Ultrasound Proximity Networking on Smart Mobile Devices for IoT Applications

Ed Novak*, **Zhuofan Tang***, **Qun Li**†

***Abstract*—Sharing small pieces of information such as URLs, Internet of Things (IoT) commands, or encryption keys is an extremely common use case in IoT applications. These are examples of transient, spontaneous proximity networking, in which both the sender and receiver are physically co-located. In this work we aim to provide a mechanism for proximity networking based on very high-frequency sound waves emitted and captured by the speaker and microphone found on commodity smartphones. Our approach has several benefits over existing solutions including easy deployment, lower cost for manufacturers, and intuitive security guarantees based on the physical characteristics of ultrasound signals. We implement a software based modem called "Hush," which we provide in an open source library for use in Android applications. It is practically inaudible and fast, achieving an effective transmission rate of 4900 bits per second at an ideal distance of 5cm - 20cm.**

***Index Terms*—Modems, Smart Devices, Transceivers, Filters, Device-to-device computing, Internet of Things, Acoustics**

I. Introduction

Smart mobile devices have become ubiquitous in the modern world and Internet of Things (IoT) devices are steadily growing in popularity. Users now expect a simple and easy way of exchanging small pieces of information between these devices. A key application example is sharing a URL with a friend or colleague, or turning on or off a smart light bulb [1]. The information transmitted in these scenarios may be considered sensitive; while one user may be willing to tell another their phone number, they may not want random third parties to overhear it. To support this, we propose the use of high-frequency sound (ultrasound) transmitted and received using low quality speakers and microphones like those commonly found on modern smartphones. A software based ultrasound modem is quick, easy, non-intrusive, does not require special hardware, and is highly directional with rapid attenuation in air, increasing security. All of which, make it an apt solution for this use case.

Ultrasound based proximity networking shines in the areas that competing technologies falter. First, the necessary hardware (speaker and microphone), and sample rate (44.1kHz), is relatively cheap, does not require any licensing to use, and are already ubiquitous on smart mobile devices. This means that our solution is low cost and easy to deploy. It supports the "bring your own device" paradigm; users only need to install an app on the device they already own. From a security viewpoint, ultrasound has inherently narrow propagation, and an ideal range of 10's of centimeters, which also allows for intuitive aiming of the transmission to the intended receiver. Even attackers with highly sensitive equipment will not be able eavesdrop on transmissions from a distance, especially when they are not the intended receiver, because the signal will be lost in the background noise. To further strengthen the security of our system we propose a fingerprinting technique, which allows the receiver to identify the sender of a data packet. This is possible because the sender will necessarily alter the ideal signal due to imperfections in the speaker. These alterations are consistent, and very difficult to imitate by an attacker with a different device because they are introduced at the hardware layer. Fingerprinting frees our system from requiring any pairing protocol, or prior exchange of encryption keys. Without better mechanisms for proximity networking users are forced to use imperfect alternatives, which potentially exposes their information to a variety of third parties including attackers.

We choose to implement our modem in the spectrum between 17.5kHz and 21kHz, which is extremely difficult to hear [2], [3] but is achievable by smart mobile devices. However, there are still audibility challenges, because simple carrier wave modulation manifests as audible sound. We also face several other challenges related to sample-rate and frequency response, which we detail in Section IV. We also worked hard to find appropriate features in order to implement our fingerprinting mechanism for the receiver to recognize packets sent from a specific sender. This finger printing system is novel and our system is the only, to the best of the authors' knowledge, that aims to send data via inaudible sound waves on commodity smartphone hardware for IoT applications.

The current state of the art in this domain is a system titled "Dhwani" [4], which attempts to transmit data using sound as a wireless communication medium as well. The authors use quadrature phase-shift keying (QPSK) on a 6kHz carrier achieving 2.4kbps. Altering this system to operate in the ultrasound spectrum directly fails for two reasons. First, QPSK results in a very high error rate in our target spectrum, due to the limited sample rate on commodity devices. Second, the sudden changes in phase caused from the modulation manifest as audible sound. Our system is different in that it utilizes a mechanism known as orthogonal frequency division multiplexing (OFDM), and it incorporates a sender fingerprinting mechanism to protect against replay and masquerade attacks.

Manuscript received Feb. 28, 2018; revised Apr. 16, 2018, accepted Jun. 5, 2018.

E. Novak and Z. Tang are with *Franklin and Marshall College 415 Harrisburg, Ave. Lancaster, PA. For email use: {enovak, ztang}@fandm.edu

Q. Li is with †The College of William and Mary, P.O. Box 8795 Williamsburg, VA. For email use: liqun@cs.wm.edu

We go to great lengths to ensure that our modem does not create any disruptive or irritating noises.

There are a wide variety of existing solutions that attempt to solve the proximity networking problem. The least technical (and unfortunately very common) is for users to simply tell one another the information verbally. This is slow, tedious (e.g., for URLs), and insecure due to eavesdroppers. Users may also choose traditional networking approaches such as a cloud-based file sharing apps, email, SMS messages, or similar. These approaches assume that the users have exchanged references (e.g., phone numbers) before hand, and they will expose the information to the service provider. Additionally, encryption is required to protect against eavesdroppers on the network. Other existing proximity networking technologies include near field communication (NFC) [5], Bluetooth (BT), and WiFi (ad-hoc mode / WiFi direct). WiFi direct and Bluetooth are relatively long range compared with ultrasound because they penetrate solid walls (including the floor and ceiling) and can propagate over 100ft. As such, these technologies usually require cumbersome client selection, pairing, special hardware, and encryption protocols to remain secure, which often require user involvement. Regarding WiFi, the spectrum 2.4GHz, is becoming increasingly crowded and WiFi networks in the home typically rely on a single consumer-grade network appliance that acts as a switch, router, WiFi access point, NAT device, and firewall. When all traffic must move through this device it becomes an obvious single point of failure making it an attractive target for malicious actors. For NFC, the range is impractically short ($< 2cm$ apart [6], [7]) and some security problems have been explored recently [8], [9].

We present an alternative proximity networking solution called "Hush," which operates in the ultrasound spectrum to remain inaudible. The modem operates on commodity smart mobile devices, but is intended to work on a wide variety of ubiquitous computers. Our contributions in this work are as follows:

- We propose the use of ultrasound as a means of data transmission for proximity networking on commodity, consumer hardware. We are the first to closely examine different modulation / demodulation schemes that are inaudible to humans and achieve a high data transfer rate.
- We implement and publish our ultrasound modem "Hush" as a Java library for easy use in Android applications. Our library is designed to run well on low-powered devices, and provides a socket-like interface for easy deployment by developers.
- We propose a fingerprinting mechanism that can learn and later verify the sender of data packets based on characteristics of the data signal. This helps protect users from various attacks improving the security of our system.
- We evaluate several aspects of our system including bit error rate, audible noise, and the accuracy of our fingerprinting scheme. Although Hush is sensitive to sender / receiver orientation, our effective transmission rate is 4.9kbps in ideal settings, and we achieve near perfect fingerprinting accuracy.

II. Application Scenarios

Our ultrasound modem, Hush, has been implemented and tested on commodity smart phones. However, we intend for our approach to be implemented on other IoT devices in a wide variety of scenarios. For example when giving a presentation; connecting the presenter's laptop to the projector in the room is often error-prone and generally frustrating requiring special physical connectors and specific configurations of both the display / projector and the user's computer. An alternative is a smart display equipped with an ultrasound microphone and a very simple operating system running our Hush modem. The user can then install the Hush app on their smartphone or portable computer and transmit the relatively small PDF file of their slides to the smart display via ultrasound. The slides can be projected, and the smartphone can then double as a presentation "clicker" to advance the slides using the same ultrasound technology. This sort of approach is ill-advised over WiFi, because the smart display would necessarily run a network service on an open port and is therefore vulnerable to anybody on the network (at best) or the Internet (at worst). For a large enterprise this might be an attacker in another room, or even in another building. Hush requires physical proximity, because it uses sound waves in air to transmit data.

A second application scenario is the increasingly common consumer-oriented smart light-bulb. Typically these devices are "always-on" WiFi enabled, requiring the user to have access to a properly configured WiFi network. A much better alternative is to use the Hush ultrasound modem between the user's smartphone and the smart light-bulb fitted with a microphone. This allows the light-bulb to be controlled in cases where there is no WiFi network or when the user does not have authorization to configure the WiFi network; such as in an office building, school, or other large enterprise. Furthermore, Hush is more intuitive in that communicates with devices only in the same room in an ad-hoc manner. Our fingerprinting system can also be used by the light-bulb to ignore transmissions from those other than the true owner protecting against potential attackers.

In both of these scenarios, our ultrasound modem affords the user better security and convenience. Hush does not require the setup, configuration, and maintenance of a WiFi network. NFC is completely infeasible in these scenarios, because the practical transmission range is limited to about 1cm. And, Bluetooth requires configuration (pairing) and may be overheard, or interfered with, by eavesdroppers in adjacent rooms. With the Hush modem, the user can simply walk into the room and start making use of IoT devices immediately from a comfortable distance. At the same time, some basic security is provided by the physical proximity requirement. Furthermore, with some zero explicit prior configuration our fingerprinting mechanism can allow a user to become recognizable by the device in order to differentiate them from other potential attackers. For more details please see Section VI.

III. Related Work

Audio communication has been proposed in the past and many legacy systems propose audible signals [10], [11], [12], [13]. A few works [14], [15], [16] propose ultrasound as a full competitor with radio frequency (RF) and infra-red (IR) technologies. However, they use transducers specifically designed for ultrasound, which is fundamentally different from our attempt to build an ultrasound modem on commodity smart mobile devices. These works explore modulation of ultrasound, but our work is the first to explore modulation that is both (1) intended for typical smartphone hardware, and (2) aims to be as inaudible as possible. Additionally, our work is the first to fingerprint sender devices using the raw ultrasound signal, which is useful for a wide range of applications [17].

There has been a recent proliferation in attempts to achieve a variety of proximity networking applications on smartphones using ultrasound by commercial projects [18], [19], [20], [21], [22], [23]. Unfortunately, none of these projects published technical details, so we cannot compare them to our system with much granularity. However, two of them advertise concrete bit rates; Zoosh by Narette advertises 300bps [21], and SSCConnect advertises 2.2kbps [22], both of which are much slower than our system.

There are several relevant academic works in which ultrasonic audio in air is used to transmit data. (1) In [24] the authors propose a concept called "room-area networks" in which they develop a network stack and implement ultrasound at the physical layer as a proof of concept. They simply adapt the existing 802.11a modulation scheme to achieve high throughput on commodity laptops. (2) In [11] one section of their work implements an ultrasound modem, but it achieves only 8bps. (3) Authors Hanspach and Goetz [25] re-purpose two existing audio modems by changing the carrier frequency to 18.6kHz. They incorporate these modems in malware designed to bridge traditional "air-gap" systems. They can achieve at best 20bps at a range of 19.7 meters using two Lenovo brand laptops. (4) Matsuoka et. al. [26] apply a similar OFDM based modulation scheme to our own, but are only able to achieve approximately 1kbps. (5) Roy et. al. [27] generate sounds much higher than human hearing (40kHz) which can be detected by commodity smartphone microphones thanks to non-linear combination which occurs in the microphone. They require special hardware to generate the tones at the sender side. (6) Nittala et. al. [28] propose the use of ultrasound to send signals to smartphones via existing infrastructure such as television programs and in-store audio systems. Their modulation scheme is a very simple implementation of frequency shift keying achieving 8bps. (7) Lee et. al. [29] implement a long range, low throughput ultrasound modem for smartphones as well. They face some similar challenges and achieve a rate of 16bps. (8) "PriWhisper" [30] implements acoustic transmission for key exchange only achieving approximately 1kpbs. Our system is faster than nearly all of these, it is inaudible, designed specifically to operate on commodity smart mobile devices, and implements a sender fingerprinting defense.

In the recent past, work in audio based networking was focused on making the audio tones audible but melodic and more pleasant for the user [11], [12]. However, even melodic tones are not desirable in all situations. They are also subject to background noise, which is typically more prevalent for audio in the audible spectrum. Previous work, "Dhwani" [4], emits audible noise in all of their modulation implementations (OFDM BASK, QPSK, and 8-PSK) because they operate on 1kHz of spectrum centered on 6.5kHz. Their system achieves at best 2.4kbps with 80% packet success using 8-PSK. Because our system uses near-ultrasound, it is practically silent, and faces a substantially greater technical challenge.

There has also been some recent literature on protocols that enable secure, authenticated proximity networking. EnCore [31], Smokescreen [32], BlueID [33], and SDDR (secure device discovery and recognition) [34] are all protocols for neighbor discovery. They rely on existing network mediums; Bluetooth, WiFi (ad-hoc / Direct). Our fingerprinting technique complements these systems, making masquerade and replay attacks more difficult for attackers. It is similar in premise, and underlying concept to recent work "S2M" from Chen et. al., [35]. In their work, devices are fingerprinted based on the frequency response, which is characteristic of the specific hardware used for transmission. Our system uses frequency response, but also some other key characteristics of the signal introduced by the hardware. The key difference is that S2M requires a separate, pre-determined signal to be sent, which is then checked by the verifier. Our system performs fingerprinting on the data signal itself. Which means that it is impossible for the attacker to inject packets (a key weakness of S2M) without being recognized. Several other works have selected to fingerprint devices based similarly on idiosyncrasies such as clock skew [36], and driver / firmware quirks introduced near or at the physical layer [37], [38], [39].

IV. Challenges

In this section we detail several technical challenges in implementing our system. First, we aim to transmit data while remaining inaudible. Second, we face poor frequency response in the ultrasonic spectrum. Third, the sample rate is limited and not always consistent, causing various accuracy and audibility problems. Finally, we face synchronization issues between the sender and receiver.

Inaudibility Commonly, to transfer data, the sender will produce a sine wave with frequency f and phase θ. The sine wave's frequency, amplitude, and phase can each be altered over time to encode information (modulation). This sine wave is sampled, digitally as $S = s_1, s_2, ..., s_i$ using Eq. 1, where the time t is derived from the sample rate F_s and index i, $(t = \frac{i}{F_s})$.

$$s_i = sin(2 * \pi * f * t + \theta) \quad (1)$$

The fundamental modulation schemes modify amplitude, frequency, or phase independently and are referred to as amplitude, frequency, or phase shift keying; ASK, FSK, and PSK respectively. Binary (two symbol) versions of these modulations are illustrated in Fig. 1. We quickly discovered that these modulation schemes produce audible noise, even if the carrier sine wave is above the audible frequency threshold (e.g., higher than 20kHz), due to the modulation pattern.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2848099, IEEE Internet of Things Journal

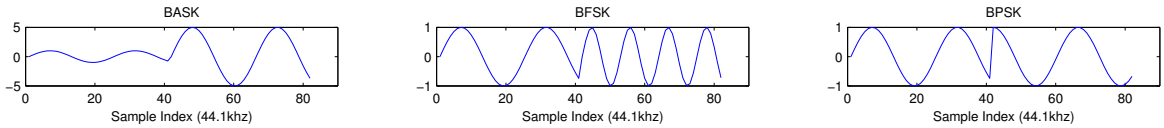

Fig. 1. Example sine wave modulated in amplitude (left), frequency (middle), and phase (right). In this example, "low" corresponds to 0 and "high" corresponds to 1 except for phase which encodes 0 as 0 and π as 1.

This can be most easily visualized in the case of ASK. In Fig. 2 we plot a BASK signal which encodes a repeating bit pattern (0, 1, 0, 1, 0, ...). The periodic amplitude changes generate an audible "artifact" wave, shown in bold, which corresponds to the bit pattern. If the bit pattern is regular, as is the case in our example, the artifact wave will be a specific frequency, corresponding to the length of the symbols. When the bit pattern is random, which is the case for our intended application, the artifact wave is similar to white noise with many random frequencies.

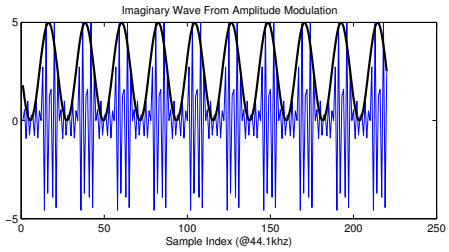

Fig. 2. Imaginary wave (black) generated from ASK

Audible artifacts are also caused by any instantaneous changes in the waveform (near sample number 41 in the plots of Fig. 1). The speaker's diaphragm physically moves synchronously with the path of the sine wave, and any discontinuity causes the speaker to produce an audible "click" noise. To achieve practically inaudible transmission, instead of encoding bits in the time domain, we focus instead on encoding information in the frequency domain, using orthogonal frequency division multiplexing (OFDM), which we detail later in Section V.

Poor Frequency Response The frequency response of cell phone speakers and microphones is highly volatile. The physical size and price of cell phones limits their speaker and microphone quality and as a result, audio signals are heavily distorted. This is compounded by the non-linear interaction between frequencies when many are present simultaneously. To demonstrate this problem, we generated frequencies, 50Hz apart in the range [18kHz - 22kHz] and [5kHz - 8kHz], one at a time and recorded the sound using two representative Android smartphones. We plotted the recording, in the frequency domain, in Fig. **??**. These experiments show that smartphones produce lower frequency sounds with greater energy and more uniformly, compared with high frequency sounds.

Sample Rate Limitations Our target devices ubiquitously have a limited sample rate for audio hardware of 44.1kHz limiting the spectrum we can use to 22.050kHz according to the Nyquist rate. Also, as previously mentioned, the hardware has difficulty reproducing tones at the top of the spectrum. While it is common knowledge that most humans can hear

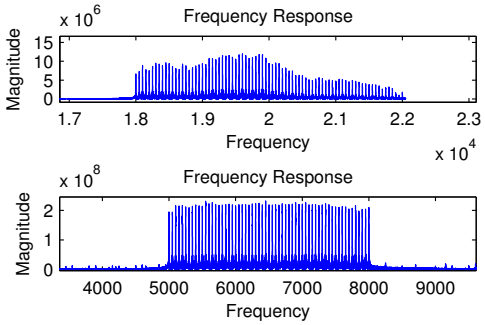

Fig. 3. Frequency response of typical cell phones.

between 20Hz and 20kHz, research shows that the threshold of perception sharply increases above 16kHz [2], [3], and that background noise in this spectrum is minimal [4], [40]. This leaves us a narrow spectrum to operate in from 16kHz - 22kHz. In this spectrum there are less than three samples period (e.g., 44100Hz / 18000Hz = 2.45 samples / period). This means that when recovering the phase, if there is even a single sample error in determining the starting point of the signal, there will be a large error in recovered phase. As shown in Fig. 4; a large phase discrepancy is apparent between samples zero and one, indicated by the ←'s in the figure.

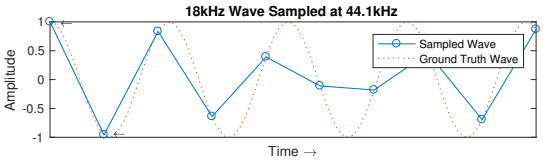

Fig. 4. 18kHz sine wave sampled at 44.1kHz.

This problem is exacerbated if the sender and receiver are not synchronized (their samples do not a-line in time) or if their sample rates are not constant. In our experiments, we find that both of these situations arise, with the sender and receiver sometimes having a relative difference of as much as 300Hz or a one sample mis-alignment after 150 samples. We discuss how to recover the phase despite this challenge in Section V-D.

Multi-path Ultrasound is more directional than low frequency sounds, and it reflects off of surfaces more easily. Because of this, signal collisions caused by multi-path may occur. Fortunately, ultrasound attenuates quickly, and the signals we send are relatively weak. Because of this, multi-path is not a significant issue.

Count	Frequency	Use	Amplitude = α	Phase = θ
5	17528.03 17571.10 17614.16 17657.23 17700.30	size		
21	17743.36 ... 18604.69	data		
1	18647.76	pilot	100%	π
26	18690.83 ... 19767.48	data		
1	19810.55	pilot	100%	π
26	19853.62 ... 20930.28	data		

TABLE I
SUB-CARRIER DATA FRAME MODULATION TABLE

V. SYSTEM DESIGN

Our ultrasound modem can be implemented on any device that has a speaker (for sending) or a microphone (for receiving) and supports a sample rate of 44.1kHz. Devices then communicate using ultrasound signals in an ad-hoc manner. Our target hardware platform is smartphones, in which these hardware requirements are ubiquitous. However, our modem can also be implemented on mobile payment stations, smart watches, IoT devices, smart home appliances, etc. The system diagram in Fig. 5 gives the components of Hush. The user inputs some information, the Hush software converts that information to an audio signal, and transmits it using the device speaker. A second device receives the signal and the Hush software decodes it. The information is then displayed to the user in the appropriate way.

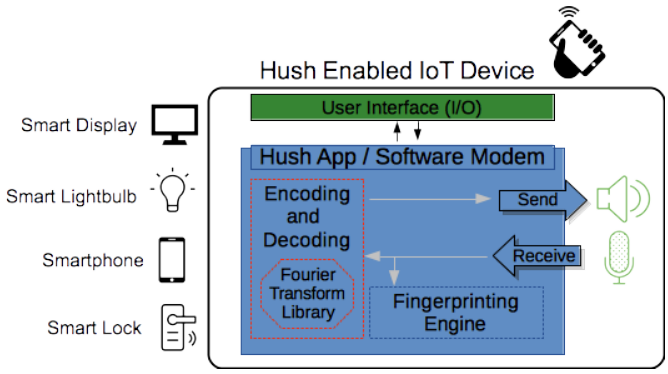

Fig. 5. Context free system diagram of the Hush software. We envision our system running on a wide variety of devices such as smart light-bulbs, smart locks, and other IoT devices.

To ease implementation, we make our modem available as a Java library for Android, which exports a HushSocket class that can be used to both send and receive data. The HushSocket sends audio signal "packets" which we detail in the rest of this section. A logical diagram of a packet can be seen in Fig. 7.

A. Signal Generation

To send data we must encode it as a sound signal. Keeping in mind the challenges mentioned previously, we design the sender to generate a linear combination of 78 "sub-carrier" frequencies from 17528.03Hz - 20930.27Hz as specified in Table I. Each sub-carrier / row in the table corresponds to a digitally sampled sine wave (see Eq.1) of that frequency. We modulate the amplitude and phase of each wave according to the data to be transmitted. All of the sub-carriers are combined into a single signal according to Eq. 2. Here S_i is a single sample at index number i, α is the amplitude, f is the sub-carrier frequency, F_s is the sample rate (44.1kHz), and θ is the phase. At the receiver side, Fourier analysis is used to recover the amplitude and phase of each sub-carrier. The values of amplitude and phase are direct encodings of the data so, at the receiver side, high amplitude indicates a "1" in the data stream, and low amplitude encodes "0" (likewise is true for phase).

$$\mathbb{S}_i = \sum_{f=17528.03kHz}^{20930.28kHz} \alpha * sin(2 * \pi * i * \frac{f}{F_s} + \theta) \qquad (2)$$

As shown in Table I, the first five sub-carriers encode a ten-bit "size" field, which is used to indicate to the receiver how many data bits are in the message. The next 21 sub-carriers encode 42 bits. Sub-carriers 18647.76 and 19810.55 are pilot or "calibration" frequencies which are always set to constant amplitude (100%) and phase (π). Details for how these are used at the receiver side are given in the Section V-C. The blocks of 26 sub-carriers after the first pilot, and second pilot are used to encode 52 bits each. The entire signal ($\mathbb{S} = [\mathbb{S}_0, \mathbb{S}_1, \mathbb{S}_2, ..., \mathbb{S}_{1023}]$) comprises a "data frame." The sender generates up to three data frames in a single packet depending on the length of the message (the size field indicates the number of bits in one packet, across all three frames).

In order to help the receiver locate the packet in time, the sender precedes each packet with a 300 sample "hail" signal. The hail signal is simply a linear chirp from 18kHz to 19kHz. Because the hail signal is known, the receiver can scan over the microphone data as it comes in, calculating the cross correlation between that and the known hail signal. When the correlation is very high, the receiver knows it has found the packet. We use a linear chirp (i.e., a sweep), because it correlates with it self best when it is perfectly aligned, and very poorly otherwise. Additionally, it is unlikely to occur in random background noise and it is nearly silent. Inevitably we will have some errors in transmission due to corruption of the signal. To combat this we implement hamming forward error correction coding (ECC) to correct single bit errors and detect multiple bit errors in each block [41]. Hamming codes allow for blocks of $n = 2^r - 1$ bits, which contain a message of $k = 2^r - r - 1$ bits, and r detection bits. Choosing r intelligently depends on the error rate, which we investigate in our evaluation and select $r = 8$.

B. Signal Quieting

We intentionally designed our modem to use an OFDM scheme, which minimizes noise because the signal remains constant for a longer period of time (1024 samples). However, there are still audible artifacts caused by a few sudden changes, which act like discontinuities. They cause the speaker to fail in it's attempt to follow the signal, and therefore create an audible

"click" sound. These discontinuities occur in four places per packet according to our design; when the hail signal starts, at the transition between the hail signal and the first frame, at the transitions between the subsequent second and third frames, and when the final frame ends. To help minimize these sudden changes, and therefore audible noise, we develop a technique based on a modification of windowing.

Windowing is a technique where a vector of coefficients between 0-1 (the window) are multiplied with a section of a target signal. The samples outside the window are discarded. Windows are typically used for analysis of a transient or impulse signal, found in a longer signal. Usually they have a bell-curve shape, based on a cosine waveform, which reduces the amplitude of the signal significantly at the end points. When applied to an audio signal this bell shape has the fortunate side effect of reducing the amplitude of artifact noises to a negligible level.

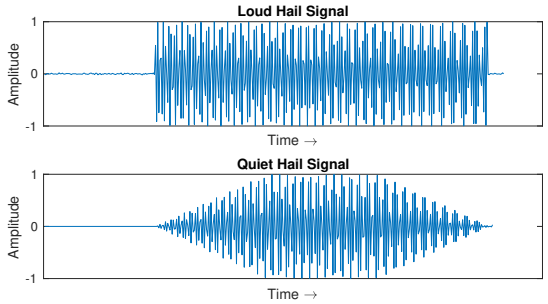

Fig. 6. Top: default noisy hail signal. Bottom: windowing technique applied to minimize instantaneous changes and reduce audible noise.

We apply a triangle window to the beginning and end of the hail signal as shown in Fig. 6. Thanks to this windowing, the transition from background noise at the start and end of the hail signal is far less dramatic, and the audible noise generated is minimal. However, windowing has a dramatic impact on the output of the Fourier transform, which disrupts the decoding of the signal at the receiver side. To combat this, we devise our own custom window for the frames. A Hann window of 20 samples is broken into two halves and the frame is extended by 20 samples, from 1024 to 1044. The Hann window coefficients are applied to the first and last 10 samples of the frame, and the middle 1024 samples of the frame are left unaltered. This allows for the quieting effect, and the receiver can simply apply the Fourier analysis to the middle 1024 samples, which remain unchanged. This is illustrated in Fig. 7.

It is important to note that 1024 samples are necessary only as an implementation detail. Using more samples would result in finer sub-carrier granularity, and we could encode more data. However, the bit-rate is not improved because of the direct correlation between number of samples (signal length in time) and sub-carrier granularity. Our FFT implementation assumes that the input will be a power of two, so we selected 1024, which means we have $\frac{44100Hz}{1024} \approx 43.07$ Hz between each sub-carrier as shown in Table I. Additionally, frames and the transitions between them also occur at a rate of 43.07 Hz, which is very near the minimum human audible frequency.

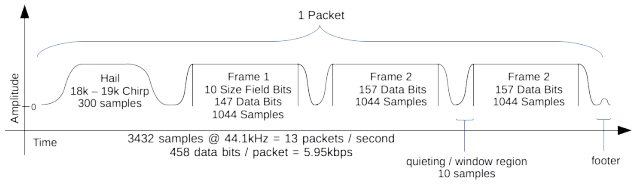

Fig. 7. Packet map consisting of the maximum three frames, and a hail signal. The theroetical throughput of our packet design (assuming no time between packets and perfect transmission) is 5.95kbps

C. Signal Decoding

At the receiver side, the microphone accepts sound signal samples continuously. Before attempting to decode a packet, the receiver must determine if a packet is present in the signal at all. To determine this we take a 100 sample window, filtering it using a custom designed high-pass FIR filter, and compute the root-mean-squared (RMS) value of the window. If the RMS is above an empirically chosen threshold, an attempt is made to decode the packet, otherwise, the window is advanced 50 samples. This method is fast enough to run on smart mobile devices (it does not compute a cross-correlation, or Fourier transform), and it processes samples 100 at a time. Yet, it is effective in detecting the presence of a packet with few false positives or negatives.

If a packet is detected, instead of advancing the window, the audio is instead treated as packet data and decoded. The first step of this process is to determine the actual starting point of the packet in the data. This previous filter and RMS approach only locates a packet with 100 samples of accuracy. The cross correlation is computed between the known hail signal and the first 1024 samples of the audio stream. This locates the hail signal in the audio to approximately one sample of accuracy. However, there may be sub-sample differences caused by syncopation between the sender and receiver. Or, in the case that the sender or receiver (or both) are not truly sampling at exactly 44.1kHz. In either case, there will be sub-sample differences between the received and ground truth signal. We use a Hilbert transform to locate the starting point of the signal, with sub-sample accuracy.

After the hail signal is found, the receiver can locate the subsequent frames at 1044 sample offsets. Each frame is windowed and a Fourier analysis is run in order to recover the amplitude; $abs(FFT_{out})$ and phase; $arctan(FFT_{out})$ values of each sub-carrier. FFT_{out} indicates the output of the faster Fourier transform algorithm, which is an array of complex values.

1) Decoding Amplitude: As mentioned previously, and is shown in Fig. 8, the amplitude of each sub-carrier is non-uniform, despite the uniformity at the sender side, due to distortion of the signal. Because of this, a simple threshold value is insufficient to decode the values accurately. To overcome this challenge, we design a technique called *adaptive amplitude recovery* which attempts to follow the trend of amplitudes. It first creates two first in first out queues, *up* and *down*, which contain the values of the previous α "up" amplitudes and "down" amplitudes respectively.

The first pilot frequency, ≈18.6kHz, is statically held at

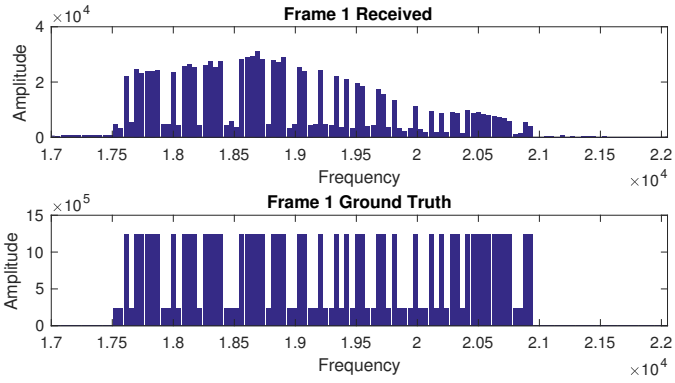

Fig. 8. Amplitude readings from the first data frame. Values are computing as the absolute value of the (complex valued) output of the Fourier Transform of the first frame of audio data .

100% amplitude. We use this reading to bootstrap our demodulator, placing the value in *up* and placing a zero in *down*. Then, each data-encoded sub-carrier amplitude is read and decoded. A reading above the threshold is decoded as "1" and placed in *up*, readings below the threshold are decoded conversely as "0" and placed in *down*. $thresh$ is defined below.

$$thresh = [(\overline{up} - \overline{down}) * \gamma] + \overline{down} \quad (3)$$

Here $\overline{X}$ is the average of the values in X. γ (where $0 \leq \gamma \leq 1$) is a parameter that determines where, between the two averages, the threshold is placed. At the half-way point, the second pilot is inserted to better calibrate to higher frequencies, which tend to be weaker overall. As the sub-carriers are processed, the queues are emptied to guarantee they always contain only the most recent α readings. In our implementation we empirically choose $\alpha = 2$, and $\gamma = 0.35$. Choosing optimal values, and exchanging them automatically, is left for future work.

D. Decoding Phase

Decoding the phase values is nearly identical to decoding amplitude values, except that the values are computed from the $arctan$ of the Fourier transform output. However, to decode the phase accurately, we must find the starting point with sub-sample accuracy at each frame. Due to the sample rate challenges outlined previously in Section IV, we cannot find the frames at exactly 1044 sample offsets. To solve this problem we use a novel technique we call *analytic phase recovery*. The sender sets the phase of the two pilot sub-carriers equal. The decoder reads these phase values ($\theta_{18.6}$ and $\theta_{19.8}$) from the frame directly, assuming the starting point is correct. It then calculates the phase at 100 sub-sample points before and after that point. At each sub-sample point, we use the sine wave formula, Eq. 1 to approximate the phase. The sub-carrier frequency f is known, so we substitute in the relative time t and solve for θ. Each time $abs(\theta_{18.6} - \theta_{19.8})$ is calculated. When the difference between the two phase readings is minimized, we have found the optimal t. Similarly, the phase values of the data-encoded sub-carriers is read using the appropriate frequency and the now known t.

Similarly to the amplitude values, the phase values are not uniform at the receiver side. Again, a simple threshold is insufficient for decoding. To account for this, we calculate the $diff = \theta_p - \theta_f$, where θ_p is the phase of the nearest pilot sub-carrier and θ_f is the phase of the target sub-carrier. We then decode each value as follows:

$$\text{if} \quad abs(diff) > \tfrac{1}{2} * \pi \quad \rightarrow \quad 0$$
$$\text{if} \quad abs(diff) <= \tfrac{1}{2} * \pi \quad \rightarrow \quad 1$$

Phase readings are always between $\pm\pi$, but $diff$ may be greater than π. If this occurs we simply set $diff = (2 * \pi) - diff$.

VI. Fingerprinting

In order to raise the bar for the security of Hush, we design and implement a prototype fingerprinting algorithm, which is inspired by the naive-bayes algorithm. Our algorithm allows the receiver of a packet to learn, and later verify the sender of that packet based on idiosyncracies introduced by the sender's hardware.

The fingerprinting algorithm will learn from a training dataset of several packets, and verify later incoming packets based on five features:

1) RMS of the raw signal - A measure of the strength or power of the signal. For audio, the "root mean squared" (RMS) of the signal intuitively maps to the average "volume" of the sound over a period of time. We included this feature because we noted that some device models were able to produce a stronger ultrasound signal.

2) Symmetry of the raw signal - A measure of the symmetry about the x-axis. Despite the sender always sending a perfectly symmetrical signal (like that shown in Fig. 6), the sender may recover a signal that drifts away from the x-axis over time becoming more positive or more negative. For ultrasound signals near the Nyquist rate the sample points will alternate above and below the x-axis as is shown in Fig. 4. Symmetry is calculated by computing the average value of each pair of points such that one of the pair is positive and the other is negative. We then compute the standard deviation of these values. A small value (near 0) indicates high symmetry.

3) Correlation of frequency response - As mentioned previously, mobile devices have poor frequency response in the near-ultrasound spectrum. This feature is computed as the cross-correlation between the received signal and a reconstruction of the ideal signal. Because the ground truth signal is not necessarily known at the receiver side, this feature is computed over the hail signal, sent at the beginning of every packet, which is constant and known. The cross-correlation is computed on the signals in the frequency domain i.e., the output from the Fourier transform.

4) Beginning largest value of the raw signal - The largest signal sample in the first 100 samples of the signal. The reasoning behind including this feature is empirical. We noticed that some devices produce a distinctive, early peak in the waveform.

5) Strength of calibration sub-carriers - As mentioned previously, the signal contains two sub-carrier frequencies that are always transmitted at 100% power with 0 phase. In practice, we note that the receiver measures these two at different strengths due to the poor frequency response as shown in Fig. 3. This is calculated as the average value of the two.

The values of each of these five features will be measured in each packet and our system will rank and weight each feature in terms of which best predicts the sender. In practice, the learning phase can occur when the user explicitly initiates it, or during normal operation in the background by assuming the first few packets are from the same (not malicious) user. In either case, several model packets are transmitted, and used to gather feature values. These values are used to create a range of plausible values, and are stored with the ID of the now authenticated user. Optionally, the user can train the system to include multiple authenticated users, assuming devices from different manufacturers.

In the verification phase, when a packet arrives (i.e., a sample), the algorithm will compute the signal's five feature values. Our algorithm is similar to naive-bays; the probability of the sample belonging to the user is determined by calculating how many of the feature measurements fall into the previously computed verification range. Each feature casts a vote based on which class's verification range the new sample falls into (the authenticated user's or "no match"). The votes of each feature are weighted and the class with the most votes is output as the prediction of the sender. The weights of each feature are then adjusted to improve the influence of the features that were part of the consensus. Finally, the system uses this prediction to verify the proposed sender of the packet.

VII. Limitations

Our modem attempts to decode phase values despite the sample rate challenges mentioned previously. In contrast, Dhwani [4] is able to decode phase modulations (without finding the precise starting point), because they're working at a much lower carrier frequency (i.e., 6kHz - 7kHz). At these frequencies, there are three times as many samples per period so the phase reading difference from adjacent samples is small and sub-sample differences are negligible. While we are able to recover the phase with reasonable accuracy, the rate of phase errors is much higher than that of amplitude errors. To address this, we implemented a second version of our modem that does not attempt to modulate phase at all, thereby reducing the theoretical bit rate by one half. In our evaluation we present the bit error rate of both versions.

Our modem is very sensitive to orientation, distance, and sender volume due to the directionality, and fast attenuation of ultrasound waves. Because of this, without the proper orientation (the speaker of the sender pointed directly at the microphone of the receiver), the bit error rate rises substantially. This impacts usability, but also provides some intuitive security. In order to receive a transmission with reasonable accuracy, an attacker must have excellent positioning in an obvious location such as directly in front of the intended receiver where they are sure to be noticed. However, in the current implementation the orientation requirements are too strict. Some related works propose systems that do not have this limitation, but which have lower data transmission rates (e.g., [25], [26]).

Our fingerprinting system identifies the sender of a packet based on idiosyncrasies introduced from the sender's hardware. Therefore, an attacker is likely able to fool the system (achieve a false positive) if they are able to obtain a device that is the same make and model as the authenticated user's device. In this way, the system is not perfectly secure, but it does increase the burden on the attacker.

VIII. Evaluation

We evaluate the bit error rate, audibility, and orientation requirements of our modem. We also measure the accuracy of our fingerprinting algorithm. Throughout our evaluation we use three smartphone models, each with different speakers and microphones; the Samsung Galaxy S7, the Google Nexus 5X, and the LG G4.

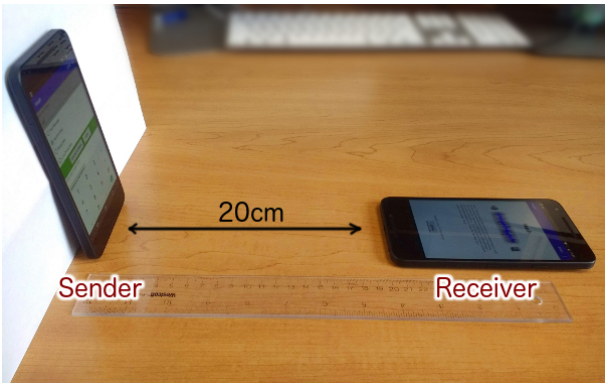

Fig. 9. Typical experimental setup. Pictured above are two Nexus 5X devices spaced 20cm apart with ideal orientation.

A. Transmission Bit Error Rate

To measure the bit error rate (BER) achieved by our modem we sent packets containing random data between each pair of sender and receiver, and measured the number of bits that were decoded correctly (before ECC). In this experiment the sender power (87.5%), distance ($20cm$), and orientation are kept at the ideal with minor alterations depending on the phones being used (e.g., the G4 is placed slightly closer, because it is a weaker sender). A typical setup for two Nexus 5X devices can be seen in Fig. 9. We sent packets repeatedly until the margin of error around the mean error rate is less than 1 bit, with a 95% confidence interval. In Fig. 10 we present the results. We can see from this figure that the BER is below 3% in almost all cases. This figure also reveals that the BER is most dependent

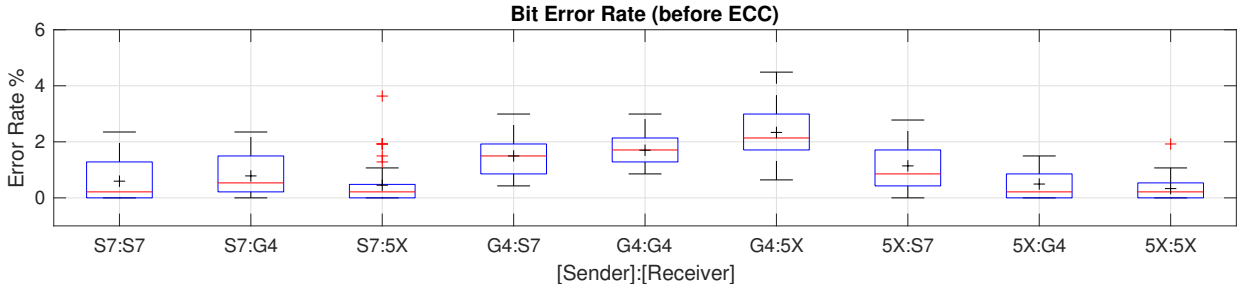

Fig. 10. Bit error rate sending random data between all nine pairs of three test devices.

Snd.	Rec.	Mean Err.	Num. Trials	No Phase-50 Trials-12150 bits Mean Err.
S7	S7	0.60%	38 *(17784 bits)*	0%
S7	G4	0.78%	42 *(19656 bits)*	0%
S7	5X	0.45%	45 *(21060 bits)*	0%
G4	S7	1.50%	40 *(18720 bits)*	0.043%
G4	G4	1.70%	24 *(11232 bits)*	0.043%
G4	5X	2.34%	73 *(34164 bits)*	0.10%
5X	S7	1.14%	50 *(23400 bits)*	0%
5X	G4	0.49%	26 *(12168 bits)*	0%
5X	5X	0.33%	20 *(9360 bits)*	0%

TABLE II
MEAN BIT ERROR RATE VARYING SENDER AND RECEIVER. "FAST" TRIALS WERE REPEATED UNTIL THE MARGIN OF ERROR IS SMALLER THAN ± 1 BIT WITH A 95% CONFIDENCE INTERVAL. ONE TRIAL = 1 PACKET = 468 BITS.

on the sender. The G4 is least capable reproducing ultrasound signals, and our prototype code was written and tested using the Nexus 5X, which performs best. The mean values (shown as "+" in Fig. 10) are summarized in Table II.

If we implement error correction coding to correct all packets with 3% error or lower, we require fourteen ECC blocks spanning thirty-three bits each. This means six detection bits in each ECC block for a total of 84 detection bits per packet. In this case we can achieve an effective rate of 4.99kbps. Based on these results, we also implemented a second version of our modem, which does not utilize phase. This reduces the error substantially, as shown in the rightmost column of Table II. But, it also reduces the theoretical bit rate by one half. For our "no phase" version, we can achieve an effective throughput 2.9kbps. However, these analyses assume uniform error distribution, which in our experiments is not the case. It may be possible to use "fire" error correction coding, which is specifically design for bursty errors, which we leave to future work.

To investigate the orientation requirements of our modem we transmit a packet from one sender to three receivers (all 5X). The first receiver has perfect orientation (shown in Fig. 9), the second is placed a few cm to the side, with the microphone pointed at the sender ($\approx 30°$ angle), and the third is placed behind the intended receiver. This is to mimic the setup an attacker might attempt. The results of this experiment are given in Table III.

Receiver		Error Rate	Error Rate (No Phase)
1	(perfect)	0.0%	0.0%
2	(30°)	6.12%	0.0%
3	(behind)	47.85%	45.32%

TABLE III
ERROR RATE WITH VARYING ORIENTATION.

B. Audibility

Using a third party Android app [42] we measured about -19dB of sound when using our modem, compared with the background noise of -50dB in a quiet office room. However, this measurement is not very insightful, because it includes ultrasound, which humans are much less sensitive to.

To evaluate how much sound from our modem is perceived by the user, we asked eight participants (mean age: 27) to listen to our modem while it transmitted random data. The sending volume was set at 60% and the experiment is repeated five times. The participants rated how much noise they heard on a scale of 1 "silent" to 100 "fire alarm" each time. The results are plotted as a histogram in Fig 11. The highest rating (21%) indicates that our modem is very difficult to hear. Several participants made comments that they "can't hear anything at all," and thought perhaps the modem was malfunctioning.

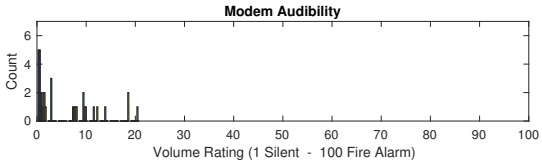

Fig. 11. Histogram of user audibility rating. Results of eight participants shown together.

While it is possible that our modem is more perceivable by young children, or animals, or that ultrasound might have unforeseen effects on users in general. A truly robust analysis of audibility must be left for future work, as these questions are largely out of the scope of this paper.

C. Fingerprint Accuracy

To evaluate the fingerprinting scheme we proposed, we gather 20 representative (low error rate) transmissions for each pairing of our three test devices. We choose a device to act as receiver, which will act as the leaner / authenticator. Then, we select a random n of the 20 transmissions from a random target sender device, which forms the training set. We test against the remaining $20 - n$ from the target device, as well as the 20 trails from each of the other two devices. The $20 - n$ trails from the target device measures the "true positive" rate, i.e., the probability that the receiver can correctly identify the target sender. The 20 trails from the other devices measures

the "true negative" rate, i.e., the probability that the receiver can correctly identify an attacker (any device other than the target sender). The results of this experiment are shown in Fig. 12.

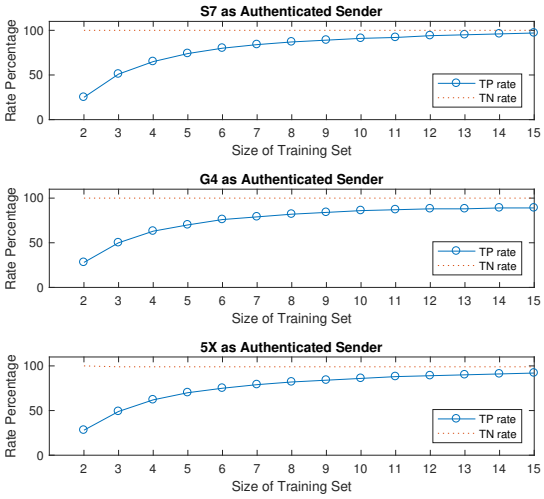

Fig. 12. Accuracy of identifying target sender (TP rate) vs. attacker (TN).

As we can see in this figure, the system's ability to reject attackers (devices other than the trained sender) is very good. However, a larger training set is necessary to correctly identify the sender (the false negative rate is high). This means the initial pairing phase must be long enough to accommodate sending fifteen training packets. Fortunately, this would take less than one second. To improve the accuracy over time correctly identified sender packets can be added to the training set.

IX. Conclusion

In this paper we present "Hush" a software modem, which utilizes very high frequency sound to send data between commodity smart mobile devices. Hush modulates ultrasound in a way that is fast, low error, and practically unnoticeable by users. Hush incorporates a fingerprinting scheme that makes it more difficult for attackers to masquerade by allowing the receiver to learn and recognize packets sent from the intended sender. We evaluate our system and show high accuracy in fingerprinting, as well as an effective transmission rate of 4.99kbps.

References

[1] C. Pereira, A. Pinto, A. Aguiar, P. Rocha, F. Santiago, and J. Sousa, "Iot interoperability for actuating applications through standardised m2m communications," in *2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pp. 1–6, June 2016.

[2] G. Elert, "Frequency range of human hearing," September 2014. http://hypertextbook.com/facts/2003/ChrisDAmbrose.shtml.

[3] K. Ashihara, "Hearing thresholds for pure tones above 16khz," *The Journal of the Acoustical Society of America*, vol. 122, no. 3, pp. EL52–EL57, 2007.

[4] V. N. P. Rajalakshmi Nandakumar, Krishna Kant Chintalapudi and R. Venkatesan, "Dhwani : Secure peer-to-peer acoustic nfc," in *Proceedings of ACM SIGCOMM 2013*, Sigcomm '13, (New York, NY, USA), ACM, 2013.

[5] "ECMA", "Near field communication interface and protocol (nfcip-1)," June 2013. http://www.ecma-international.org/publications/standards/Ecma-340.htm.

[6] T. Baker, "Up to what distance can near field communication (nfc) operate?," May 2011.

[7] Wikipedia, "Wikipedia nfc article," August 2013. http://en.wikipedia.org/wiki/Near_field_communication.

[8] J. J. Gummeson, B. Priyantha, D. Ganesan, D. Thrasher, and P. Zhang, "Engarde: Protecting the mobile phone from malicious nfc interactions," in *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '13, (New York, NY, USA), pp. 445–458, ACM, 2013.

[9] R. Zhou and G. Xing, "nshield: A noninvasive nfc security system for mobiledevices," in *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '14, (New York, NY, USA), pp. 95–108, ACM, 2014.

[10] V. Gerasimov and W. Bender, "Things that talk: using sound for device-to-device and device-to-human communication," *IBM Syst. J.*, vol. 39, pp. 530–546, July 2000.

[11] A. Madhavapeddy, D. Scott, and R. Sharp, "Context-aware computing with sound," in *UbiComp 2003: Ubiquitous Computing* (A. Dey, A. Schmidt, and J. McCarthy, eds.), vol. 2864 of *Lecture Notes in Computer Science*, pp. 315–332, Springer Berlin Heidelberg, 2003.

[12] A. Madhavapeddy, D. Scott, A. Tse, and R. Sharp, "Audio networking: The forgotten wireless technology," *IEEE Pervasive Computing*, vol. 4, 2005.

[13] M. Uddin and T. Nadeem, "A2psm: Audio assisted wi-fi power saving mechanism for smart devices," in *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications*, HotMobile '13, (New York, NY, USA), pp. 4:1–4:6, ACM, 2013.

[14] W. Jiang and W. M. D. Wright, "Ultrasonic wireless communication in air using ofdm-ook modulation," in *2014 IEEE International Ultrasonics Symposium*, pp. 1025–1028, Sept 2014.

[15] W. Jiang and W. M. D. Wright, "Wireless communication using ultrasound in air with parallel ook channels," in *Signals and Systems Conference (ISSC 2013), 24th IET Irish*, pp. 1–6, June 2013.

[16] C. Li, D. A. Hutchins, and R. J. Green, "Short-range ultrasonic digital communications in air," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 55, pp. 908–918, April 2008.

[17] D. Takahashi, Y. Xiao, Y. Zhang, P. Chatzimisios, and H.-H. Chen, "Ieee 802.11 user fingerprinting and its applications for intrusion detection," *Comput. Math. Appl.*, vol. 60, pp. 307–318, July 2010.

[18] A. Kauffmann and S. Boris, "Tone: An experimental chrome extension for instant sharing over audio," May 2015.

[19] Y. Eonnet and H. Manceron, "Tagattitude," April 2011. http://www.tagattitude.fr/en/products/technology.

[20] M. K. Evyatar Hemo and R. Lehman, "Wimbeep," 2012. https://sites.google.com/site/wimbeep/.

[21] B. Paulson, "Zoosh," 2011. http://www.naratte.com/.

[22] B. Ray, "Crafy app lets phones send data by ultrasound with speakers, mics," November 2012. http://www.theregister.co.uk/2012/11/08/ultrasonic_bonking/.

[23] V. Sokolovsky, "Illiri," July 2013. http://www.illiri.com/.

[24] P. A. Iannucci, R. Netravali, A. K. Goyal, and H. Balakrishnan, "Room-area networks," in *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*, HotNets-XIV, (New York, NY, USA), pp. 9:1–9:7, ACM, 2015.

[25] M. Hanspach and M. Goetz, "On Covert Acoustical Mesh Networks in Air," *Journal of Communications*, vol. 8, pp. 758–767, Nov. 2013.

[26] H. Matsuoka, Y. Nakashima, and T. Yoshimura, "Acoustic communication system using mobile terminal microhphones," in *NTT DoCoMo Technical Journal Vol. 8 No. 2*, 2006.

[27] N. Roy, H. Hassanieh, and R. Roy Choudhury, "Backdoor: Making microphones hear inaudible sounds," in *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '17, (New York, NY, USA), pp. 2–14, ACM, 2017.

[28] A. S. Nittala, X.-D. Yang, S. Bateman, E. Sharlin, and S. Greenberg, "Phoneear: Interactions for mobile devices that hear high-frequency sound-encoded data," in *Proceedings of the 7th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '15, (New York, NY, USA), pp. 174–179, ACM, 2015.

[29] H. Lee, T. H. Kim, J. W. Choi, and S. Choi, "Chirp signal-based aerial acoustic communication for smart devices," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, pp. 2407–2415, April 2015.

[30] B. Zhang, Q. Zhan, S. Chen, M. Li, K. Ren, C. Wang, and D. Ma, "$ssrPriWhisper$: Enabling keyless secure acoustic communication

for smartphones," *IEEE Internet of Things Journal*, vol. 1, pp. 33–45, Feb 2014.

[31] P. Aditya, V. Erdélyi, M. Lentz, E. Shi, B. Bhattacharjee, and P. Druschel, "Encore: Private, context-based communication for mobile social apps," in *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '14, (New York, NY, USA), pp. 135–148, ACM, 2014.

[32] L. P. Cox, A. Dalton, and V. Marupadi, "Smokescreen: Flexible privacy controls for presence-sharing," in *Proceedings of the 5th International Conference on Mobile Systems, Applications and Services*, MobiSys '07, (New York, NY, USA), pp. 233–245, ACM, 2007.

[33] J. Huang, W. Albazrqaoe, and G. Xing, "Blueid: A practical system for bluetooth device identification.," in *INFOCOM*, pp. 2849–2857, 2014.

[34] M. Lentz, V. Erdélyi, P. Aditya, E. Shi, P. Druschel, and B. Bhattacharjee, "Sddr: Light-weight, secure mobile encounters," in *23rd USENIX Security Symposium (USENIX Security 14)*, (San Diego, CA), pp. 925–940, USENIX Association, Aug 2014.

[35] D. Chen, N. Zhang, Z. Qin, X. Mao, Z. Qin, X. Shen, and X. y. Li, "S2m: A lightweight acoustic fingerprints-based wireless device authentication protocol," *IEEE Internet of Things Journal*, vol. 4, pp. 88–100, Feb 2017.

[36] S. Jana and S. K. Kasera, "On fast and accurate detection of unauthorized wireless access points using clock skews," *IEEE Transactions on Mobile Computing*, vol. 9, pp. 449–462, March 2010.

[37] T. Kohno, A. Broido, and K. Claffy, "Remote physical device fingerprinting," in *2005 IEEE Symposium on Security and Privacy (S P'05)*, pp. 211–225, May 2005.

[38] J. Franklin, D. McCoy, P. Tabriz, V. Neagoe, J. Van Randwyk, and D. Sicker, "Passive data link layer 802.11 wireless device driver fingerprinting," in *Proceedings of the 15th Conference on USENIX Security Symposium - Volume 15*, USENIX-SS'06, (Berkeley, CA, USA), USENIX Association, 2006.

[39] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, "Wireless device identification with radiometric signatures," in *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking*, MobiCom '08, (New York, NY, USA), pp. 116–127, ACM, 2008.

[40] P. G. Kannan, S. P. Venkatagiri, M. C. Chan, A. L. Ananda, and L. Peh, "Low cost crowd counting using audio tones," in *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, SenSys '12, (New York, NY, USA), pp. 155–168, ACM, 2012.

[41] Wikipedia, "Wikipedia hamming code article," November 2013. http://en.wikipedia.org/wiki/Hamming_code.

[42] B. M. Solutions, "decibel application." url= https://play.google.com/store/apps/details?id=bz.bsb.decibel&hl=en.

Ed Novak earned his undergraduate degree at Monmouth College in 2010. He went on to earn an M.S. and his Ph.D. in computer science from The College of William and Mary in 2016. His thesis "Security and Privacy for Ubiquitous Mobile Devices" was advised by Dr. Qun Li. He is currently beginning his third year as an assistant professor of Computer Science at Franklin and Marshall College in Lancaster, PA. His research interests include security and privacy, smart mobile devices, software engineering, and the emerging Internet of Things.

Zhuofan Tang recently earned his undergraduate degree at Franklin and Marshall College in 2018. He worked with Dr. Novak on this project. His research interests include machine learning, security and privacy of IoT devices and computer vision.

Qun Li is a professor in the Department of Computer Science at the College of William & Mary. He holds a Ph.D. degree in computer science from Dartmouth College. His research interests include edge computing, wireless networks, sensor networks, pervasive computing, and security & privacy. He is an IEEE Fellow.