `https://www.generaltools.com/digital-tools/inspection-calibration-digital-tools/sound-level-meters/class-1-sound-level-meter-with-data-logging-sd-card`.

[143] Tim Trippel, Weisse Ofir, Wenyuan Xu, Peter Honeyman, and Kevin Fu. WALNUT: Waging Doubt on Integrity of MEMS Accelerometers by Injecting Acoustics. In *IEEE EuroS&P*, 2017.

[144] Tim Trippel, Weisse Ofir, Wenyuan Xu, Peter Honeyman, and Kevin Fu. WALNUT: Waging Doubt on Integrity of MEMS Accelerometers by Injecting Acoustics. In *IEEE EuroS&P*, 2017.

[145] Yannis Tsividis. Digital signal processing in continuous time: a possibility for avoiding aliasing and reducing quantization error. In *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages ii–589. IEEE, 2004.

[146] Yazhou Tu, Zhiqiang Lin, Insup Lee, and Xiali Hei. Injected and delivered: fabricating implicit control over actuation systems by spoofing inertial sensors. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 1545–1562, 2018.

[147] Yazhou Tu, Sara Rampazzi, Bin Hao, Angel Rodriguez, Kevin Fu, and Xiali Hei. Trick or heat? attack on amplification circuits to abuse critical temperature control systems. *arXiv preprint arXiv:1904.07110*, 2019.

[148] Vyacheslav Tuzlukov. *Signal processing noise*. CRC Press, 2002.

[149] International Telecommunication Union. Technical features of push-button telephone sets. *General Recommendations on Telephone Switching and Signalling*, 11 1988. `https://www.itu.int/rec/T-REC-Q.23-198811-I/en`.

[150] Matt Vasilogambros. Voting by phone is easy. but is it secure? `https://gcn.com/articles/2019/07/18/vote-by-phone.aspx`, 2019.

[151] Zhenbo Wang, Kang Wang, Bo Yang, Shangyuan Li, and Aimin Pan. Sonic gun to smart devices. *Black Hat USA*, 2017.

[152] David A Ware. Effects of intentional electromagnetic interference on analog to digital converter measurements of sensor outputs and general purpose input output pins. Master's thesis, Utah State University, 2017.

[153] Alma Whitten and J Doug Tygar. Why johnny can't encrypt: A usability evaluation of pgp 5.0. In *USENIX security symposium*, volume 348, pages 169–184, 1999.

[154] Wikipedia. List of sensors. `https://en.wikipedia.org/wiki/List_of_sensors`, 2019.

[155] Jon S Wilson. *Sensor technology handbook*. Elsevier, 2004.

[156] Mark J Wilson, Steven R Ford, and Paul L Rinaldo. *The ARRL Handbook for Radio Communications 2007*. Amer Radio Relay League, 2006.

[157] J. Xu and R. Tsuchiyama. Ultra-low-flying-height design from the viewpoint of contact vibration. *In Tribology International*, 36:459–466, 2003.

[158] W. Xu, C. Yan, W. Jia, X. Ji, and J. Liu. Analyzing and enhancing the security of ultrasonic sensors for autonomous vehicles. *IEEE Internet of Things Journal*, 5(6):5015–5029, Dec 2018.

[159] Chen Yan, Kevin Fu, and Wenyuan Xu. On cuba, diplomats, ultrasound, and intermodulation distortion. *Computers in Biology and Medicine*, 104:250–266, 2019.

[160] Chen Yan, Yan Long, Xiaoyu Ji, and Wenyuan Xu. The catcher in the field: A fieldprint based spoofing detection for text-independent speaker verification. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1215–1229, 2019.

[161] Chen Yan, Hocheol Shin, Connor Bolton, Wenyuan Xu, Yongdae Kim, and Kevin Fu. Sok: A minimalist approach to formalizing analog sensor security. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 233–248. IEEE, 2020.

[162] Chen Yan, Wenyuan Xu, and Jianhao Liu. Can you trust autonomous vehicles: Contactless attacks against sensors of self-driving vehicle. *DEF CON*, 2016.

[163] Chen Yan, Guoming Zhang, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. The feasibility of injecting inaudible voice commands to voice assistants. *IEEE Transactions on Dependable and Secure Computing*, 2019.

[164] Hyun Seok Yang, Jun Jeong, Cheol Hoon Park, and Young-Pil Park. Identification of contributors to HDD servo errors by measuring PES only. *IEEE Transactions on Magnetics*, 37(2):883–887, 2001.

[165] Guoming Zhang, Chen Yan, Xiaoyu Ji, Taimin Zhang, Tianchen Zhang, and Wenyuan Xu. DolphinAttack: Inaudible voice commands. In *Proceedings of the 2017 ACM Conference on Computer and Communications Security (CCS)*. ACM, 2017.

[166] Li Zhang, Parth H. Pathak, Muchen Wu, Yixin Zhao, and Prasant Mohapatra. Accelword: Energy efficient hotword detection through accelerometer. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '15, pages 301–315, New York, NY, USA, 2015. ACM.

[167] Yang Zhang, Peng Xia, Junzhou Luo, Zhen Ling, Benyuan Liu, and Xinwen Fu. Fingerprint attack against touch-enabled devices. In *Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices*, pages 57–68, 2012.

[168] Youqian Zhang and Kasper Rasmussen. Detection of electromagnetic interference attacks on sensor systems. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 203–216. IEEE, 2020.