Characterizing and Mitigating Touchtone Eavesdropping in Smartphone Motion Sensors

Connor Bolton*
University of Michigan
Ann Arbor, USA
mcbolto@umich.edu

Yan Long*
University of Michigan
Ann Arbor, USA
yanlong@umich.edu

Jun Han
Yonsei University
Seoul, Korea
jun.han@yonsei.ac.kr

Josiah Hester
Georgia Institute of Technology
Atlanta, USA
josiah@gatech.edu

Kevin Fu
Northeastern University
Boston, USA
k.fu@northeastern.edu

ABSTRACT

Smartphone motion sensors provide cybersecurity attackers with a stealthy way to eavesdrop on nearby acoustic information. Eavesdropping on touchtones emitted by smartphone speakers when users input numbers into their phones exposes sensitive information such as credit card information, banking PINs, and social security card numbers to malicious applications with access to only motion sensor data. This work characterizes this new security threat of touchtone eavesdropping by providing an analysis based on physics and signal processing theory. We show that advanced adversaries who selectively integrate data from multiple motion sensors and multiple sensor axes can achieve over 99% accuracy on recognizing 12 unique touchtones. We further design, analyze, and evaluate several mitigations which could be implemented in a smartphone update. We found that some apparent mitigations such as low-pass filters can undesirably reduce the motion sensor data to benign applications by 83% but only reduce an advanced adversary's accuracy by less than one percent. Other more informed designs such as anti-aliasing filters can fully preserve the motion sensor data to support benign application functionality while reducing attack accuracy by 50.1%.

CCS CONCEPTS

• **Security and privacy** → *Side-channel analysis and countermeasures.*

KEYWORDS

smartphone, motion sensor, eavesdropping, touchtone, DTMF

1 INTRODUCTION

Touchtones, the sounds produced by a smartphone when a numerical key is pressed, are an established communication standard widely used to encode user feedback in telephony channels [40]. In modern telephony systems, touchtones often encode important information such as credit card numbers (during call-based activation), bank pins, various account numbers, social security numbers, selections for various options in automated services, and possibly even votes in a phone-based federal election [41].

Recent side channel research has shown that sound produced by a smartphone's speaker may "leak" into the same phone's motion sensors, particularly speech audio during phone calls. This side-channel vulnerability results in a security breach in smartphone operating systems including the most prevalent Android system because third-party applications do not need any user permissions to use these motion sensors including gyroscopes and accelerometers. In contrast, the use of smartphone microphones, which are usually perceived as the sensor for receiving acoustic information, do require explicit user permissions. As a result, malicious applications may stealthily collect motion sensor data that can contain acoustic information without user notice.

Our work investigates *touchtone leakage*, a new security threat that this side-channel vulnerability causes where touchtone's acoustic information leaks into motion sensor data. Touchtone leakage occurs with a signal-to-noise ratio sufficient to be observed even visibly (Figure 1a). This leakage enables malicious smartphone applications (e.g., a seemingly benign health monitoring app running in the background) to eavesdrop on numerical user input that produces touchtones as shown in Figure 1. This work seeks to characterize the root causes and limits of touchtone leakage and eavesdropping. Specifically, we investigate why 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 a multitude of ways such as harmonics and aliases of harmonics. An adversary only needs to be able 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 instead utilize several of these manifestations simultaneously for a more proficient attack. We demonstrate these ideas by designing an eavesdropping classifier based on the XGBoost machine learning model. Our experiments with four Android smartphones suggest that 12 smartphone touchtones can be recovered by an adversary at over 99 % accuracies.

Besides revealing this new security threat of touchtone eavesdropping attacks, this paper also investigates the possible design improvements that can be employed to mitigate such acoustic eavesdropping attacks in future devices. Despite the previous side

*Equal-contribution co-first authors of this paper.

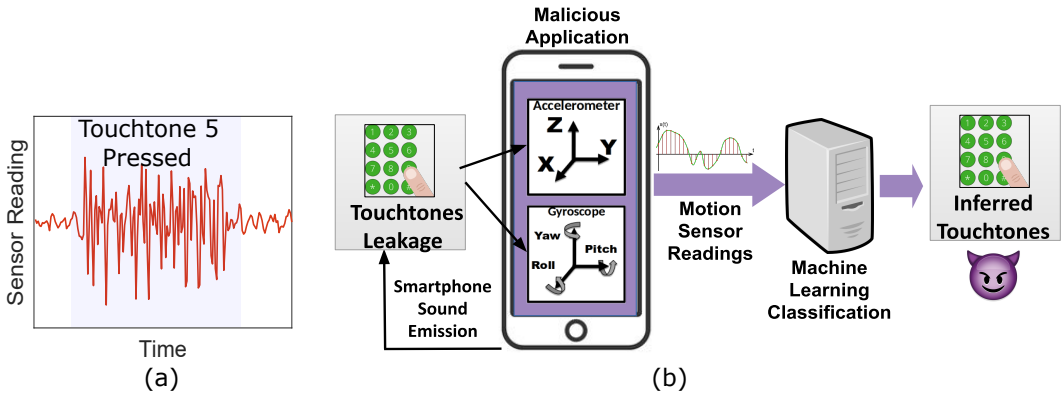

Figure 1: Touchtone leakage and eavesdropping. (a) A touchtone, indicating a "5" on a smartphone number pad, leaks into accelerometer data. (b) A malicious smartphone application can classify this leakage to discern that a "5" touchtone was emitted, inferring user input of a "5" for purposes such as dialing a phone number or inputting information into automated services.

channel research of motion sensor eavesdropping, reducing acoustic leakage remains an open research problem as previous papers focus more on adversarial exploitation than mitigation efforts [1, 2, 12, 14, 27]. This paper's second goal is to open discussion on how to reduce acoustic leakage into nearby motion sensors, using touchtone eavesdropping as an exemplary case study.

Specifically, our work analyzes functionality-aware, software-updatable mitigation designs for touchtone leakage. We observe that mitigations can reduce touchtone leakage by reducing the total information in motion 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 ineffective and effective solutions to demonstrate mitigation designs to follow or avoid. For example, we analyze and evaluate how some apparent mitigations briefly suggested in previous works such as sampling rate reduction and digital low-pass filtering are less effective at reducing touchtone leakage. We found sampling rate reduction can reduce the available data bandwidth to all (including benign) applications by more than 80% yet our touchtone eavesdropping classifier maintains accuracy over 95% for three of the four tested phones. Other designs that are more informed by our analysis of the vulnerability, 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%.

To summarize, the main contributions of this work include:

- The investigation of the new security threat of touchtone eavesdropping using smartphone motion sensors. Our discovery completes the picture of the security analysis on motion sensor-enabled side channel attacks.
- Characterizations of the root causes, factors, and limits of touchtone eavesdropping. We demonstrate the relevant physics and signal processing theory of touchtone leakage to reveal challenges that mitigations must consider.
- Design and analyses of possible mitigations against touchtone eavesdropping. We explicitly include the idea of maintaining benign application's functionality as a design goal. Our analyses also inform future researchers of defense methodologies against other types of motion sensor eavesdropping attacks.
- Implementations and evaluations of the attack and mitigation mechanisms for touchtone eavesdropping. Our machine learning classifier achieves over 99% accuracy on recognizing 12 touchtones. Our mitigation of anti-aliasing filter reduces accuracy by over 50.1% with no information loss.

2 BACKGROUND AND RELATED WORK

2.1 Touchtones

Touchtones, also known as dual-tone multi-frequency (DTMF) signals, are a standardized code of two-tone audible acoustic signals that play upon a numerical key press [40]. Touchtones are often used in telecommunications and various other applications with a numerical touchpad [8, 19]. The sound is produced by a phone when users 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, etc. There are 12 unique touchtones commonly used by smartphones (Figure 2), each consisting of two frequencies taken from two separate frequency sets, used for the numbers 0-9, the symbols * and #. As they are unique, hearing one touchtone is indicative of a certain number press. These dual-tone combinations have been chosen specifically to be easily understood in the presence of noise for reliable communication.

2.2 Relevant Signal Processing Concepts

2.2.1 Aliasing. Aliasing can have several definitions depending on the context, but the most relevant definition in the context of this paper refers to distortions caused by the improper sampling of a signal [22, 38]. As defined by the Nyquist sampling theorem [24, 33] the highest frequency a sensor with sampling rate f_s can properly

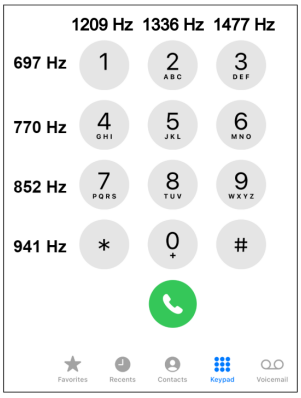

Figure 2: Touchtone frequencies. Touchtones are comprised of two single-frequency tones emitted simultaneously to convey numerical input.

sample is the Nyquist frequency $f_N = f_s/2$. If a signal has frequencies greater than f_N the sensor output *will* contain aliases of the original signal. The formula for the frequency of the alias, f_a, given the Nyquist frequency f_N and the frequency of the original signal f is $f_a = |2mf_N - f|$.

2.2.2 Sensor Data Bandwidth. The signal path of physical signals can normally only allow signals in a certain range of frequency to pass without strong attenuation because of the signal path's frequency response which is determined by its physical properties. Generally, the difference between the upper and lower bounds of such frequency range is defined as the bandwidth of the signal path. For an ideal motion sensor (with a flat frequency response) with the sampling rate of f_s, its bandwidth is the same as the Nyquist frequency f_N since only motion signals under the Nyquist frequency can be properly sampled and passed to the processor of smartphones. Although the nominal bandwidth is pre-determined by the properties of the signal path, people usually reduce the actual bandwidth by means of filtering.

2.2.3 Filtering. Filtering is the process of reducing the bandwidth of a signal path by blocking (attenuating) unwanted signals at certain frequencies and only allowing the desired signal to reach the destination, i.e., the smartphone processor in the context of this paper [43]. Common filters include low-pass filters, which block high-frequency signals and pass through high-frequency signals, and high-pass filters which achieve the opposite. The band of frequencies that the filters let pass through is called the pass band. Filters can be implemented either in software or in hardware and can be implemented in different forms such as simple RC impulse response filters, Butterworth filters, Chebyshev filters, etc. Different implementations of filters have different frequency responses, meaning the abilities of blocking and passing signals at different frequencies are different. Ideally, the frequency response in the filters' passband should be flat, so that the desired signals won't be distorted. However, such distortions are usually unavoidable for both software and hardware filters in the real world.

2.3 Related Work

Side-channel Acoustic Eavesdropping Using Motion Sensors. Previous works have shown the feasibility of acoustic eavesdropping attacks using motion sensors similar to touchtone eavesdropping in this paper. Gyrophone [27] demonstrates an attack that recognizes spoken digits produced by electronic speakers using smartphone gyroscopes by extracting speech spectral information. Similarly, Spearphone [1], AccelEve [2], AccEar [14] use smartphone accelerometers to eavesdrop on spoken digits. AccelWord [45] investigates the feasibility of leveraging smartphone's accelerometer to capture acoustic signals for low-power hotword detection. PitchIn [12] fuses across multiple uni-model sensors (e.g., only accelerometers or gyroscopes) to reconstruct intelligible human speech by interleaving sensor readings from multiple sensors to increase the effective sampling rate. Our work differs from the previous works in that: 1) We assess the security issue of eavesdropping touchtone information from smartphones, which requires a different analysis methodology than the previous acoustic eavesdropping targets. 2) We analyze and evaluate the effectiveness as well as the feasibility of several mitigations that can be implemented, and open up the discussion of future functionality-aware mitigations. 3) We inspect how acoustic leakage can manifest itself differently in separate axes' sensor readings of even a single sensor and uncover the fact that an advanced attacker may combine multidimensional information from different axes.

Other motion sensor side-channels. Recent research also demonstrates novel side-channel attacks utilizing smartphone motion sensors to infer victims' locations or keystrokes. ACComplice [13] leverages the smartphone accelerometer to infer the victim driver's driving routes as well as starting point. Narain et al. further extended the findings of ACComplice and demonstrated the feasibility of such attacks on a large scale across ten cities [29]. (sp)Iphone [25] accesses accelerometer readings to infer typed text on nearby keyboards by observing the relative physical position and distance between the smartphone and keyboards and the vibration detected. Similarly, ACCessory [30] utilizes an accelerometer to infer keystrokes as the victim user types on his/her smartphone. Due to minute differences in taps, it is able to sufficiently infer the typed keys. Tapprints [28] 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. These motion sensor side-channel attacks against locations and keyboard inputs complement our work of eavesdropping touchtone information.

Acoustic injection attacks. Previous work has also explored motion sensors' susceptibility to vibrations caused by acoustic signals, namely to affect motion sensor readings via acoustic injection. For instance, Son et al. propose an attack to bring down and crash drones only by acoustic injection as MEMS gyroscopes are vulnerable to acoustic noises at their resonant frequencies [34]. Similarly, [37, 39] propose acoustic injection attack on MEMS accelerometers to manipulate the output of the sensors by injecting certain acoustic signals at their resonant frequencies. Unlike these attacks that inject acoustic signals into motion sensors, we demonstrate the feasibility of capturing privacy-sensitive information such as touchtone from acoustic signals naturally emitted from victim smartphones.

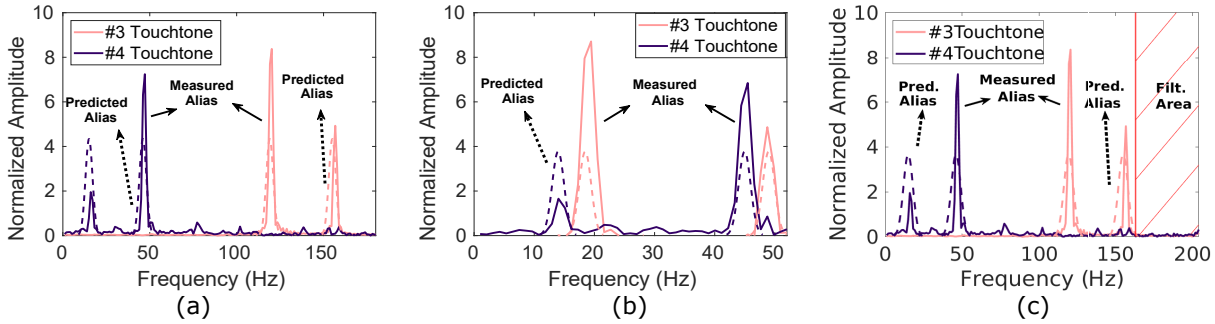

Figure 3: Predictable and discernible touchtone leakage. Touchtone leakage for #3 and #4 touchtones in a Google Pixel 2's accelerometer's x-axis. These signals remain discernable and predictable in the frequency domain with (a) a normal, unaltered signal, and also despite apparent mitigations suggested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

3 TOUCHTONE EAVESDROPPING CHARACTERIZATION

This section analyzes the information leakage threats posed by touchtone eavesdropping using smartphone motion sensors (Fig 1) and the multitude of reasons why it can be difficult to mitigate. We investigate how touchtones produced by a phone's speaker leak distinguishable, deterministic side-channel signals into the smartphone's accelerometer and gyroscope sensor readings.

3.1 Threat Model

This paper considers an adversary whose goal is to determine a user's numerical key presses 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 running in the background with motion sensor access, as has been assumed in all previous works of acoustic eavesdropping using smartphone motion sensors(Section 2.3). Note that popular smartphone platforms such as Android do not require applications to ask for user permission to use motion sensors. As a result, any applications running in the background can potentially collect motion sensor data stealthily for malicious purposes.

We also assume the adversary has access to the same model as the victim's phone(s); a phone's model can be determined by an application using fingerprinting techniques [4, 32, 46]. 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 as the victim data can be saved and sensitive information (e.g. credit card numbers, bank pins, social security numbers) may not change often.

3.2 Touchtone Signals in Motion Sensor Data

Touchtone leakage manifests itself in motion sensor data in a multitude of forms due to various physical and signal processing phenomena. Each of these manifestations can contain complementary information to the original touchtone. This section investigates why adversaries can infer touchtones of user inputs from motion sensor readings.

3.2.1 Acoustic Waves and Sensor Construction. Acoustic waves produced by the smartphone's speaker alter the output of microelectricalmechanical systems (MEMS) accelerometers and gyroscopes [3] due to how these sensors *approximate* motion. MEMS accelerometers and gyroscopes approximate the motion of a larger body (i.e. a smartphone) 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. For example, small acoustic vibrations via the air or contacted surfaces can move the small sensing masses even if minimally affecting the connected body (i.e. smartphone) due to effects such as varying frequency responses [1, 27, 37]. In this case, MEMS accelerometers and gyroscopes may capture acoustic signals.

3.2.2 Touchtone Aliasing. Aliasing, described in Section 2.2, is a key factor in both making touchtone leakage occur and making it difficult to mitigate. Touchtones have frequencies higher than the Nyquist sampling rate for most smartphone motion sensors (lower than 250 Hz), and thus have aliases. However, the frequencies of these aliases can be predicted as the touchtone frequency and sampling rate are both known (Fig 3). An attacker can use these known aliases to indicate the presence of the missing original touchtone frequencies. Furthermore, the placement of these aliases — how all touchtone frequencies can lie somewhere in the sampled signal's frequency band — can make touchtone eavesdropping resistant to certain apparent mitigations. For example, reducing the sampling rate will not get rid of aliases, but only move them to other deterministic frequencies (Fig 3b). Furthermore, low-pass filters may remain ineffective unless the cutoff frequency is placed low, as touchtone aliases could be close to 0 Hz (Fig 3c).

3.2.3 Leakage Signal Manifestations. The two above factors enable touchtone leakage, but information leakage may manifest in a multitude of forms simultaneously (e.g., different axes of a sensor's readings having different signals caused by touchtones) due to a

variety of physical and signal processing phenomena; these manifestations can provide complementary and distinct information for the purpose of classifying touchtones (and thereby user input) and an attacker may need only one of these manifestations in some cases to determine the touchtones.

First, different types of sensors or different axes of a single sensor can contain complementary or different information about the same set of touchtones. One factor that can affect how information manifests is the varying frequency responses in phone mechanical construction, speakers, sensors, or even individual axes of sensors. Different frequency responses inherent to physical materials and sensors can lead to one sensor axis having a higher signal-to-noise ratio (SNR) for certain frequencies (i.e. touchtones) where a separate axis could have a higher SNR for other frequencies [5, 10, 34], as shown in Fig 4a. With access to readings of all sensor axes, an adversary may be able to exploit this fact and combine useful information.

Additionally, even in the same sensor axis, information about the same touchtone can manifest in different manners. 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 as shown in Fig 4b. 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.

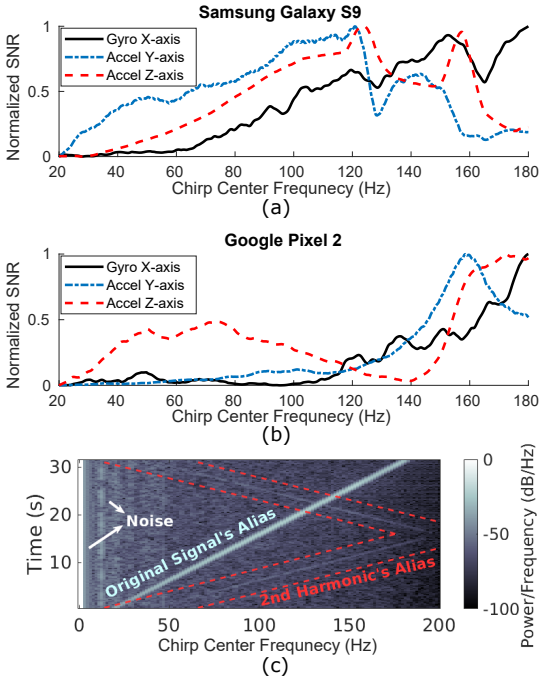

Figure 4: Touchtone information manifestations. Touchtone information can be manifested in a variety of forms or to varying extents in motion sensor data. In (a) and (b), two axes have distinct non-linear frequency responses to a 420 Hz to 580 Hz chirp from the speaker of smartphones. Different axes may thus be better predictors for certain tones. (c) shows how there may be many subtle artifacts in touchtone data. An attacker could use any of these artifacts to perform touchtone eavesdropping.

3.3 Adversarial Touchtone Recovery

The goal of the attacker is to recognize which touchtone is pressed by discerning the presence of the seven individual touchtone frequencies (Fig 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 3.2.3. The most straightforward approach to do this is by making a machine learning-based classifier with all accessible sensor data as the classifier's inputs. The advent of easily usable machine learning tools makes this task not arduous 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) and multiple axes. 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 [20], body-sensor networks [11], and much more. Building a classification model to specifically use this fact should lead to more efficient attacks.

Summary: Challenges for Designing Mitigations. From the analysis above, we know an attacker may only need one manifestation of touchtone information to achieve an eavesdropping attack, depending on the amount of information the attacker wants to recover. Defenders, however, must consider how to block as much of this information as possible while allowing benign applications to use sensor readings.

4 FUNCTIONALITY-AWARE SOFTWARE MITIGATION

Touchtone eavesdropping mitigations require careful forethought and consideration of leakage mechanics to effectively reduce leakage while not harming benign application behavior. To accomplish this task, mitigations should reduce touchtone information in motion sensor data while minimally altering or reducing any other information. Our work further considers additional criteria 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.

4.1 Designing for Both Security and Functionality

While protecting the security of smartphone users from touchtone eavesdropping attacks is an urgent issue this paper is addressing, we consider ensuring functionality to be a second—but no less critical—

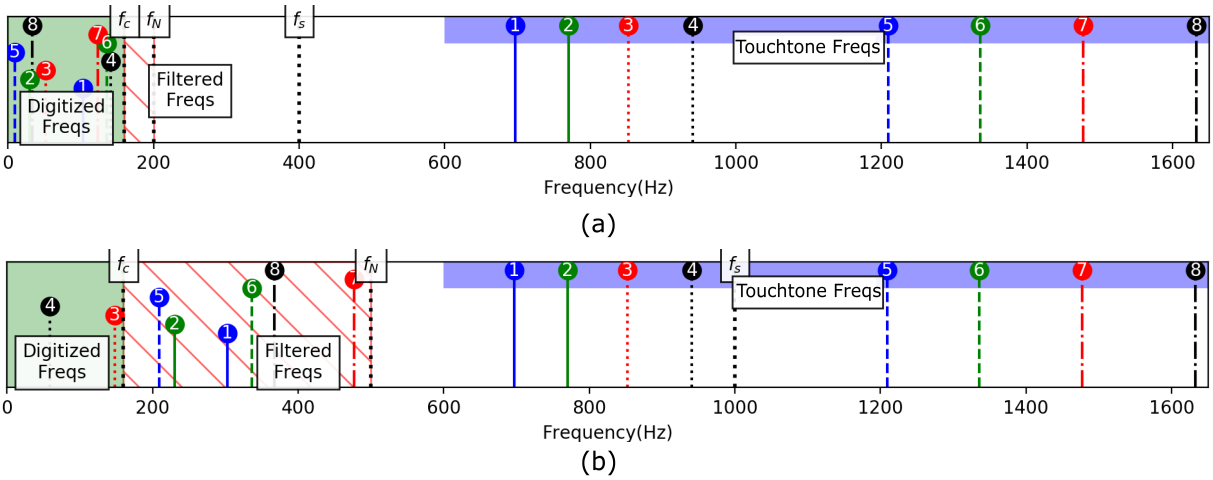

Figure 5: A need for oversampling. Digital anti-aliasing filters can attenuate more touchtone aliases than low-pass filters without reducing available bandwidth due to the use of oversampling. In the example with sampling rates of $f_S = 400Hz$, $f_N = 200\ Hz$, and $f_c = 180\ Hz$, touchtone aliases (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 4.2.2). (b) A digital anti-aliasing filter with the same f_c can filter more frequencies due to the use of oversampling (Section 4.3.1).

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 dependant applications. It is widely accepted that security must support some level of functionality and usability [26, 42, 44] 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. From a practical use standpoint, designing such mitigation for security protection thus also 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 near-unanimous trend of higher sensor bandwidth leading to higher performance in previous research in various activities such as human activity recognition [18], animal health monitoring, [31], road quality assessment [6], etc. In addition, more commercialized techniques such as using motion sensor readings for smartphone image stabilization [15] and rolling shutter correction [17] 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. An ideal mitigation should support the original bandwidth with minimal distortion, only removing traces of touchtone byproducts.

4.2 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.

4.2.1 Sampling Rate Reduction. Lowering sampling rates of sensors directly reduces available bandwidth (Section 2.2) in an attempt to also lessen the threat of acoustic eavesdropping; however, it is ineffective at attenuating leakage (Fig 3b) primarily due to how aliasing places touchtone information in a digital signal no matter the sampling rate (Section 3.2.2). Even at very low frequencies, the eight touchtone frequencies still have aliases and thus leave discernible traces. Although with such an extremely low sampling rate, it might be difficult for an attacker to realistically differentiate between different touchtone aliases. But this also affects dependent application's functionality similarly. Our experiments back this intuition (Section 6.2.1), as reduced sampling rates achieve minimal accuracy reduction for our touchtone eavesdropping attack until having sampling rates under 100 Hz, $\frac{1}{4}$ of the original sample rate.

4.2.2 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 security while preserving functionality (Fig 3c). To note, previous

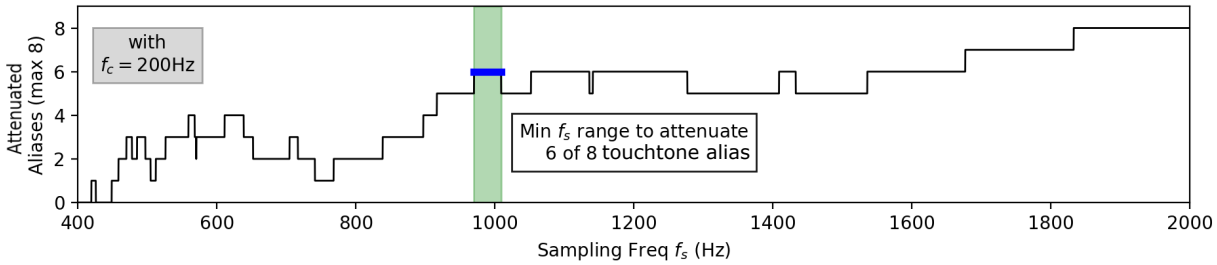

Figure 6: Choosing an optimum sampling rate to mitigate touchtone leakage. A mitigation designer desiring to attenuate the most touchtone aliases using the lowest sampling rate f_S when given a bandwidth f_c to support can make use of the non-linear but predictable nature of aliased frequencies. As the oversampled rate increases, the number of aliased frequencies above f_c will change. The designer can calculate this number of attenuated aliases and then select the minimal sufficient sampling rate to meet additional design constraints such as power consumption.

papers often do not 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 3.2.2, 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 security. Our experiments demonstrate this pathology (Section 6.2.2).

4.3 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 defending against acoustic leakage. Additionally, we describe how one can utilize the predictable nature of touchtone aliases in defense design.

4.3.1 Oversampling and Digital Anti-aliasing Filters. Oversampling is the act of sampling at a faster rate than the bandwidth that one wishes to eventually provide. Oversampling can be used to create anti-aliasing filters that reduce touchtone leakage while still providing the original bandwidth to current applications (Fig 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 operations on the oversampled signal and then downsample the signal to the original sampling frequency. If the oversampled frequency is a multiple of the original, this can be trivially done by selecting x of y number of samples from the oversampled data. If the oversampling frequency is a non-multiple, this could result in very small distortions being introduced 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 5. 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.

4.3.2 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 sampling frequency to maximize the number of targeted sensitive frequencies above f_N, as demonstrated in Fig 6. The basis for this lies in the non-linear relationship between signal frequency, sampling frequency, and the frequency alias as seen in Section 2.2. One can set the filter cutoff frequency f_c to design for a 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.

Table 1: Motion Sensor Information for Tested Phones.

Phone Model	IMU Model	Sampling Rate (Hz)	
		Reported	Measured
Google Pixel 1	BMI160	400.00	401.69
Google Pixel 2	LSM6DSM	400.00	409.96
Samsung Galaxy S8	LSM6DSL	400.00	429.27
Samsung Galaxy S9	LSM6DSL	415.97	413.61

Reported inertial measurement unit (IMU) model, which contains both an accelerometer and gyroscope, and sampling rates.

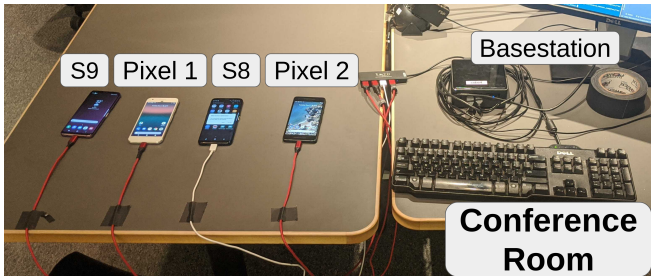

Figure 7: Data collection setup in a conference room.

5 EXPERIMENTAL SETUP AND METHOD

To evaluate how much touchtone information can adversaries recover from motion sensor readings and the effectiveness of different mitigations, we collected touchtone samples on multiple phones with and without mitigations in place. We then designed machine learning classifiers for classifying a test set of sensor recordings to provide accuracy numbers. Our machine learning classifier uses a variety of time and frequency features along with selective axis integration to evaluate a more advanced adversary.

5.1 Data Collection

5.1.1 Setup. We have three different hardware setups for motion sensor data collection. The first two setups collect data from the four Android phones listed in Table 1 for baseline (no mitigation) 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[36]. Each setup used an Intel NUC running Ubuntu 18.04 [16] as a base station, smart-phones (Table 1), cables, and base station peripherals on a table (Figure 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 [9] to upload a custom Android data collection program to each phone and for other communications.

The third hardware setup collects data at faster sampling rates for software anti-aliasing filters and for testing onboard hardware anti-aliasing filtering. Phone hardware can collect at rates faster than what is made available to applications in smartphones to limit power consumption. Although current smartphone software API does not support it, we test it with external sensors to emulate possible future mitigation. To that end, our setup uses an LSM9DS1 breakout board, a very similar chip to the ones in three of the phones (Table 1), a Teensy 3.6 micro-controller, 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.

5.1.2 Sensor Data Recording. To reduce temporally correlated biases from data collection 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 phone's 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.

For each individual setup, we recorded the motion sensor data of the 12 touchtones in Figure 2. Each individual dial-tone sample was played for 0.5 s, with each tone being recorded 250 times per setting for a total of 3000 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 12 touchtones).

5.2 Touchtone Classifier

To serve as an evaluation metric we made a machine learning classifer (Figure 8) to mimic that of an advanced adversary.

5.2.1 Selective Integration of Sensor Data. To emulate a more advanced adversary, we build classifiers that selectively combines 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 3.3). Previous work has demonstrated classifiers for acoustic leakage onto motion sensor [1, 27, 45], 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 empirically 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 the 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 Figure 8.

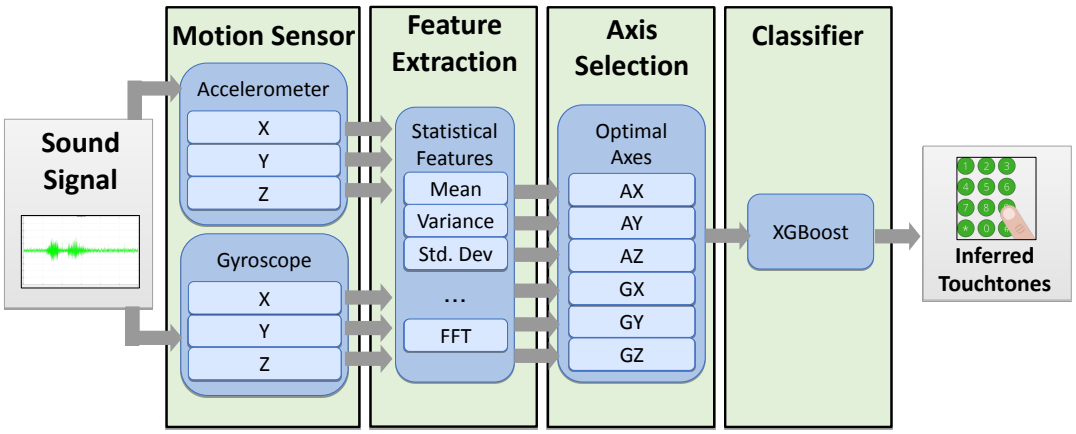

Figure 8: Eavesdropping classifier. Our system extract signal features and selectively integrate useful motion sensor data from multiple sensors and axes to better classify touchtones.

Table 2: List of Statistical Features Used in Classification.

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

The signal would be split into windows where the above features were calculated.

5.2.2 Features and Classifier Design. We briefly detail the feature extraction and classifier of our touchtone classifier in this section. As a reminder, features are calculated per sensor axis, then features of only the optimal combination of axes are included in the model as described in Section 5.2.1.

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 (i.e. window 1 in one signal correlates with window 1 of another signal).

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, as shown in Table 2, are very similar to those used in previous work [1].

Zero-padding: Feature vectors with a different number of time windows, which may happen due to experimental error, must have the same number of features for the classifier to compare properly. The system zero-pads each feature vector to ensure 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 [7].

5.2.3 Implementation Details. 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 individual axis. These axes would then be ranked by individual accuracy performance and 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 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 3 and 4 respectively with the best, selected parameters shown in bold. We tested these settings against a commonly used feature set for audio classification with Mel Frequency Cepstral Coefficients (MFCCs) [27] and another common classifier with Random Forest [1] to provide a comparison against other commonly used selections. We found that the XGBoost model with the statistic features constantly outperforms the other classifier-feature combinations. We took the highest accuracy result to select feature and classifier settings. These settings stayed the same through all testing.

Specifically, we used the statistical features in Table 3, which are calculated with a window size of 50 sensor reading samples and a

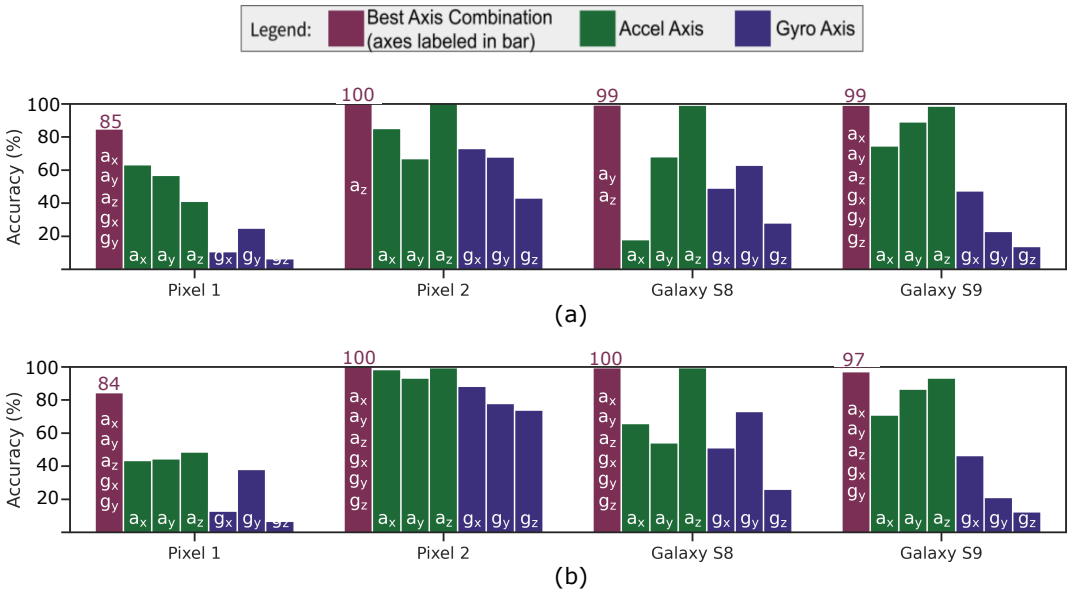

Figure 9: Baseline results for the touchtone eavesdropper without any mitigation. (a) Conference room and (b) Server room hardware setups. For each phone, we show the accuracy of classification models trained on individual axes alone, then show the accuracy for the model trained on the optimal combination of axes.

step of 5 samples. For the XGBoost classifier, it uses a learning rate of 0.2, a max depth of 5, a min child weight of 3, a gamma value of 0.1, and a colsample_bytree value of 0.5. Based on the assumption that the adversary knows the model of the victim's phone and can acquire a duplicate device in advance, we train the classifiers on the data collected with the same phone as the test data to evaluate the upper limits of the recognition accuracies. We will further discuss the possibility of cross-device training and testing in Section 7.3. On average, it takes less than 0.02 seconds to classify an eavesdropped touchtone.

5.3 Signal Processing Mitigations

5.3.1 Selection. We selected a total of four signal processing mitigation designs to evaluate. Two designs, a software-only low pass filter and reduced sampling rates, were chosen as they are briefly mentioned in previous papers as possible mitigations for related work and may seem like natural mitigations for touchtone leakage. However, our analysis (Section 4.2) predicts that both mitigations will 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 OS-governed oversampling combined with filtering as shown in Figure 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.

The hardware digital anti-aliasing filter refers to the onboard anti-aliasing filter provided by the LSD9DS1 sensor, 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 over the software version in that it requires no computational burden and should require 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 the sensor driver.

5.3.2 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 the effect of reducing sensor sample rate by n. 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

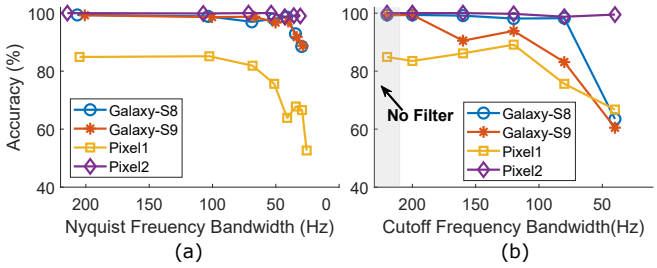

Figure 10: Functionality-unaware mitigation results. (a) Results for the down-sampling mitigation with listed bandwidth equivalent to half the sensor sampling rate. (b) Results for the low-pass filter mitigation with listed bandwidth equivalent to the cutoff frequency used. Both mitigations do not greatly reduce touchtone eavesdropping accuracy until bandwidth is under 50 Hz, which could hinder the functionality of benign applications using motion sensors.

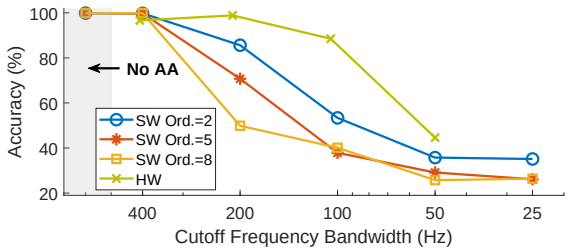

Figure 11: Anti-aliasing filtering results. While not completely eliminating the attack, the software anti-aliasing filter is able to significantly reduce the eavesdropping accuracy.

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 final desired bandwidth. We vary this desired bandwidth to use as a comparison against other mitigations. The filtered signal is then down-sampled (similarly to the reduced sampling rate mitigation) to the desired bandwidth. We also vary filter order in this evaluation.

(4) **Hardware (digital) 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 is then processed by the classifier.

6 EVALUATION RESULTS AND ANALYSIS

In this section, we report the attack and mitigation results with the setups described in Section 5. We analyze and summarize the findings of our assessment of the eavesdropping attack and different mitigations. Software low-pass filtering and 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 are able to significantly mitigate the threat while also preserving more data bandwidth.

6.1 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 Figure 9, demonstrating that malicious applications can effectively recover user input.

6.1.1 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%. As shown in Table 1, we notice that Pixel 1's IMU is produced by a different manufacturer than the other three phones. This result demonstrates that factors other than sampling rates can vary recognition rates. These factors could include signal propagation path that attenuates the acoustic signal, less sensitive sensors, different frequency responses, or different sensor configurations and MEMS structures. This result also suggests that some motion sensors may be more resistant to touchtone leakage than others. We believe a dedicated future study examining which motion sensors are less susceptible could provide insight into future hardware-based mitigations.

6.1.2 Accelerometer vs. Gyroscope Axis Accuracies. Classification based on data from an accelerometer axis achieved higher average accuracy than 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 vibrate and these vibrations are quantized. Thus, the intent of each sensor changes the effectiveness of this particular scenario.

6.1.3 Selective Combination of Sensor Axes. The selective combination of axis data achieved significantly higher results for one phone model, the Google Pixel 1, and improved accuracy versus a single axis for all but one case. This exception 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.

6.2 Mitigation Strategy Evaluations

6.2.1 Reduced Sampling Rates. The results for the reduced sampling rates mitigation verify our theoretical analysis in Section 4.2.1 to show that this approach does not greatly affect touchtone eavesdropping until sampling rates are reduced significantly (Figure 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.

6.2.2 Software Low-pass Filter. An evaluation of software low-pass filter show that, as expected (Section 4.2.2), they may also not seem to affect accuracy significantly until lower bandwidths are reached. As Figure 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 actually 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 for touchtones remained essentially unaffected even at a 40 Hz cutoff frequency.

6.2.3 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. In both cases, the oversampling rate is 952 Hz. 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 Figure 11), while neither the pure low-pass filter nor the reducing sampling rate mitigation could reduce the accuracy to below 80%. Even with the same bandwidth as the original smartphone sensors (around 200 Hz), we observe over 50% decrease in the accuracy with an order 8 software filter.

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 actually 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.

7 DISCUSSION

In this section, we discuss the limitations of this work and possible future works.

7.1 Hardware Solutions

In this paper, we do not evaluate sensor hardware changes for mitigations as they cannot be implemented as a software update. However, some mitigations embedded into circuitry could serve as long-term solutions. Analog filtering mitigation 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 be used to mitigate some of the aliasing effects.

7.2 Application to Other Acoustic Leakage

The analysis of the paper focuses on touchstones, but it is largely applicable to other forms of acoustic leakage. We 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. Thus, mitigation strategies that work for touchtones should largely work for speech. Specifically, we conducted a preliminary experiment on speech eavesdropping mitigation by reusing the third setup in Section 5 and collecting speech audio signals from the TIDIGITS corpus [21], which is used by previous speech eavesdropping research [1, 27]. There are 11 digits, 0-9 with two separate ways to say zero, "zero" and "oh". There are 225 speakers who say each digit twice. This results in 4950 audio samples from 225 speakers. We then apply the same features, classifiers, and mitigations to the speech samples. We observe a user-independent digit recognition accuracy of 59% without any mitigation. The accuracy decreases to 35% and 24% when the order 8 software anti-aliasing filter uses a bandwidth of 200 and 100 Hz respectively. We believe dedicated future studies can further explore the effectiveness of the mitigation on other types of acoustic and motion signals.

7.3 Cross-device Attack & Device Coverage

This work assumes a threat model of a known victim smartphone model. This allows the adversary to collect training data with the same phone model to achieve the best eavesdropping performance. As pointed out in Section 3.2.3 because different phone models can employ different models of motion sensors and phone mechanical structures that could change the characteristics of the signals. This assumption is also made by most previous speech audio eavesdropping works. Nevertheless, we envision that cross-device attacks that use training data collected with different phone models than the victim's phone could further advance the attack in extreme cases where the victim's phone model is completely classified. To that end, we tested our trained models on different phones. On average, we observe that the recognition accuracy decreases from over 90% to about 50%, showing that the eavesdropping design provides a certain level of generalizability to device variations. We note that these results may be significantly increased by redesigning the classifier and features to be more resilient to cross-device diversity. As an example, a dedicated previous work [35] explored how to reduce the dependency of motion sensor speech eavesdropping attacks on the training devices. They demonstrate that by only

selecting features that perform better across different devices, it is possible to increase the recognition accuracies on unseen devices by more than 30% at the cost of a slight decrease in the single-device recognition accuracy. We believe future research can use a similar methodology to improve the cross-device attack performance of touchtone eavesdropping.

One major limitation of this work is the relatively small coverage of the tested phone models. In addition, the four phones are all released before 2018. The older phone models allow us to acquire their motion sensor information and find similar sensor breakout boards to test anti-aliasing filter mitigations but have more different phone constructs than the latest smartphones. It is worth noting that newer phones are very likely to see higher touchtone eavesdropping accuracies because of the higher sensitivity and sampling rates of the new motion sensor modules and the more powerful speakers used by these phones. This has been shown by recent research works [2, 23]. The only factor of newer phones that could potentially reduce the eavesdropping performance is the use of different phone constructs, e.g., those that intentionally reduce the mechanical coupling between the phones' speakers and motion sensors by providing isolation or dampening. To the best of our knowledge, however, no smartphone manufacturers have made such improvements to mitigate the motion sensor eavesdropping risks. We believe a re-evaluation of these attacks needs to be carried out should such design improvements be implemented in future phone models.

8 CONCLUSION

This paper investigated the threat of smartphone motion sensor-based acoustic eavesdropping against a new target—touchtones, and examined how to mitigate such eavesdropping attacks by malicious applications while preserving the functionality of benign applications. We showed that adversaries may achieve over 99% accuracy in inferring touchtone inputs. Some of the more obvious mitigations, such as software low-pass filters or reduced sampling rates, could actually have very little effect on mitigating touchtone leakage. We instead investigated software and hardware digital anti-aliasing filtering designs which achieve moderate success and can be implemented as a software update. We intend for this work to motivate the need for deployable mitigations against acoustic leakage on smartphone motion sensors, including but not limited to touchtones, while also providing a basis for future mitigations to improve upon.

ACKNOWLEDGMENTS

This work is funded in part by a gift from Analog Devices, Inc. We thank our reviewers for their constructive feedback that helped improve this paper.

REFERENCES

[1] S Abhishek Anand, Chen Wang, Jian Liu, Nitesh Saxena, and Yingying Chen. 2019. Spearphone: A Speech Privacy Exploit via Accelerometer-Sensed Reverberations from Smartphone Loudspeakers. *arXiv preprint arXiv:1907.05972* (2019).

[2] Zhongjie Ba, Tianhang Zheng, Xinyu Zhang, Zhan Qin, Baochun Li, Xue Liu, and Kui Ren. 2020. Learning-based Practical Smartphone Eavesdropping with Built-in Accelerometer.. In *NDSS*.

[3] Stephen Beeby, Graham Ensel, Neil M White, and Michael Kraft. 2004. *MEMS mechanical sensors*. Artech House.

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

[5] Connor Bolton, Sara Rampazzi, Chaohao Li, Andrew Kwong, Wenyuan Xu, and Kevin Fu. 2018. Blue Note: How intentional acoustic interference damages availability and integrity in hard disk drives and operating systems. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1048–1062.

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

[7] Tianqi Chen and Carlos Guestrin. 2016. XGBoost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Aug 2016). https://doi.org/10.1145/2939672.2939785

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

[9] Android Developers. 2023. *Android Debug Bridge*. https://developer.android.com/studio/command-line/adb.

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

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

[12] Jun Han, Albert Jin Chung, and Patrick Tague. 2017. PitchIn: Eavesdropping via Intelligible Speech Reconstruction using Non-Acoustic Sensor Fusion. In *16th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*.

[13] Jun Han, Emmanuel Owusu, Le T. Nguyen, Adrian Perrig, and Joy Zhang. 2012. ACComplice: Location inference using accelerometers on smartphones. In *Communication Systems and Networks (COMSNETS)*. https://doi.org/10.1109/COMSNETS.2012.6151305

[14] Pengfei Hu, Hui Zhuang, Panneer Selvam Santhalingam, Riccardo Spolaor, Parth Pathak, Guoming Zhang, and Xiuzhen Cheng. 2022. Accear: Accelerometer acoustic eavesdropping with unconstrained vocabulary. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1757–1773.

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

[16] Intel. 2023. *Intel NUC*. https://www.intel.com/content/www/us/en/products/boards-kits/nuc.html.

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

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

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

[20] Seoungjun Lee, Dongsoo Har, and Dongsuk Kum. 2016. 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)*. IEEE, 84–89.

[21] R. Gary Leonard and George R. Doddington. 1993. TIDIGITS. https://catalog.ldc.upenn.edu/LDC93S10.

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

[23] Ahmed Tanvir Mahdad, Cong Shi, Zhengkun Ye, Tianming Zhao, Yan Wang, Yingying Chen, and Nitesh Saxena. 2022. EarSpy: Spying Caller Speech and Identity through Tiny Vibrations of Smartphone Ear Speakers. *arXiv preprint arXiv:2212.12151* (2022).

[24] Robert J. Marks, II. 1991. *Introduction to Shannon Sampling and Interpolation Theory*. Springer-Verlag, Berlin, Heidelberg.

[25] Philip Marquardt, Arunabh Verma, Henry Carter, and Patrick Traynor. 2011. (Sp)iPhone: Decoding Vibrations from Nearby Keyboards Using Mobile Phone Accelerometers. In *Conference on Computer and Communications Security (CCS)*. ACM, New York, NY, USA. https://doi.org/10.1145/2046707.2046771

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

[27] Yan Michalevsky, Dan Boneh, and Gabi Nakibly. 2014. Gyrophone: Recognizing Speech from Gyroscope Signals. In *23rd USENIX Security Symposium (USENIX Security 14)*. USENIX Association, San Diego, CA, 1053–1067. https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/michalevsky

[28] Emiliano Miluzzo, Alexander Varshavsky, Suhrid Balakrishnan, and Romit Roy Choudhury. 2012. Tapprints: Your Finger Taps Have Fingerprints. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services* (Low Wood Bay, Lake District, UK) *(MobiSys '12)*. ACM, New York, NY, USA, 323–336. https://doi.org/10.1145/2307636.2307666

[29] S. Narain, T. D. Vo-Huu, K. Block, and G. Noubir. 2016. Inferring User Routes and Locations Using Zero-Permission Mobile Sensors. In *2016 IEEE Symposium on Security and Privacy (SP)*. 397–413. https://doi.org/10.1109/SP.2016.31

[30] Emmanuel Owusu, Jun Han, Sauvik Das, Adrian Perrig, and Joy Zhang. 2012. ACCessory: Password Inference Using Accelerometers on Smartphones *(HotMobile)*. ACM, New York, NY, USA. https://doi.org/10.1145/2162081.2162095

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

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

[33] C.E. Shannon. 1949. Communication in the Presence of Noise. *Proceedings of the IRE* 37, 1 (Jan 1949), 10–21. https://doi.org/10.1109/JRPROC.1949.232969

[34] Yunmok Son, Hocheol Shin, Dongkwan Kim, Youngseok Park, Juhwan Noh, Kibum Choi, Jungwoo Choi, and Yongdae Kim. 2015. Rocking drones with intentional sound noise on gyroscopic sensors. In *24th USENIX Security Symposium*. 881–896.

[35] Weigao Su, Daibo Liu, Taiyuan Zhang, and Hongbo Jiang. 2021. Towards device independent eavesdropping on telephone conversations with built-in accelerometer. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5, 4 (2021), 1–29.

[36] General Tools. 2023. *DSM403SD*. https://generaltools.com/class-1-sound-level-meter-with-excel-formatted-data-logging-sd-card.

[37] Timothy Trippel, Ofir Weisse, Wenyuan Xu, Peter Honeyman, and Kevin Fu. 2017. WALNUT: Waging doubt on the integrity of MEMS accelerometers with acoustic injection attacks. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 3–18.

[38] Yannis Tsividis. 2004. 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*, Vol. 2. IEEE, ii–589.

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

[40] International Telecommunication Union. 1988. Technical Features of Push-Button Telephone Sets. *General Recommendations on Telephone Switching and Signalling* (25 11 1988). https://www.itu.int/rec/T-REC-Q.23-198811-I/en.

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

[42] Alma Whitten and J Doug Tygar. 1999. Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0.. In *USENIX security symposium*, Vol. 348. 169–184.

[43] Steve Winder. 2002. *Analog and digital filter design*. Elsevier.

[44] Chen Yan, Yan Long, Xiaoyu Ji, and Wenyuan Xu. 2019. 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*. 1215–1229.

[45] Li Zhang, Parth H. Pathak, Muchen Wu, Yixin Zhao, and Prasant Mohapatra. 2015. AccelWord: Energy Efficient Hotword Detection Through Accelerometer. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services* (Florence, Italy) *(MobiSys '15)*. ACM, New York, NY, USA, 301–315. https://doi.org/10.1145/2742647.2742658

[46] Yang Zhang, Peng Xia, Junzhou Luo, Zhen Ling, Benyuan Liu, and Xinwen Fu. 2012. Fingerprint attack against touch-enabled devices. In *Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices*. 57–68.

APPENDIX: TOUCHTONE INFERENCE MODEL INFORMATION

Table 3 shows all the feature parameters we tested in our randomized grid search. We observed better eavesdropping performance with the statistical features in Table 2 than the MFCC features that are commonly used by speech recognition applications. Table 4 shows all the hyper-parameters we tested. The XGBoost classifier outperforms the random forest classifier.

Table 3: Feature Settings.

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

The optimum feature settings used in the final model are in bold.

Table 4: Classifier Settings.

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	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

The optimum feature settings used in the final model are in bold.