An Autonomic and Permissionless Android Covert Channel

Kenneth Block
Northeastern University
Boston, MA 02115
block.k@husky.neu.edu

Sashank Narain
Northeastern University
Boston, MA 02115
sashank@ccs.neu.edu

Guevara Noubir
Northeastern University
Boston, MA 02115
noubir@ccs.neu.edu

ABSTRACT

Demand for mobile devices continues to experience worldwide growth. Within the U.S., there is a significant shift away from broadband usage towards Smartphones as the primary Internet entry point for consumers. Although technological advancements have helped fuel demand for greater features and functionality to enhance the user experience, they have also drawn attention from malicious actors seeking to access and exfiltrate increasingly available sensitive and content rich personalized information.

In traditional Android based exfiltration channels, the application engaged in information acquisition is granted permission to execute off-board communications. This tactic increases the possibility of detection by applications designed to identify this form of behavior. In this paper, we sever the acquisition / exfiltration bundling by assigning independent responsibilities to two apps communicating via a stealthy, permissionless, self-configuring and self-optimizing ultrasonic bridge. We present a framework for analyzing channel feasibility and performance, and apply it to 28 popular mobile devices. We demonstrate basic channel capability on 13 devices, achieving in certain cases, Bit Error Rates lower than 10^{-4} and Shannon capacity approaching 14 bps. We further demonstrate two performance boosting solutions that build on these results: a multichannel implementation which improves performance by nearly 80% and; a single channel Amplitude Shift Keying solution that increases capacity three-fold.

ACM Reference format:
Kenneth Block, Sashank Narain, and Guevara Noubir. 2017. An Autonomic and Permissionless Android Covert Channel. In *Proceedings of WiSec '17 , Boston, MA, USA, July 18-20, 2017,* 11 pages.
DOI: 10.1145/3098243.3098250

1 INTRODUCTION

In the U.S., broadband usage has slowed with the increased consumption of Smartphones [5]. Mobile device adoption is widespread with 3.9 billion smartphone subscriptions sold worldwide through November 2016 with projected sales exceeding 6 billion by 2020 [14]. Smartphone sales alone in Q2 2016 reached 343.3 million units [18]. In order to enhance the smartphone user experience, manufacturers frequently incorporate technological enhancements. Examples include increasingly accurate MEMs sensors, cameras and microphone arrays in addition to more powerful processors, enhanced communication radios and high capacity storage chips. These enhancements combined with user-friendly and incentive-driven App Stores, help fuel demand for greater functionality, including the control and storage of personal information. Unfortunately, this increases security concerns due to the appeal of acquiring such rich information for malicious purposes.

Google™ and Apple™ have tried to address these concerns by implementing multi-layered security architectures. For example, Android implements Application Sandboxing and a Permission based framework, enabling users to control and grant / deny access to sensitive resources. Security enhancements are regularly incorporated such as the introduction of permission groups (*normal* and *dangerous*) and runtime authorization for *dangerous* permissions (as of Android 6.0). Nevertheless, vulnerabilities are periodically discovered with malicious applications attempting to trick users by exploiting design and implementation flaws [4]. These exploits ([10, 26, 28]) continue to be difficult to detect.

This paper presents a stealthy covert channel built upon an ultrasonic communications bridge between two co-resident Android apps. The bridge, which uses the speaker and the local sensor package, leverages the smartphone's resonance behavior, where the resonance points are a function of the sensor design as integrated into the device's housing. Manufacturers typically design the operating band in the linear region, well below the resonance frequencies. However, if a stimulus generates frequency components in the non-linear region near a resonance point, sensor sensitivity increases and the accelerometer behaves as an amplifier / resonator.

There is a threefold benefit to the adversary. First, this is an out-of-band communications pathway. Second, the channel's high frequency speaker emissions operate well beyond the voice band, affecting audibility. Third, it's permissionless implementation greatly contributes to stealthy operations, 'hiding in plain sight' with data flowing freely as it circumvents system defenses.

Our contributions can be summarized as follows:

- To the best of our knowledge, we are the first to report the existence of a same-device, permissionless and ultrasonic, covert channel.
- Unique among Android covert channels, the transmitter monitors the received data which supports self-configuration, self-optimization, error correction and flow control.
- The channel is resilient to Android's non-uniform event reporting effects.
- We developed an automated framework to discover, identify and characterize the channel since it exists only in very narrow bands of the spectrum and is device unique.
- We applied the framework on a set of 28 mobile devices (spanning 18 different models) and established channels on 13 devices (4 unique models).

WiSec '17 , Boston, MA, USA
© 2017 ACM. 978-1-4503-5084-6/17/07. . . $15.00
DOI: 10.1145/3098243.3098250

- We evaluated channel capacity and throughput in three environments: a laboratory, the Xamarin TestCloud™ and the AWS Device Farm™ .
- We achieved device dependent theoretical Shannon capacity approaching 14 bps.
- Beyond our basic channel, we achieved 80% and 3× improvements with our multichannel and Amplitude Shift Keying enhancements.

The remainder of this paper is constructed as follows. In Section 2, we describe the exploitation opportunity. Section 3 details the system design and some of the practical limitations presented by Android devices for this attack type. Section 4 describes test environment details, device selection and measures of effectiveness. Results, including performance differences within identical device types, are noted in Section 5. In Section 6, we highlight measured performance and performance boosting techniques. Section 7 describes mitigation options and we end with a related works discussion followed by our conclusion in Section 8 and Section 9 respectively.

2 BACKGROUND AND MOTIVATION

Smartphone covert channels follow either *inter* or *intra*-device patterns. Channel endpoints in the former case reside in separate physical structures and face elaborate proximity challenges. For example, Do [11] used Frequency Shift Keying of ultrasonic waves to communicate with a pre-positioned, external *sink*. Farshteindiker [15] used a critically positioned, external ultrasonic device, affixed to the target Android device. In each case, position, external coupling and a participant unencumbered by Android limitations (i.e., permissions including microphone usage) were key.

With *intra*-device channels, the *source* and *sink* are co-located, exploiting same device resources for communication. Traditional tactics include setting manipulation, state modification, status manipulation and microphone based channels. These are becoming increasingly less feasible due to recent improvements in Android security. Deprecation of android.permission.ACCESS_SUPERUSER and the default enforcement mode configuration for SELinux have eliminated the granting of 'su' privileges, useful in accessing kernel data structures. Further, adoption of runtime and deploy time permission checks enables the user to determine the presence of inappropriate resource consumption or unrelated activities. As a result, system resource manipulation attacks such as the /proc attacks described by Marforio [23] now have limited effectiveness. Other attacks such as volume setting manipulation could be throttled with Operating System (OS) modifications that limit change rate, consistent with human behavior. Using the finger tap rate example, these channels would be limited to 7 changes per second [12], severely limiting performance.

Despite these aggressive security policies, attackers will continue to seek alternatives. If the objective remains to execute a stealthy same-device attack, then hiding from entities (including humans) that monitor permission requests, resource consumption or perform unusual shared system resource manipulation is vital. To date, Android covert channels circumvent at most, two of these detection mechanisms. The closest examples to encompassing all three are an air-gapped work demonstrated by Al-Haiqi [1] and Deshotels [9]

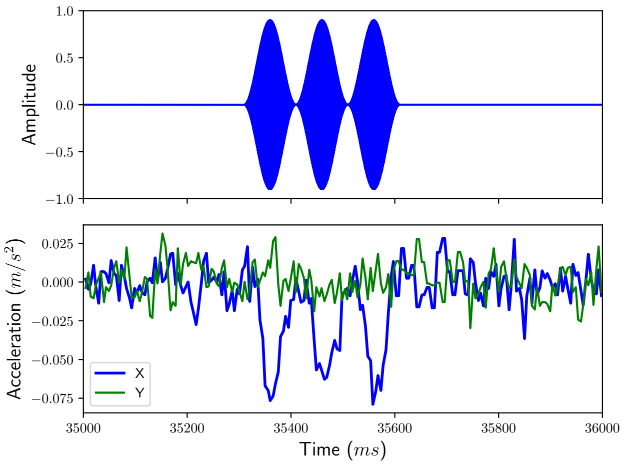


Figure 1: Sensor Response to 21050 Hz Tone Bursts on Samsung Galaxy S5

using the vibrator as the *source*. There are two shortcomings to this attack in that the vibrator requires permissions and, the vibration may be audible either directly or from device movement relative to the physical surface it rests upon, i.e., friction rub.

In the channel proposed herein, we address all three conditions. We extend the two app approach offered by Marforio [23], Laland [20] and Gasior [16] but in our context, the first app, isolated from network connectivity, accesses sensitive information while promising privacy. This provides the illusion of comfort to the user. Examples include calendars, contact lists, journals, password file managers and credit card managers. The channel can be established even with Do's [11] recommendation to ask for permissions for any resource request. Although applying this scheme to the device's speakers, which currently do not require permissions, audio access can be justified as the need for an alerting mechanism. The second app has sensor and network connectivity yet is blocked from direct access to sensitive information. Examples include games and fitness apps where sensor and network access are rationalized as functionally relevant.

This attack differs from Michalevsky [25], who reconstructed voice from sub-200 Hz signals and Zhang [38] who used the accelerometer to process speech, intending to detect 'hotwords' via energy pattern identification. In both cases, the energy from the voice content was near to or below the sensor Nyquist frequency. Although some aliasing exists, sufficient energy to yield results is present in the passband. In our channel, the frequency identification suffers aliasing effects since the operating band is ≈ 2 orders of magnitude above the Nyquist frequency.

Besides aliasing, High frequency signals face another impactful challenge, filtering. Typical MEMS devices will apply a Low Pass Filter (LPF) to limit high frequency artifacts in the sensor signal chain. A simple single order filter attenuates signals at 20 dB per decade resulting in 40 dB of roll off at our channel's operating frequencies (≈ 20 kHz). The amount of power needed to detect the signal becomes significant. This is supported by O'Reilly [31], who used MEMs accelerometers as guitar pickups which realized 40 dB attenuation at 20 kHz for each X, Y, Z axis above 1 kHz. Assuming that the pickups are exposed to an average stimulus of 60 dB of sound, near that of typical human voice, the power needed to achieve the in-band detection level is 100 dB. This is the equivalent audio power of standing next to a powered lawnmower. The 60 dB sensitivity level is consistent with Michalevsky [25], noting that the gyroscope could detect voice signals at 57 dB.

This paper's foundational concept is based on our discovery that under specific conditions, highly correlated responses are observable on the accelerometers when *high frequency* signals are Amplitude Modulated (AM) when emitted from a device's speaker(s). The signal is amplified [13] when operating near the accelerometer's mounted resonance frequency yielding a response emulating an *AM non-coherent detector*. We show in Figure 1, an example of this effect with a pattern of three, 21,050 Hz, AM modulated tones and the corresponding X axis accelerometer response. Significantly, this shows the envelope, meaning that the channel receiver *need not know the carrier frequency*, eliminating the need for a priori knowledge of the operating frequency.

3 SYSTEM DESIGN

3.1 Challenges

We faced four key challenges in developing this channel.

Stealth: This requires operating without needing permissions and avoiding resources that may be monitored or perform unusual operations.

Device and Environmental Diversity: Device diversity affects frequency identification, bit rate and channel orthogonality. The system response limits operation to small, *device specific* frequency bands with narrow coherence bandwidth, necessitating an autonomous solution to identify frequency and pulsewidth. Additionally, environmental conditions require signal extraction from noisy accelerometers.

Configuration: To avoid attribution and linkage, the channel endpoints must function without direct communication.

Android Limitations: The channel must operate despite sensor sampling and event reporting intervals that are orders of magnitude larger than the carrier frequency's period, violating the Nyquist rate. In addition, the Android system's sensor event reporting scheme is non-uniform.

3.2 Solution Overview

As mentioned, we assign data theft and off-board communications to two separate apps. These apps communicate with one another via an ultrasonic bridge using only two system shared and permissionless resources, the speaker and the sensors. Typical operation includes the bridge's *source* forming the high frequency wave and using the Android *MediaPlayer API* to control tone transmission through the local speakers. The receiver (the *sink*) uses the *SensorManager API* to monitor the resultant accelerometer perturbations prior to decoding and exfiltrating the data to an off-board third party using a cellular or Wi-Fi network (Figure 2). Post installation, the attack occurs in two stages: 1) Channel Identification (Phase 1) (exclusively a *source* activity) which addresses channel parametrization (e.g., carrier frequency, pulse width) and; 2) Data Transfer (Phase 2) (*source* and *sink* activity) which addresses the compromised data transfer. There are three precondition assumptions: 1) Both apps were willingly deployed by the victim; 2) The *source* has access to the speakers and; 3) The *source* has access to the compromised data.

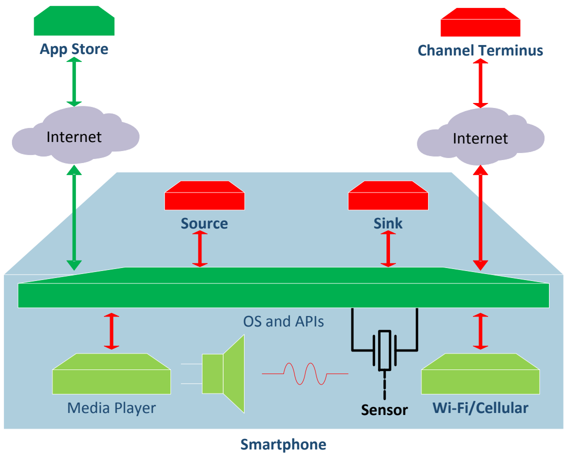


Figure 2: Single Device Covert Communications Channel System Design

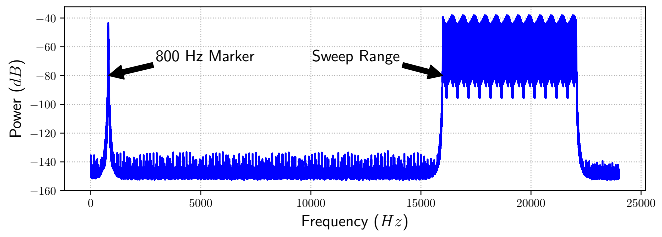


Figure 3: Spectral Components Illustration, Channel Identifcation Sweep Pattern

The technical challenge in operating ultrasonically is to address carrier detection when accelerometer packages typically have either cut-off or sampling frequencies below 2 kHz. As mentioned in Section 2, the carrier provides the energy stimulus to the sensor which follows the shape of the Amplitude Modulated envelope making receiver knowledge of the carrier irrelevant. It simply needs to know as shared secrets, minimal configuration information i.e., frame format and a small parameter set (i.e., a magic number).

The channel is quite unique. Unlike traditional channels, the transmitter may monitor the identical data observed by the receiver, thereby establishing a channel and message feedback loop. This supports self-characterization and eliminates handshaking and acknowledgements associated with reliable transmission. This mitigates device uniqueness (Dey [10], Das [8]) concerns and addresses the remaining Configuration and Device and Environmental Dissimilarity challenges.

3.3 Phase I: Channel Identification

The *source* synthesizes a frequency identification (FID) sweep pattern comprised of a set of discrete, coded frequency subpatterns using short interval spacing (i.e., 50 Hz and less). The frequencies range from 22,050 Hz to 16,000 Hz as shown in Figure 3 (see Section 3.5 for rationale). Note that an 800 Hz marker was included to support testing.

We adopt a spreading technique commonly used in wireless Direct Sequence Spread Spectrum (DSSS) applications. Using a Pseudo Noise (PN) sequence in the FID subpattern improves signal and clock recovery processing in the presence of interference. This technique is useful for signal extraction as the correlation result is strong when matched to sequence. The sequence (code) length depends on the spreading gain needed, per Equation (1).

$$Spreading\,Gain(dB) = 10 \times log_{10}(code\ length) \tag{1}$$

Using a Barker Code of length 11 (used in Wi-Fi DSSS) we can achieve $\approx 10dB$ of spreading gain. M-Sequences, for example, can be used for generating flexible length PN sequences [6], achieving (93.3 dB) with lengths in excess of $2,147,483,647$. In a real attack, the adversary can dynamically select the length based on accelerometer noise measurements and the needed bandwidth.

We generate the coded pattern based on bit and frequency values and modulate it using a Hanning window pattern. This window pattern was selected due to it's limited distortion relative to others such as Tukey and Flat top window functions albeit the latter two provide more energy. The resulting pattern is sent to the media player and played over the device's speaker(s). While transmitting, the *source* concurrently monitors the accelerometer's X, Y and Z axes responses and applies post-processing as follows:

- Generate a matched filter of temporal length equal to $PN\ sequence\ length \times pulsewidth$ with sample points derived from the event times.
- Correlate the sliding received signal data with the dynamically created matched filter. Correlation is computed for each encoded frequency sequence f to obtain a $score_f$ value using Equation (2), where l is the encoding scheme length, $x[i]$ is the sensor measurement at i within l, μ_x is the mean of all x over the sensor data length and $EE[i]$ is the encoding scheme's i^{th} code value within l, e.g., -1 or 1 for each chip as needed.

$$score_f = \left|\sum_{i=1}^{l} (|x[i]| - |\mu_x|) \cdot EE[i]\right| \tag{2}$$

- Identify the frequency, $\hat{f}$, yielding the maximum correlation magnitude using Equation (3). $\hat{f}$ is coincident with the best signal-to-noise ratio.

$$\hat{f} = \underset{f \in \{f_1, f_2, f_3, \ldots f_n\}}{\text{argmax}} \quad score_f \tag{3}$$

Correlation and Synchronization

The matched filter window is dynamically generated by the app(s) as it slides over the sensor reporting events. Matched filters are ideal signal representations, commonly used to extract the original signal in noisy environments. This contrasts with traditional filters which may degrade the original signal during noise removal. Here, the sample count is based on the number of event reports within the sliding window while the values are determined from the time positions relative to an ideal, uniformly spaced time based sample set. As illustrated for a Samsung Galaxy S5 in Figure 4, non-uniform sampling rates are present, affecting the matched filter sample points over the coded window. Options included processing without correction or synthesizing evenly spaced events. However,

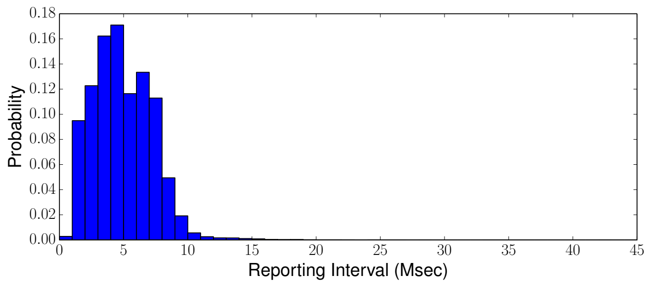


Figure 4: Sensor Event Time Histogram

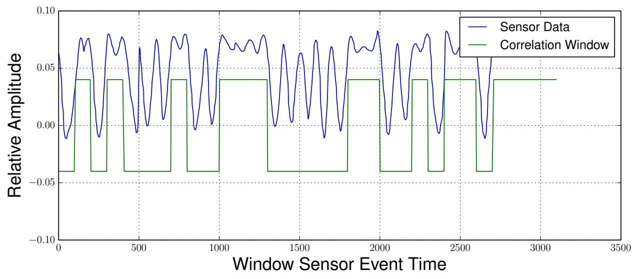


Figure 5: Synchronization Window

Table 1: Correlation and Synchronization Notations

Notation	Definition
T	The set of all sensor recorded times
j	The relative position of each time report in T
CW_j	The correlation window (matched filter) starting at position j
$Sv_j \subset S$	The set of sensor values S within a correlation window starting at j
T_{synch}	The time yielding the best synchronization estimate
R	The set of all correlations
R_j	The j^th correlation result where $R_j = CW_j \cdot Sv_j$

Marvasti [24] asserted that non-uniform sampling intervals may be used in signal reconstruction if the average of the interval rate satisfies the Nyquist rate. Therefore, the matched filter approach suffices provided this condition is met. Furthermore, it must be true for the event samples relative to the pulsewidth. If so, events are processed as is, otherwise the pulsewidth is increased until the condition is met. This addresses the Android Nyquist sampling limitation.

The dot product of the matched filter with the signal window is computed. The result is normalized over the window's event count and maintained for analysis.

Table 1 shows the notations defined for synchronization. We derive T_{synch} (Equation 4), the sequence's communication frame temporal reference start point.

$$T_{synch} = \max |R_j| \forall R \tag{4}$$

where

$$R_j = CW_j \cdot Sv_j \tag{5}$$

Figure 5 illustrates this synchronization method applied to a Samsung Galaxy S6's X axis accelerometer. The window's amplitude was reduced in size to support visualization.

Pulse Width Determination

After obtaining the operating frequency(ies), the *source* synthesizes a new set of test patterns using those frequency(ies) while varying the pulsewidth. The *source* measures the bit error rate (BER) associated with each pulsewidth, calculates the throughput using Equation (8), selects the desired throughput level and uses the corresponding pulsewidth for Phase II. Operationally, pulsewidth selection is a function of reporting rate which is device limiting per the provider's configuration and / or sensor characteristics. If narrow pulsewidths, ≈ 10 msec or less are desired, repetition codes, interpolation or other compensation techniques may be needed to account for lower sensor event reporting rates.

Sweep Granularity

Initially, we used 50 Hz sweep increments to restrict transmission time and reduce power consumption. Since the risk is missed channel recognition, most of our reported results can be considered a lower bound for channel identification and performance.

3.4 Phase II: Data Transfer

Once the optimal channel parameters have been identified, the *source* packages the compromised data into transmission frames. Each frame consists of a preamble, the payload, a CRC field and an end of frame marker. The initial preamble consists of a synchronization sequence, parametric information and a shared secret magic number used by the *sink* to determine pulse width. With higher BER environments, payload coding (spreading) may be added. An additional shared secret in the initial preamble designating the spreading gain configuration could suffice. Like the *source* in Phase I, the *sink* processes the received data using the matched filter and synchronizes to the transmission frame. It subsequently applies discrimination techniques to retrieve the parametric information and compromised data prior to repackaging for off-board transmission.

3.5 Key Stealth Factors

Zero Permissions and Detectability: Engaging only zero permission resources, i.e., the speaker, sensors and media players is key. Detection risk is reduced by avoiding microphones, the vibrator, shared files and message passing Android APIs.

Audibility from Spectral Components: User hearing is another detection concern. We selected the operating frequency band in part, based on speech studies by Beiter and Talley [2], who studied the auditory response of college age women. They observed hearing threshold changes of −160 dB/octave between 16 kHz and 20 kHz. More recently, Jungmee Lee et. al. [21] demonstrated that significant sound pressure level power increase (≥ 8 dB) is needed to normalize (flatten) hearing above 15 kHz.

Consequently, we selected 16 kHz as our lower frequency limit. Looking ahead, most vulnerabilities were uncovered at frequencies greater than 20 kHz, while all were above 17.7 kHz. Operating below 20 kHz depends on the attacker's risk tolerance. The upper limit was set at below the Nyquist frequency, assuming the media player used a 48 kHz sampling rate. Although operable to 24 kHz, distortion is perceived at frequencies near the Nyquist frequency.

Audio Volume and Distortion: Certain devices produced audible artifacts during FID testing. These intermittent clicks we suspect are the result of clipping, inter-modulation distortion or mechanical noise from speaker movement. These were noticeable in most devices under test (DUT) when sending tones at the maximum (unity) sine wave amplitude (SWA) and at full player and full speaker volume settings. In virtually all cases, reducing the SWA to 0.99 and speaker volume to $getStreamMaxVolume - 1$ eliminated the artifacts.

3.6 Performance Boosting Design

We developed two techniques to enhance performance: Multichannel operation and Amplitude Shift Keying (ASK). For multichannel operation, performance improvement is a function of the total number of the device's contributing sensor axes. In this case, we demonstrated multi-axial, single sensor responses. For example, the Galaxy S6 and S5 X and Y accelerometer axes respond to different and non-harmonically overlapping ultrasonic frequencies. By generating a waveform representing a weighted summation (to avoid clipping) of independent axis-specific waves, we can evoke axial responses concurrently. Channel setup requires executing a series of tests with different weights, measuring axial BERs and identifying those factors yielding the highest aggregate capacity.

ASK feasibility resulted from accelerometer sensitivity to speaker power levels. With ASK, the bit rate (BR) is a function of symbol rate and the log_2 of the symbol set size, see Equation (6). Channel setup requires a series of tests where each symbol's max and min over an observed range should not overlap with any other symbols' max or min. The degree of amplitude separation drives the symbol count. The receiver can be informed of the symbol count and symbol amplitude range in the Phase II preamble. Advanced approaches can combine these techniques to further increase channel capacity.

$$BR = log_2(\#symbol\ levels) \times\ Symbol\ Rate \tag{6}$$

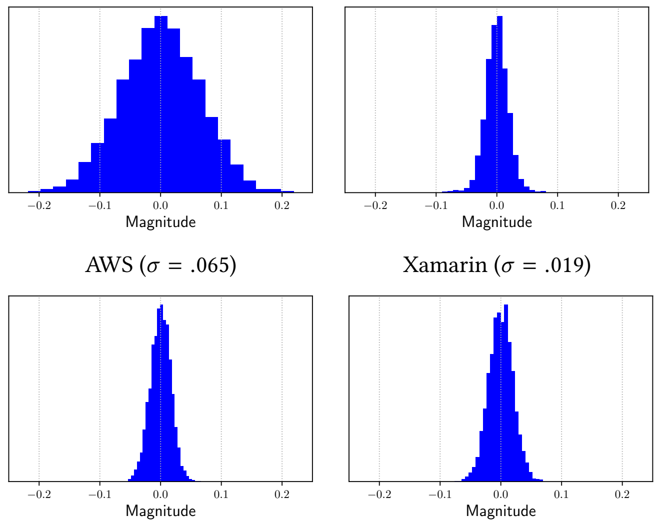
4 TESTING AND EVALUATION APPROACH

4.1 Test Environments

Three primary test environments were used in this study; a software development laboratory, the *AWS Device Farm*™ and the *Xamarin TestCloud*™. The laboratory test pool consisted of devices available to laboratory personnel. Within the lab environment, location and orientation (screen up) were fixed. Placement was near laboratory personnel performing normal activities. Within several hundred feet outside the site was an uncooperative environment that included construction projects, nearby rail lines and vehicular traffic from the urban surroundings. Less frequent perturbations emanated from the Android device directly i.e., ringing and event alerts. We believe that the environment was well suited for examining covert channel potential due to its similarity to urban residential environments. Since the TestCloud and Device Farm environments were uncontrollable (i.e., orientation, ambient acoustic and vibrational noise, co-resident testing), the devices were tested as is. Device Farm and TestCloud devices were oriented similarly (screen up), verified by Z axis readings. At testing inception, TestCloud had over 1200 unique Android device types while Device Farm had more than 150. We also ran FID tests in a busy café to further evaluate channel robustness in highly frequented locations.

Environmental Assessment

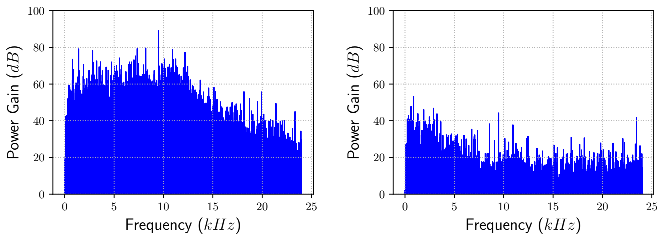
Microphone and accelerometer readings were recorded concurrently for a Samsung Galaxy S6 in each test environment. We assumed that calibration errors and component drift tracked similarly among identical device types. Since calibration data nor access



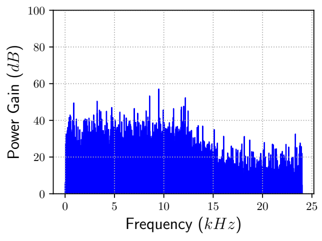
AWS (σ = .065) Xamarin (σ = .019)

Café (σ = .017) Lab (σ = .020)

Figure 6: Accelerometer Distribution



AWS Xamarin



Café

Figure 7: Environment Noise Spectra

to the physical devices themselves in the cloud cases was available, each measurement was relative to its last calibrated reference.

Figure 6 and Figure 7 show the distribution of the accelerometer magnitudes ($\sqrt{x^2 + y^2 + z^2}$) and their corresponding ambient environments' frequency spectra respectively. Since the distributions are *Gaussian* in form, their standard deviation σ can be used to estimate the accelerometer noise. The recorded audio noise gain, P, can be used to estimate environmental noise levels. The AWS hosted devices exhibit high accelerometer noise ($0.035 \leq \sigma \leq 0.15$) and environmental noise gain ($30\,\text{dB} \leq P \leq 80\,\text{dB}$) relative to its Laboratory equivalent (i.e., three to eight orders of magnitude higher), suggesting that the AWS™ environment is very noisy.

As a group, the Xamarin TestCloud™, café and laboratory had similar accelerometer and environmental noise levels. Note that the sensors report ultrasonic, audible and infrasonic signal components which all compete for bandwidth.

4.2 Data Collection

Phase I testing consisted of executing FID tests at least five times for each device under test (DUT). Follow-up bit error rate testing confirmed channel existence thereby avoiding false positive results. Phase II testing consisted of a series of test and evaluate cycles until the aggregate payload size was at least ten times the inverse of the aggregate BER.

Automation: *Calabash-android* scripts, based on a user interface (UI) automation library for Android apps, controlled the device data collection process using the following test sequence: 1) Start the audio pattern, 2) Wait for audio pattern completion, 3) Upload the sensor files to our server, 4) Wait for upload completion and 5) Loop as configured.

Noise Mitigation: Each DUT underwent repetitive frequency identification tests to mitigate spurious environmental effects. Although results were averaged over the number of tests, smoothing effectiveness depends on perturbation magnitude and duration. Although most environmentally induced noise is observable in the Z axis accelerometer when at rest (assuming it rests with the screen up), occasionally the perturbation couples over to X and Y leading to inter-run variation. Sustained background noise would require additional processing such as noise cancellation, greater spreading and larger pulse widths and window lengths. Mitigation is easier than in traditional communication systems due to the closed loop nature of the channel since the transmitter has access to the same sensor data as the receiver, allowing for dynamic channel parameter adjustment and / or frame retransmission.

4.3 Device Selection

We implemented our *source* and *sink* app pairs on 18 models of smartphones (a total of 28 physical devices). These models were selected based on their popularity, availability in the test environments, Android versions, and sensor sampling rate. Although we sought to target the most popular phones during the selection process, we saw no clear consensus market ranking. However, Suvarna and Top101news [35, 36] suggest that the Samsung S6 is one most popular smartphones with a 5.47% global market share as of the first half of 2016. The same source writes that the Nexus 6, Galaxy S6 and Galaxy Note 4 comprise ≈ 20% of all Android devices in the US market. Finding multiple instances of devices across environments was challenging. Balancing popularity, availability and repeatability needs, we reduced our sample size to the 28 devices. The specific list is available in Table 2.

4.4 Evaluation Approach

We applied classic communications theory to evaluate channel capacity and throughput. Shannon capacity provides a theoretical measure of effectiveness (MOE) using realized bit error rates. Throughput provides an MOE that includes additional implementation factors such as payload length and frame length. While there are numerous methods to improve performance, two of which (multichannel and Amplitude Shift Keying) were implemented and are discussed in Section 6.

Table 2: Tested Devices by Environment

Device	Model	Laboratory	TestCloud	Device Farm
HTC	10		*	
HTC	One M9	*	*	
Huawei	HONOR 6		*	
Huawei	Nexus 6P		*	
Huawei	P8lite	*		
LG	G5		*	
LG	Nexus 5	*	*	
LG	Nexus 5x	*		
LG	Nexus 7	*		
LG	Optimus L90	*		
Motorola	G3		*	*
Motorola	Nexus 6		*	
OnePlus	One		*	*
Samsung	Galaxy Note 4	*	*	*
Samsung	Galaxy S3	*		
Samsung	Galaxy S5	*	*	*
Samsung	Galaxy S6	*	*	*
Sony	Xperia Z3		*	

Note: * indicates evaluated within the specific environment

Theoretical Capacity: We assume a binary symmetric channel (BSC) [7], with a Bit Error Rate, *BER*. The Shannon channel capacity *C*, provides a theoretically achievable upper bound on the channel throughput and is computed as shown in Equation (7) factoring in the pulsewidth PW. This theoretical upper bound, is practically approachable with proper codes of large block length (e.g., turbo-codes, or LDPC codes) [22].

$$C = \frac{1}{PW} \times (1 + BER \times \log_2(BER) + (1 - BER) \times \log_2(1 - BER)) \quad (7)$$

Throughput: Throughput, TH offers an implementation and performance specific assessment of transmission rates. For an uncoded channel:

$$TH = \frac{1}{PW} \times \frac{Length - FramingBits}{Length} \times (1 - BER)^{Length} \quad (8)$$

5 EVALUATION RESULTS

This section presents the Channel Identification and Bit Error Rate testing results and concludes with a discussion covering intra-family similarity (identicality).

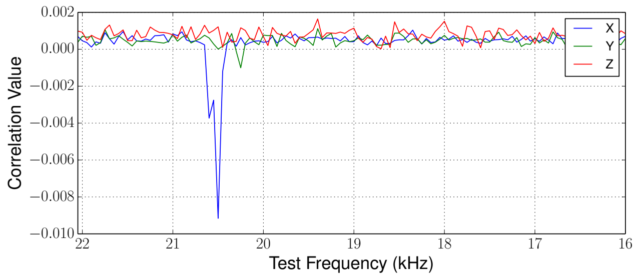
5.1 Channel Identification Results

Channel ID Results Summary

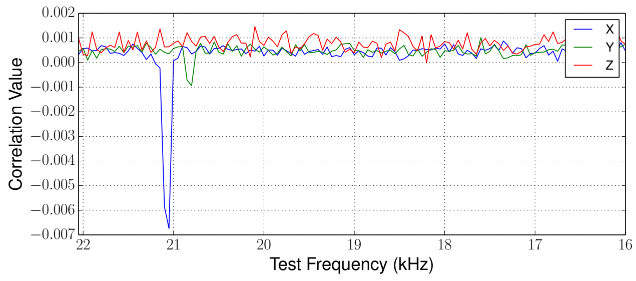
Nearly 25% of the 18 unique device models evaluated tested positive for vulnerability to our attack. We observe that the vulnerability is not universally present within a family of products lines, i.e., successful attacks could not be carried out on the Galaxy S3 vs. the Galaxy S6 and Galaxy S5, all Samsung products.

Devices with Positive Results

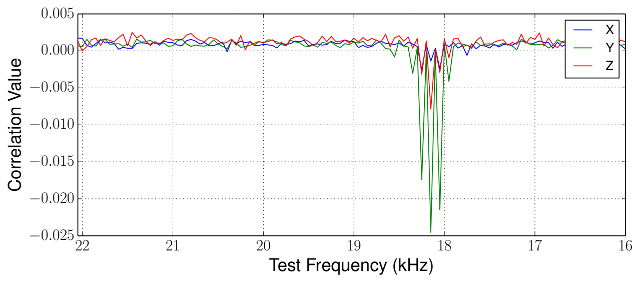
Figure 8 illustrates the sensor measurement correlation test results as a function of frequency. This highlights sensor sensitivity to the frequency sweep stimulus. Three Samsung devices, the S6, S5 and Note 4, exhibited positive responses to the attack as seen in Figures 8a to 8c respectively. Nexus 6 testing also revealed a potential channel, see Figure 8d. These plots provide graphical representations of Equation (2), showing each frequency's correlation score. The preferred operating point is the frequency associated with the largest excursion from 0.000. Channel presence was confirmed in all cases using bit error rate testing whose results are discussed in Section 5.2.



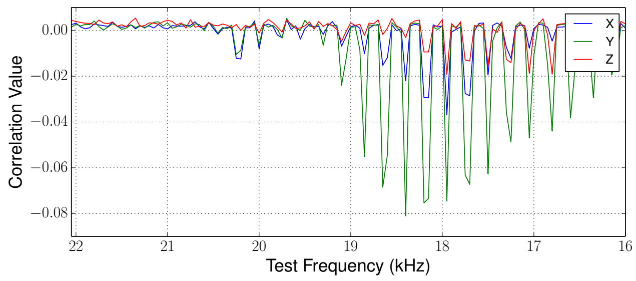
(a) Samsung Galaxy S6



(b) Samsung Galaxy S5



(c) Samsung Galaxy Note 4



(d) Motorola Nexus 6

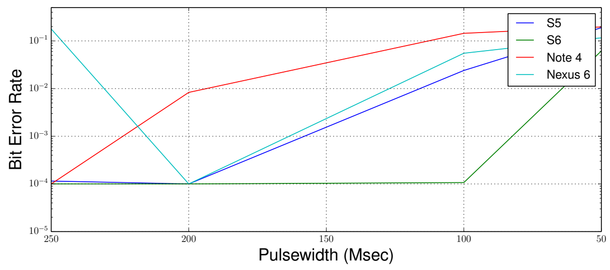
Figure 8: Devices with Vulnerability

Table 3: Devices with Indeterminate Channels

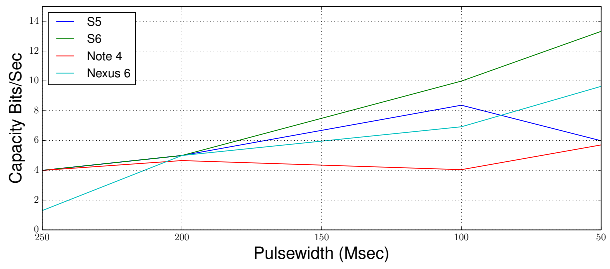
Device	Model	API	Device	Model	API
HTC	One M9	22	LG	Nexus 5x	23
HTC	Ten	23	LG	Nexus 7	17
Huawei	HONOR 6	17	LG	Optimus L90	19
Huawei	Nexus 6	23	Motorola	G3	22
Huawei	P8lite	22	OnePlus	One	19
LG	G5	23	Samsung	Galaxy S3	19
LG	Nexus 5	22	Sony	Xperia Z3	21

Devices with Indeterminate Results

We were unable to establish a channel using our frequency range in fourteen devices listed in Table 3. In cases where the correlation values deviated from the nominal level yet were weak (less than



(a) Bit Error Rate vs. Pulse Width



(b) Capacity vs. Pulsewidth

Figure 9: Error and Capacity Summary

±0.004), we confirmed the false positive with BER testing. Occasionally, simultaneous peaks occurred in more than one sensor axis. This indicated a strong external force such as seen during significant impulse noise (shock) or a periodic external stimulus. Follow-up bit error rate testing to identify false positive conditions is needed since the Note 4 and Nexus 6 have simultaneous peaks as a legitimate operating condition.

Environmental Influence

All device types demonstrating susceptibility were proven in the laboratory and/or the Xamarin TestCloud™ and café. No channel could be established with these same devices in the AWS Device Farm™ due to noise levels. From Section 4, AWS™ had the highest sound pressure levels and Accelerometer variance of all testing environments.

5.2 Error Testing Results

We executed BER testing, per Section 4.2, on each channel capable device at four pulse widths, 250, 200, 100, and 50 msec. If no error was found we conservatively set the BER to 10^{-4}.

Bit Error Rate Results Summary

The BER vs. pulse width results are provided in Figure 9a and summarized in Table 4. In general, the BER decreases inversely with increased pulse width. Excluding the Galaxy Note 4, and the Nexus 6, the S5 and S6 had BERs in the 10^{-3} or better range at 250 and 200 msec pulse widths. All, excluding the Galaxy S6, tail off below 200 msec with BERs worse than 10^{-2} as pulse widths approach 100 msec. At 50 msec, all exhibited degraded performance. Interestingly, the 250 msec BER for the Nexus 6 was less than its 200 msec measurement while the BER for the Note 4, S5 and S6 peaked at 10^{-4} and monotonically degraded as PW deceased.

Regarding the Galaxy Note 4, we anticipated that as an older device, it would have poorer performance than its test peers. This is apparent as we observe poor resolving capability below 250 msec.

Table 4: Error Rate Summary

Device	Model	Pulse Width (Msec)	BER
Motorola	Nexus 6	50	0.1162
Motorola	Nexus 6	100	0.0552
Motorola	Nexus 6	200	0.0001
Motorola	Nexus 6	250	0.1780
Samsung	Galaxy S5	50	0.1898
Samsung	Galaxy S5	100	0.0240
Samsung	Galaxy S5	200	0.0001
Samsung	Galaxy S5	250	0.0001
Samsung	Galaxy S6	50	0.0616
Samsung	Galaxy S6	100	0.0001
Samsung	Galaxy S6	200	0.0001
Samsung	Galaxy S6	250	0.0001
Samsung	Galaxy Note 4	50	0.1965
Samsung	Galaxy Note 4	100	0.1444
Samsung	Galaxy Note 4	200	0.00832
Samsung	Galaxy Note 4	250	0.0001

Table 5: Device Pool Identicality

Device	Model	Series	Location	Test Case	X Freq	Y Freq	Z Freq
Motorola	Nexus 6	-	Xamarin	1	20500	**17950**	16600
Motorola	Nexus 6	-	Xamarin	2	17950	**18400**	17950
Samsung	Galaxy S5	V	TF	1	**21050**	17600	20100
Samsung	Galaxy S5	V	TF	2	**21200**	20650	17700
Samsung	Galaxy S5	V	Xamarin	3	20150	**17950**	20400
Samsung	Galaxy S6	T	TF	1	**20500**	19400	19400
Samsung	Galaxy S6	T	TF	2	**20700**	20550	21050
Samsung	Galaxy S6	F	Xamarin	3	**20350**	20950	20200
Samsung	Galaxy S6	F	Xamarin	4	16050	**20650**	16850
Samsung	Galaxy Note 4	T	TF	1	18250	**18150**	18150
Samsung	Galaxy Note 4	F	Xamarin	2	18150	**18050**	17950
Samsung	Galaxy Note 4	F	Xamarin	3	18000	**18000**	18000

Notes: TF = Test Facility, NA = Unremarkable Correlation, **Boldface** = Best Freq.

Resident in the Xamarin cloud, the Nexus 6 exhibited unusual performance at 250 msec. This is worthy of additional study as it is the only device exhibiting counter-intuitive behavior.

5.3 Device Family Uniformity

We compared the FID performance of three Galaxy S5s, four Galaxy S6s, three Galaxy Note 4s and two Google Nexus 6s to assess performance of similar device types. The tests were conducted in either the local test facilities (TF) or in the Xamarin TestCloud™. All device types, excluding the Nexus 6s, had at least one instance in both.

Each tested device exhibited positive FID responses, see Table 5 with the Nexus 6s exhibiting multi-frequency responses. In at least one case there was overlap in sensitive frequencies (18.4 kHz), albeit not at the peak sensitivities. Regarding the remaining devices, no two 'like' devices sampled demonstrated sensitivity to the same frequencies for a given axis. Within a specific device family, i.e., the Galaxy S6 (excluding the Nexus 6), the frequency differences are less than 1100 Hz. The sample size is too small to draw any conclusions regarding the extent of the differences. Despite identical configurations as in the 'T' series S6 and the 'F' series Note 4, differences were observed. However, it demonstrates the effect of element (component, OS, manufacturing process) variability between identical versions and variants. This reinforces the viability of the self-identification method described earlier, allowing for the attack independence.

Surprisingly, one of the Xamarin S6s and Xamarin S5s had an axial Y component as the preferred operating frequency that didn't occur in other phones. We checked the Z axis readings for orientation yet the gravity and magnitude levels were consistent. We suspect damage or a different or modified chip set was utilized.

6 CAPACITY, THROUGHPUT AND PERFORMANCE BOOSTING

This section includes a summary of baseline channel, multichannel and Amplitude Shift Keying performance.

6.1 Capacity

We computed the channel capacity from Equation (7) for each of the susceptible devices using the measured BERs. The results are illustrated in Figure 9b. The capacity levels which we hope to theoretically achieve, approach 14, 10, 6 and 6 for the S6, Nexus 6, S5 and Note 4 respectively.

6.2 Throughput

As with the capacity analysis, we derived the throughput from the BERs by applying Equation (8) with frame lengths ranging from 64 to 512 bits. We assumed lengths for the synchronization frame, CRC and end of frame marker (EOF) of 23, 16 and 8 respectively.

The best throughput, ≈ 4 bits/sec, occurs on the S6 (see Figure 10a) with a 64 bit frame length and a PW of 100 msec. The S5 (see Figure 10b) and Nexus 6 (see Figure 10d) similarly offer a throughput of ≈ 2 bps with a 200 msec pulse width, also with length of 64 bits. The Note 4, see Figure 10c, throughput peaks out with a 250 msec PW and a frame length of 64 bits.

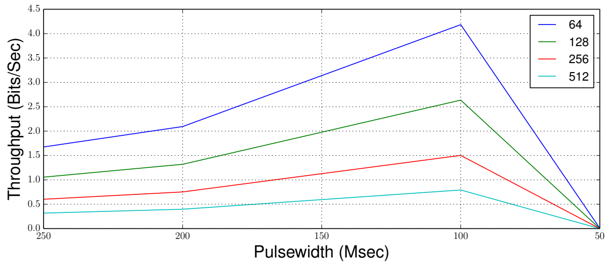
Given the discrepancy between channel capacity and throughput, it is clear that the channel is suboptimal as configured. Throughput degradation occurs at the narrowest pulse widths which are typically coincident with the highest BERs. Utilizing coding techniques [22] at these particular rates, would allow the throughput to approach theoretical capacity levels.

6.3 Performance Boosting Evaluation

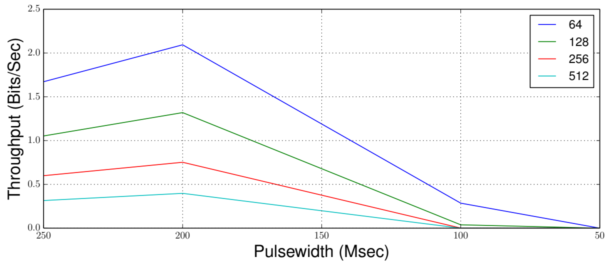
We use a simple example to put the results in context. We assume that there are 308 [3] contacts stored in a device. If each contact's size is 100 bytes, the entire contact list can be leaked in 6.8 hours assuming a 10 bps exfiltration rate. Additionally, approximately 90% of adults have less than 10 passwords to manage [19]. Assuming each has length 12 bytes, the password management file could be exfiltrated in ≈ 1.5 minutes. We can achieve greater performance with the following boosting techniques.

Multichannel Performance

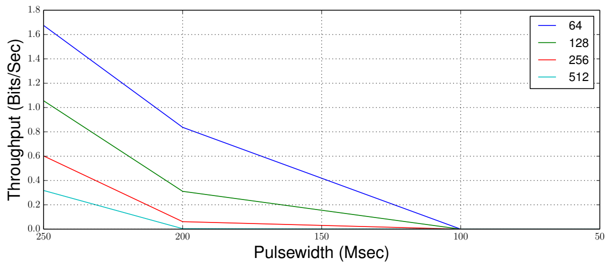
Since several devices exhibited multi-axial responses to non-overlapping and non-harmonically related frequencies, we sought to evaluate the feasibility of multichannel communications. To evaluate this hypothesis, we generated a multichannel test pattern for the Galaxy S6 by aggregating the individual X / Y axis peak response frequencies (20,500 Hz, 20,250 Hz) with amplitude weights of 1.0/0.0, 0.9/0.1, 0.8/0.2, ..., 0.0/1.0. BER tests were subsequently executed on the aggregated pattern yielding capacity gains nearly twice that of a single channel (see Figure 11a). Although suboptimal, this demonstrated the basic multichannel capability. Improved BERs may be realized by including peak to average power correction to minimize energy loss.



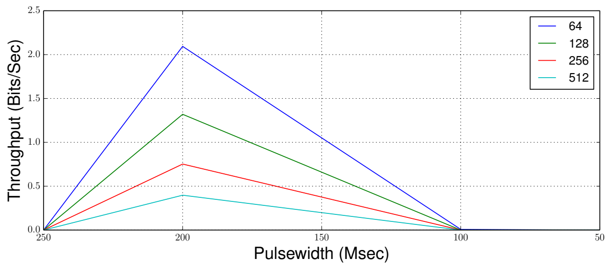
(a) Galaxy S6



(b) Galaxy S5



(c) Note 4



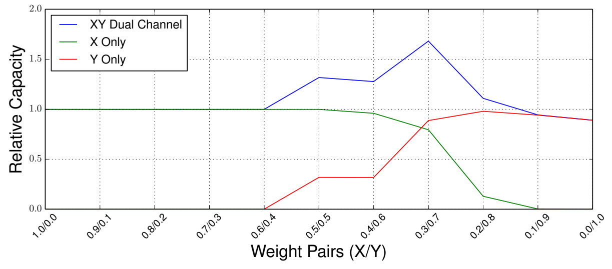
(d) Nexus 6

Figure 10: Throughput vs. Pulse Width for Uncoded Communication

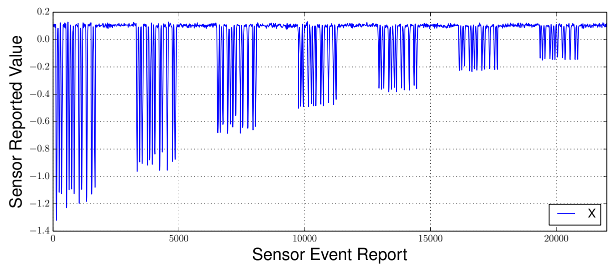
We analyzed potential cross coupling effects by calculating the Spearman [34] rank order correlation and Pearson product-moment correlation [39] coefficients for each weighted pair to identity any bleed-over, i.e., X affecting Y, Y affecting X etc. From Table 6, we see very low axial coupling. Ignoring directionality, the worst-case correlation was less than 0.119, suggesting that interference is inconsequential for this particular device-frequency pair.

Amplitude Shift Keying

We selected eight distinct, non-overlapping amplitude symbol levels to demonstrate ASK. Figure 11b shows the symbols amplitudes in 16 bit patterns separated by quiet periods for illustration



(a) Dual Channel



(b) ASK

Figure 11: Performance Boosting

Table 6: Multichannel Axial Correlation

Weight (X/Y)	Correlation Type	Axial Pair	Coefficient	Two Tailed p Value
0.0/1.0	Pearson	XY	-0.119	$< 10^{-5}$
0.0/1.0	Pearson	XZ	0.018	0.0002
0.0/1.0	Pearson	YZ	-0.001	0.7663
0.0/1.0	Spearman	XY	-0.091	$< 10^{-5}$
0.0/1.0	Spearman	XZ	0.023	$< 10^{-5}$
0.0/1.0	Spearman	YZ	-0.0014	0.767
1.0/0.0	Pearson	XY	-0.079	$< 10^{-5}$
1.0/0.0	Pearson	XZ	0.001	0.827
1.0/0.0	Pearson	YZ	0.0074	0.106
1.0/0.0	Spearman	XY	-0.0531	$< 10^{-5}$
1.0/0.0	Spearman	XZ	0.0013	0.7801
1.0/0.0	Spearman	YZ	0.0067	0.1421

purposes. This corresponding bit rate (BR) boost from Equation (6), is three times the single amplitude rate.

7 MITIGATION

Adverse effects, resource consumption and channel flexibility drive mitigation effectiveness. Spectral limiting techniques i.e., filtering frequencies above 20 kHz diminishes sound quality due to the hypersonic effect [29], discouraging those seeking higher fidelity sound. Reducing sensor reporting rates slows the transmission rate by forcing an increase in channel pulsewidth. Sensor fusion may offer a solution where multiple sensor readings combine to infer a valid condition. However, the channel's frequency summation ability supports operating with concurrent sensor stimulation, thereby potentially thwarting this defense. Some a priori characterization of valid conditions are needed to generate the appropriate approach.

Other possibilities relate to sensing activities such as monitoring player content over time. Although resource intensive and complex, a defender could monitor the sound (spectral) content and perform envelope detection on data snippets sent to the device players. Preferably, the OS could monitor apps that use the speaker(s) and the sensors, reporting resource usage normalized over access time to ascertain anomalous behavior. Unfortunately, this might be untimely for real time detection of one-time exfiltration events.

Relying strictly on permissions for access control to prevent channel execution is perilous historically. Monitoring all apps as resources are accessed and subsequently querying the user for direction, depends on user sophistication and requires patience. For example, solutions such as Audroid [30] target audio covert channel mitigation. Defending this particular attack would require approval each time there are speaker only requests. These types of frequent query approaches may encourage the user to disable *enforcement* mode or worse, root the device which encourages a broader set of attacks. Even with this defensive tactic and a sophisticated user, functional plausibility enhances obfuscation. Consider the *source* masquerading as calendar app which utilizes an audible alarm for appointment alerts. In this case, it has legitimate need to use the speaker. The *sink* alternatively, needs permission for sensor access. However, if it masquerades as a game, a legitimate use, the defense collapses. If the *source* is blocked from sensor access, it could cumbersomely and inefficiently use device family historical data, transmitting across a large spectrum without feedback.

8 RELATED WORK

Farshteindiker [15], based on Son's [33] work, developed a channel that relies on an 'implant', an external device that stimulates the smartphone's gyroscope with high frequency energy. Due to sensitivity with the sensor module location, this device contacts the victim smartphone at a predetermined position. Complexity is high, requiring a priori target information.

Schlegel [32], Marforio [23], Yue [37] and Okhravi [27] addressed system setting manipulation. Examples included vibration setting, volume setting, screen state, socket creation / breakdown discovery, intent type, processor frequency and timing manipulation. Although channel data rates ranged from 3 to over 3000 bps, current Android security features (i.e., using SELinux) limit the ability to create such channels.

Microphone based inter-device channels face the challenge of prearranged proximity. Examples include: Do [11] who used Audio Frequency Shift Keying operating within the 20 kHz to 22 kHz band to communicate between air gapped devices; Hanspac [17], who developed a mesh channel between two Lenovo T400 series computers operating in the 18 kHz+ range and O'Malley [28], who demonstrated inter-computer communications in the 20 kHz to 23 kHz band.

Dey [10] and Das [8] demonstrated the existence of device signatures unique to the manufacturing process. Variation (see section 5.3) in our channel frequencies supports their observations.

9 CONCLUSION

We demonstrated that a zero permissions, ultrasonic covert channel can be created and self-parametrized on certain Android devices using the speaker and the sensor suite as the channel's endpoints. Individual axis channel capacity approached 14 bits per second limited by the testing scope and device type. Furthermore, we demonstrated that we can increase the data rate by including multi-channel and Amplitude Shift Keying capabilities to yield rates up to and including 3 times the basic channel capability. The path forward includes broadening the test base to include next generation devices and evaluating suitable mitigation techniques.

REFERENCES

[1] A. Al-Haiqi, M. Ismail, and R. Nordin. 2014. A New Sensors-Based Covert Channel on Android. *The Scientific World Journal* (2014).

[2] Robert Beiter and James Talley. 1976. High-Frequency Audiometry above 8000 Hz. In *Audiology/Audiologie.* 207–214.

[3] Frank R. Bentley and Ying-Yu Chen. 2015. The Composition and Use of Modern Mobile Phonebooks. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15).* 2749–2758. http://doi.acm.org/10.1145/2702123.2702182

[4] A. Bianchi, J. Corbetta, L. Invernizzi, Y. Fratantonio, C. Kruegel, and G. Vigna. 2015. What the App is That? Deception and Countermeasures in the Android User Interface. In *2015 IEEE Symposium on Security and Privacy.*

[5] Pew Research Center. 2017. Mobile Fact Sheet. http://www.pewinternet.org/fact-sheet/mobile/. (2017). (visited: 2017-02-22).

[6] Thomas M. Cover and Joy A. Thomas. 1991. *Elements of Information Theory.* John Wiley and Sons Inc.

[7] Thomas M. Cover and Joy A. Thomas. 2006. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing).* Wiley-Interscience.

[8] Anupam Das, Nikita Borisov, and Matthew Caesar. 2014. Do You Hear What I Hear?: Fingerprinting Smart Devices Through Embedded Acoustic Components. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS '14).* ACM.

[9] Luke Deshotels. 2014. Inaudible Sound as a Covert Channel in Mobile Devices. In *8th USENIX Workshop on Offensive Technologies (WOOT 14).* USENIX Association, San Diego, CA. https://www.usenix.org/conference/woot14/workshop-program/presentation/deshotels

[10] Sanorita Dey, Nirupam Roy, Wenyuan Xu, Romit Roy Choudhury, and Srihari Nelakuditi. 2014. AccelPrint: Imperfections of Accelerometers Make Smartphones Trackable. In *Proceedings of NDSS.*

[11] Quang Do, Ben Martini, and Kim-Kwang Raymond Choo. 2015. Exfiltrating data from Android devices. *Computers and Security* 48 (2015).

[12] Mahmut Eksioglu and Ali Iseri. 2015. An Estimation of Finger-Tapping Rates and Load Capacities and the Effects of Various Factors. *Human Factors* 57, 4 (2015), 634–648. https://doi.org/10.1177/0018720814563976

[13] Endevco. 2009. Practical understanding of key accelerometer specifications TP328. https://www.endevco.com/news/archivednews/2009/2009_09/TP328.pdf. (2009). (visited: 2016-10-16).

[14] Ericson. 2016. Ericsson Mobility Report. https://www.ericsson.com/assets/local/mobility-report/documents/2016/ericsson-mobility-report-november-2016.pdf. (2016). (Retrieved January 22, 2017).

[15] Benyamin Farshteindiker, Nir Hasidim, Asaf Grosz, and Yossi Oren. 2016. How to Phone Home with Someone Else's Phone: Information Exfiltration Using Intentional Sound Noise on Gyroscopic Sensors. In *10th USENIX Workshop on Offensive Technologies (WOOT 16).*

[16] W. Gasior and L. Yang. 2012. Exploring Covert Channel in Android Platform. In *Cyber Security (CyberSecurity), 2012 International Conference on.* 173–177.

[17] Michael Hanspach and Michael Goetz. 2013. On Covert Acoustical Mesh Networks in Air. *JCM* 8 (2013), 758–767.

[18] IDC. 2016. Worldwide Smartphone Volumes Relatively Flat in Q2 2016 Marking the Second Straight Quarter Without Growth. Press Release. (July 2016). http://www.idc.com/getdoc.jsp?containerId=prUS41636516.

[19] Shirley Duglin Kennedy. 2016. Your Password Has Expired and Must Be Changed. http://www.infotoday.com/IT/apr16/Kennedy--Your-Password-Has-Expired-and-Must-Be-Changed.shtml. (2016). (visited: 2016-10-24).

[20] J. F. Lalande and S. Wendzel. 2013. Hiding Privacy Leaks in Android Applications Using Low-Attention Raising Covert Channels. In *Availability, Reliability and Security (ARES), 2013 Eighth International Conference on.*

[21] Junmee Lee and et. al. 2016. Behavioral Hearing Thresholds Between 0.125 and 20 kHz Using Depth-Compensated Ear Simulator Calibration. In *Ear and Hearing, 33(3).*

[22] Shu Lin and Daniel J. Costello. 2004. *Error Control Coding, Second Edition.* Prentice-Hall, Inc.

[23] Claudio Marforio, Hubert Ritzdorf, Aurélien Francillon, and Srdjan Capkun. 2012. Analysis of the Communication Between Colluding Applications on Modern Smartphones. In *Proceedings of the 28th Annual Computer Security Applications Conference (ACSAC '12).*

[24] Marvasti. 2001. *Nonuniform sampling: theory and practice.* Kluwer Academic/Plenum Publishers, New York.

[25] Yan Michalevsky, Dan Boneh, and Gabi Nakibly. 2014. Gyrophone: Recognizing Speech from Gyroscope Signals. In *23rd USENIX Security Symposium (USENIX Security 14).* USENIX Association.

[26] 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).* https://doi.org/10.1109/SP.2016.31

[27] H. Okhravi, S. Bak, and S. T. King. 2010. Design, implementation and evaluation of covert channel attacks. In *Technologies for Homeland Security (HST), 2010 IEEE International Conference on.* 481–487.

[28] Samuel Joseph O'Malley and Kim-Kwang Raymond Choo. 2014. Bridging the Air Gap: Inaudible Data Exfiltration by Insiders, In 20th Americas Conference on Information Systems - AMCIS. *Journal of Networks, Association for Information Systems.*

[29] Tsutomu Oohashi, Emi Nishina, Manabu Honda, Yoshiharu Yonekura, Yoshitaka Fuwamoto, Norie Kawai, Tadao Maekawa, Satoshi Nakamura, Hidenao Fukuyama, and Hiroshi Shibasaki. 2000. Inaudible High-Frequency Sounds Affect Brain Activity: Hypersonic Effect. 83, 6 (2000), 3548–3558.

[30] Giuseppe Petracca, Yuqiong Sun, Trent Jaeger, and Ahmad Atamli. 2015. AuDroid: Preventing Attacks on Audio Channels in Mobile Devices. In *Proceedings of the 31st Annual Computer Security Applications Conference (ACSAC 2015).* ACM, New York, NY, USA, 181–190. https://doi.org/10.1145/2818000.2818005

[31] Rob O'Reilly, Alex Khenkin, and Kieran Harney. 2009. Sonic Nirvana: Using MEMS Accelerometers as Acoustic Pickups in Musical Instruments. http://www.analog.com/en/analog-dialogue/articles/mems-accelerometers-as-acoustic-pickups.html. (Feb 2009). Accessed: Feb, 2017.

[32] Roman Schlegel, Kehuan Zhang, Xiaoyong Zhou, Mehool Intwala, Apu Kapadia, and XiaoFeng Wang. 2011. Soundcomber: A Stealthy and Context-Aware Sound Trojan for Smartphones. In *Proceedings of the 18th Annual Network & Distributed System Security Symposium (NDSS).*

[33] 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 *Proceedings of the 24th USENIX Conference on Security Symposium (SEC'15).*

[34] C. Spearman. 1904. The Proof and Measurement of Association between Two Things. *The American Journal of Psychology* 15, 1 (1904), 72–101. http://www.jstor.org/stable/1412159

[35] Sid Suvarna. 2016. Most Popular Android Smartphones of 2016, According to AnTuTu. http://n4bb.com/most-popular-android-smartphones-2016. (2016). (visited: 2016-10-02).

[36] Top101news. 2015. Top Ten Best Selling Smartphones in the World. http://top101news.com/2015-2016-2017-2018/news/products/best-selling-smartphones-world/. (2015). (Retrieved October 2, 2016).

[37] Mengchao Yue, William H. Robinson, Lanier Watkins, and Cherita Corbett. 2014. Constructing Timing-based Covert Channels in Mobile Networks by Adjusting CPU Frequency. In *Proceedings of the Third Workshop on Hardware and Architectural Support for Security and Privacy (HASP '14).* ACM.

[38] 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 (MobiSys '15).*

[39] Daniel Zwillinger and Stephen Kokosk. 2000. *CRC -Sstandard Probability and Statistics Tables and Formulae.* Chapman and Hall/CRC, Boca Raton London New York Washington, D.C.