mmSpy: Spying Phone Calls using mmWave Radars

Suryoday Basak[1] and Mahanth Gowda[1]

[1]The Pennsylvania State University, University Park, PA
Email: {sxb1332, mahanth.gowda}@psu.edu

***Abstract*—This paper presents a system *mmSpy* that shows the feasibility of eavesdropping phone calls remotely. Towards this end, *mmSpy* performs sensing of earpiece vibrations using an off-the-shelf radar device that operates in the mmWave spectrum (77 GHz, and 60 GHz). Given that mmWave radars are becoming popular in a number of autonomous driving, remote sensing, and other IoT applications, we believe this is a critical privacy concern. In contrast to prior works that show the feasibility of detecting loudspeaker vibrations with larger amplitudes, *mmSpy* exploits smaller wavelengths of mmWave radar signals to detect subtle vibrations in the earpiece devices used in phonecalls. Towards designing this attack, *mmSpy* solves a number of challenges related to non-availability of large scale radar datasets, systematic correction of various sources of noises, as well as domain adaptation problems in harvesting training data. Extensive measurement-based validation achieves an end-to-end accuracy of $83-44\%$ in classifying digits and keywords over a range of 1-6 ft, thereby compromising the privacy in applications such as exchange of credit card information. In addition, *mmSpy* shows the feasibility of reconstruction of the audio signals from the radar data, using which more sensitive information can be potentially leaked.**

***Keywords:* side channel attacks, mmWave radars, speech privacy**

I. Introduction

Millimeter wave (mmWave) communication technology is being increasingly adopted for next generation networking applications that require low latency and high throughput such as virtual/augmented reality [47], vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) networking [102] in autonomous driving, machine type communications in industrial IoT [9] etc. In addition to networking, mmWave technology is also becoming increasingly popular in a number of remote sensing applications in the areas of material detection, autonomous driving, precision agriculture, vibration sensing in industries, robotics, etc. [62], [45], [55], [53].

Motivated by these applications, mmWave communication has been incorporated in 5G and other networking standards, thereby increasing the proliferation of mmWave sensing and networking devices in a number of IoT applications. The technology is readily available on newer smartphones, virtual/augmented reality (VR/AR) headsets, as well as several off-the-shelf radar devices for autonomous driving applications. While other exciting applications are around the corner, this paper takes a step back and exposes a critical capability in mmWave devices that can enable a malicious adversary to passively overhear phone calls.

This paper proposes *mmSpy*, a system that uses off-the-shelf mmWave radar devices for eavesdropping the audio spoken by the remote caller during phone calls. The core intuition is that the earpiece[1] device that users listen to during phone calls generate minute vibrations in the order of 7 μm. *mmSpy* senses these vibrations by detecting the changes in phases of mmWave signals reflected from the body of the phone. This opens up the possibility of eavesdropping the audio content of the remote caller during a phone conversation. In particular, *mmSpy* can eavesdrop the contents of the audio even when the audio is completely inaudible to both humans and microphones nearby. In addition, since the audio is detected directly from the source of vibrations, *mmSpy*'s spying capabilities are immune to ambient noise, which makes the attack suitable in noisy and crowded spaces where suspicion is low. This opens up an interesting attack scenario. An attacker can eavesdrop on nearby users on phone calls, especially in a social setting like conferences, or parties and spy on users who might be seated and engaged in a phone conversation. Credit card numbers, one-time passwords, SSN numbers, etc. can be stolen within the capabilities of *mmSpy*.

Such an attack is challenging for a number of reasons (i) The vibrations are extremely small ($\approx$ 7 um) in comparison to the hardware noise floor (ii) The ramping and settling time of the frequency oscillator used to generate frequency modulated carrier wave (FMCW) signals introduces a characteristic noise pattern into phase measurements. (iii) The vibration needs to be decoupled from other ambient multipath signals in the environment (iv) Because of the hardware limitations, the sampling rate of the sensed vibration tends to be non-uniform (v) Finally, while robustness of audio/speech processing algorithms depend on large high quality training data, such large datasets are not available for our problem domain.

mmSpy exploits a number of opportunities to handle the above challenges. The opportunities listed below map to the above enumerated challenges in the same order. (i) The peaks corresponding to multipath components are tracked over time in order to identify stable reflections and ignore noisy peaks. (ii) We employ statistical error correction techniques to model the noise due to ramp/settle times in the oscillator and systematically subtract it from the phase data corresponding to vibrations (iii) The reflection from a static object such as a wall will have low variation in the phases. In contrast, the reflection from the phone will have higher variation in the phases because of high frequency audio vibrations. *mmSpy* isolates phone reflection from ambient multipath by exploiting this variation. (iv) The system parameters such as the chirp

[1]Earpiece speaker is used to listen to incoming calls, which is different from inbuilt loudspeakers. This paper focuses on eavesdropping earpiece devices whose vibrations are much smaller than loudspeakers

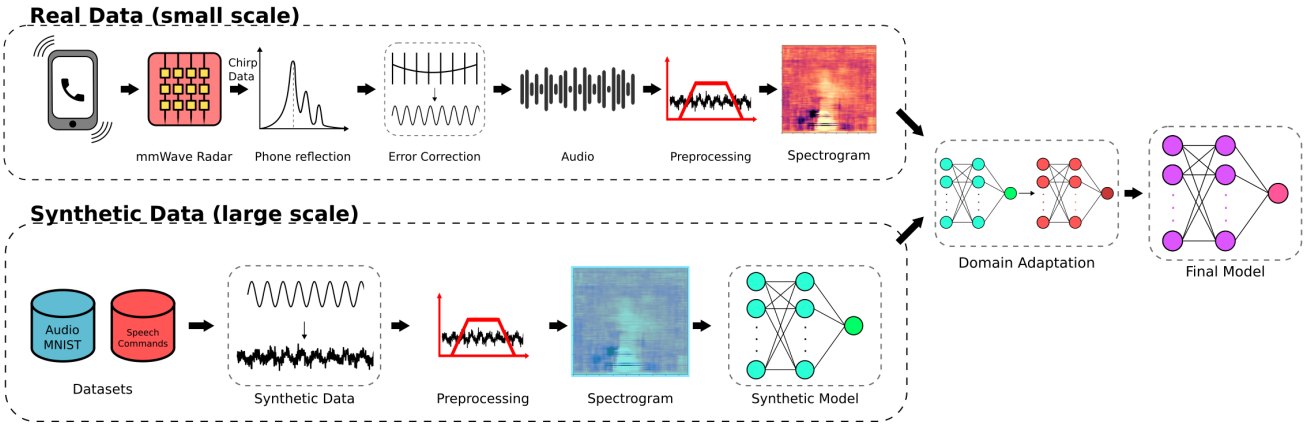


Fig. 1: Overall architecture of *mmSpy* for spying on phone calls. Synthetic training data generated from speech datasets is combined with small scale training data from real radars – this generates *mmSpy*'s audio reconstruction and speech classification models.

and frame rates of the FMCW signal, and the duty cycle etc., are carefully selected to balance the non-uniform radar sampling rate with a uniform phase sampling rate for audio. (v) We model the transformation between high quality audio to low quality vibration data. Such a modeling allows us to convert existing large-scale speech processing datasets into synthetic radar datasets. We use such synthetic datasets to train machine learning (ML) algorithms for classifying digits, keywords, as well as performing end-to-end reconstruction of audio samples. Finally, mmSpy performs fine-tuning of the models with small-datasets of sensor-data to enhance the accuracy of audio classification and reconstruction.

Prior works in the area of speech analysis with radar signals include *wireless vibrometry* [92] that can detect audio from loudspeakers using WiFi signals. Similarly, WaveEar [95] can detect speech signals using mmWave hardware based on reflection from a human throat. UWHear [90] uses UWB radios for separating speech signals from multiple loudspeakers. In contrast to these works, mmSpy differs in the following ways: (i) mmSpy shows the feasibility of eavesdropping on earpieces used during phone calls – the first such attempt to our best knowledge. (ii) Prior works focus on detecting vibrations from a loudspeaker, humans, or other sound sources which can also be heard by a colocated microphone. In contrast, by exploiting shorter wavelengths of mmWave signals, *mmSpy* shows the feasibility of detecting minute vibrations of an earpiece that cannot be heard by a microphone co-located with the radar.

mmSpy is implemented using off-the-shelf radar devices at two different frequencies – *TI AWR1843 BOOST* in 77 GHz, and *IWR6843ISK* in 60 GHz – which use FMCW signals. The attack is performed on two models of smartphones with contrasting material properties – Google Pixel 4a, and Samsing Galaxy S20. The radar sensor data is pre-processed offline with MATLAB/python modules, and fed to machine learning modules implemented in PyTorch for various applications of speech processing. The accuracy varies between $83-44\%$ over a distance of 1-6 feet for applications in digit classification and keyword recognition. The spectrograms of reconstructed audio from the spying attack match closely with the ground truth which is of critical concern from a privacy perspective.

In achieving the above reported levels of attack accuracy on smartphones with off-the-shelf radar devices, we briefly enumerate our contributions below: (i) Identification of security threats related to eavesdropping of the earpiece devices used in phone-calls with mmWave radars; (ii) Systematic preprocessing techniques for subtraction of hardware related noise and artifacts; (iii) Synthetic training data generation for training high precision machine learning models for speech classification and audio reconstruction; (iv) Domain adaptation techniques for coping up with the domain shift between synthetic training data and real sensor data; (v) Implementation and evaluation under various attack scenarios related to word/digit classification.

Fig. 1 depicts the overall architecture of *mmSpy*. A *synthetic model* in *mmSpy* is first created with large-scale synthetic training data generated using popular speech datasets. Towards handling the residual differences between synthetic and real radar data, the model is later adapted by using small-scale training datasets from real radar. The model thus generated is used for launching the eavesdropping attack. The rest of the paper will expand upon this idea.

II. Background

A. Earpiece Vibrations

Fig. 2 depicts the locations of earpieces in popular phone models such as iPhone-12, Google Pixel 4a, and Samsung Galaxy S4. The vibrations of the earpiece are much smaller than that of loudspeakers. Therefore, the users need to place their ears in direct contact with the earpiece to be able to hear the sound clearly. Because of direct physical contact, the sound waves propagate directly from one solid surface (earpiece) to another solid surface (ears), thus providing a high quality sound reception within human ears in comparison to the case where there is no physical contact with the earpiece. As a result, the leakage of the earpiece vibrations over the air is also much weaker compared to that of a loudspeaker.

However, *mmSpy* uses reflections of mmWave signals to directly track the vibrations produced by the earpiece. The

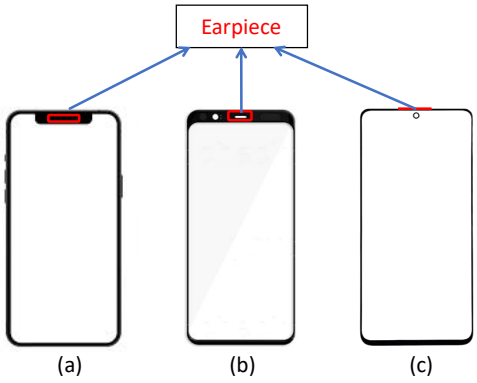


Fig. 2: Earpiece locations in popular phone models (a) iPhone 12 (b) Google Pixel 4a (c) Samsung Galaxy S20

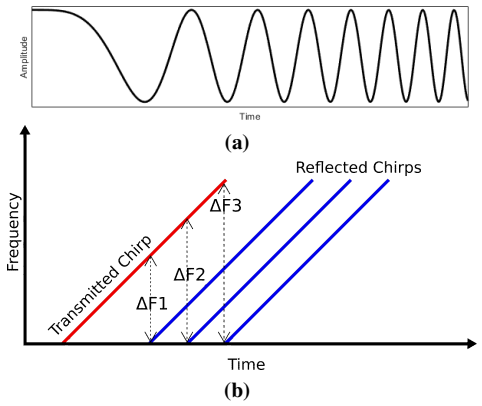


Fig. 3: (a) An FMCW signal with linearly increasing frequency (b) The reflected FMCW signals from objects in the environment maintain a constant frequency difference with respect to the transmitted FMCW signal. The distance of the reflector can be measured from this frequency difference.

earpiece vibrations will also induce vibrations in the body of the phone. *mmSpy* detects vibrations from the back of the phone which is not facing the user's ears. This enables eavesdropping of earpiece vibrations even if the leakage of sound over the air is significantly weaker. We next elaborate details on the mmWave hardware that enables this capability.

B. Overview of FMCW

mmSpy adopts an FMCW radar that works by emitting *chirps*[2]. The *chirp* is reflected back by objects in the environment of the radar and based on the time differences between transmission and reception of *chirps* and the doppler shifts, the radar can estimate the *range* (distance) of these objects as well as their velocities.

A *chirp* and the working principle of FMCW radars is visualized in Fig. 3. The signal visualized in Fig. 3(a) is a sinusoidal signal with a linearly increasing frequency which is a popular type of *chirp*. The radar modules used in this paper (TI *AWR1843BOOST* [8], TI *IWR6843ISK* [12]) employ such *chirps* with linearly increasing frequency. Since the transmitted signals are frequency-modulated signals, the reflected components will also be frequency modulated signals. However, because they are delayed, at any given point in time, there is a constant frequency difference between the transmitted and reflected *chirp* as depicted in Fig. 3(b). By computing the frequency difference ΔF between the transmitted and received chirps, the distance of the reflecting object can be computed. The below equation precisely converts the frequency difference into the *range* (r) of the object from the radar.

$$r = \frac{\Delta F}{Slope} \tag{1}$$

where $Slope$ refers to the rate at which the chirp frequency is linearly modulated.

As depicted in Fig. 3(b), multiple reflected chirps corresponding to different multipath components in the environment can be received at the transmitter. By performing an FFT operation at the receiver (called *range FFT*), different multipath components, as well as their *ranges* can be determined.

The resolution at which distances can be computed using such a method can be expressed as a function of the chirp sweeping bandwidth B as follows [83]:

$$\Delta R = \frac{c}{2B} \tag{2}$$

where c is the speed of light. In the best case scenario where the entire working bandwidth of the radar is effectively swept by a chirp, the above equation predicts a range resolution of the radar of about $3.75cm$. While this is good for a number of applications such as human activity recognition where the motion of objects are at larger scales, the resolution is not sufficient for tracking minute micrometer-level vibrations needed for capturing the earpiece vibrations during a phone call. Towards capturing a higher resolution range information, *mmSpy* exploits the *phase* of each reflected signal.

The *phase* variations can capture minute changes in motion of the reflector, as per the equation below.

$$\Delta\phi = \frac{2\pi\Delta r}{\lambda} \tag{3}$$

Given the wavelength is in the order of millimeters ($\approx 4mm$), and a typical phase noise of $0.057°$, extremely small changes in range ($\Delta r \approx 0.63$ *um*) can be tracked by exploiting the *phase* variations. *mmSpy* tracks such variations to eavesdrop on the contents of a phone call. Fig. 4 depicts extraction of continuous phase changes from the FMCW radar. A *range-FFT* operation will result in multiple peaks corresponding to reflections in the environment. Among these peaks, the peak corresponding to reflection from the phone is first isolated (Section IV-A). By measuring the phase of this FFT peak, and tracking its variations continuously over time will facilitate eavesdropping of earpiece vibrations.

C. System Parameters

mmSpy uses a commodity off-the shelf (COTS) radar to demonstrate radar-based cellphone eavesdropping. While the *phase* data can be noisy because of practical constraints,

[2]chirps are signals with varying frequency, usually increasing frequency or decreasing frequency

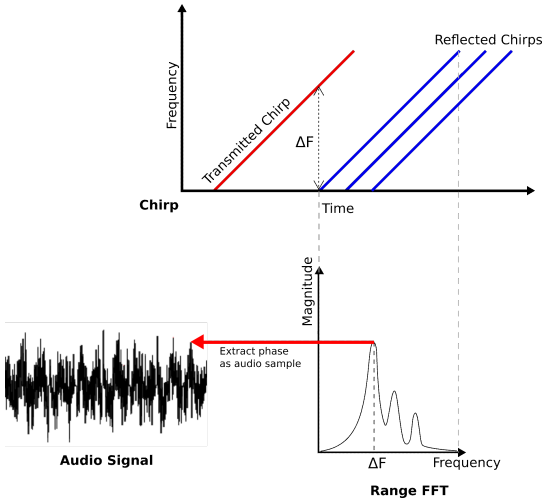


Fig. 4: A range-FFT will result in multiple peaks corresponding to objects in the environment. Tracking the phase of the peak due to phone's reflection will facilitate eavesdropping.

mmSpy chooses an appropriate set of parameters in the design space so as to facilitate high quality measurements with a high sampling rate. We elaborate below on the deliberations.

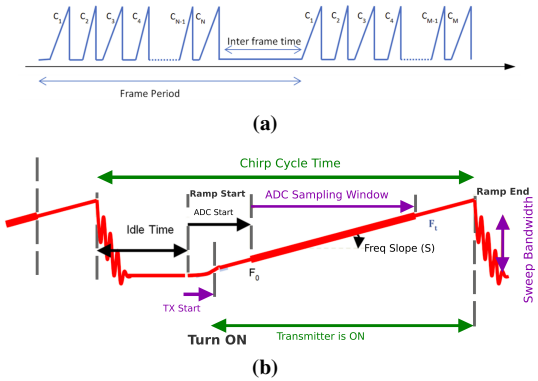


Fig. 5: (a) A series of FMCW chirps are grouped into frames (b) Key parameters involved in the cycle of a single chirp. Fig. from [16].

We begin by briefly explaining the cycle of operations in the radar for transmission of chirps. Illustrated in Fig. 5(a), The TI radar modules transmit a series of *chirps* continuously. Fig. 5(b) provides a zoomed-in view of a single *chirp*. Performing *range FFT* on each *chirp*, and isolating the reflection from the phone (discussed in Section IV-A), provides one *phase* sample per *chirp* – we call this the *phase sampling rate*. The *chirps* are grouped into *frames* as depicted in the figure. Each *frame* enforces a duty cycle of less than 100% so as to provide the radar enough time to settle down between frames. This leads to power fluctuations within the hardware due to discontinuous operation. While a higher duty cycle provides more samples, it comes with a tradeoff of noisier sensor data. Similarly, there is an inter-chirp separation time so as to let the hardware switch from the highest frequency at the end of a *chirp* back to the lowest frequency to begin a new *chirp*. The slope of the *chirp* also offers a tradeoff between sampling rate of sensor data and the hardware noise. While faster slope produces more chirps per second, the power fluctuations in the hardware can lead to low quality phase data. Keeping the above discussed tradeoffs in consideration, Appendix B expands on the details on the chosen system parameters in *mmSpy*.

III. Threat Model

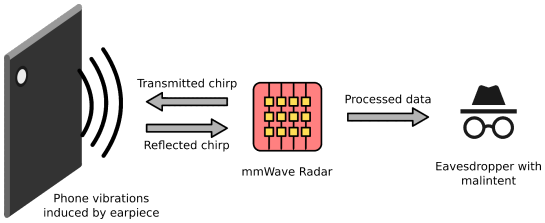


Fig. 6: Threat model of *mmSpy*. The attacker transmits mmWave signals towards a victim's phone and measures reflections. By analyzing the phases of reflections, the earpiece vibrations can be detected leading to reconstruction of the audio as well as speech classification.

Fig. 6 depicts the threat model in *mmSpy*. A malicious adversary with an mmWave radar attempts to spy on the audio contents of a phone call made by a nearby victim. Towards this end, the attacker shines mmWave signals on the victim's phone and captures the reflections. We assume that the captured reflections come from the back of the phone opposite to the side of the earpiece that faces the user's ear. By analyzing the phases of the reflection, the vibration of the earpiece device of the phone can be detected. We do not assume that the attacker has training data for domain adaptation (Section IV) from the victim's phone. The attacker generates such training data from his own phone (which is assumed to be of the same make and model as the victim's phone) for developing the speech recognition ML models (alternatives such as training from a different phone model is evaluated in Section V). *mmSpy*'s ML models are designed to perform audio reconstruction as well as speech recognition tasks from the noisy vibration data captured from reflection from the phone. Following is an example setting where such an attack is feasible. Consider a setting like a research conference or a social party. An attacker can eavesdrop on phone calls received by a nearby victim who might be seated on a chair. Given that mmWave radars can track vibrations directly from the earpiece, this is particularly effective in noisy and crowded spaces where the victim might be less suspicious of eavesdropping. Within the capabilities of *mmSpy*, the sensitive information that can be eavesdropped include credit card information, one time passwords, social security numbers, etc.

IV. Technical Modules

A. Isolation of phone reflection:

As discussed in Section II, a given *range-FFT* window will include reflection from the phone as well as multipath reflections from other objects in the environment. We face two main challenges in isolating the phone reflection: (i) Several noisy peaks show up in the range-bin which do not correspond to multipath reflections. (ii) In addition to the noisy peaks, there will be peaks corresponding to static reflectors in the environment such as walls and furniture. Towards better isolation of signal of interest from the above sources, *mmSpy*

tracks consistent peaks across successive frames. Since the noisy peaks do not consistently appear at a given distance, they are eliminated. Fig. 7 shows an example where the phone reflection is consistently tracked over time. In addition

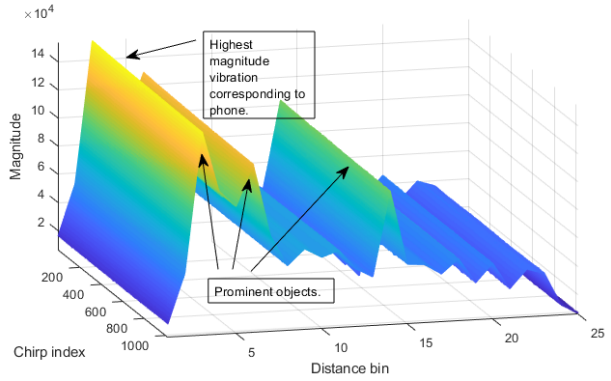


Fig. 7: Tracking of FMCW peaks over time helps eliminate noisy peaks. The phase data corresponding to the peak from phone's reflection is used to eavesdrop the audio.

to phone reflections, reflections from other objects in the environment can be seen in the figure. However, the *phase* of the reflection from the phone would oscillate (due to vibrations from the audio) whereas phase from other reflectors will not oscillate. By exploiting this property, *mmSpy* designs a shallow convolutional neural network based model to first classify reflections due to audio vibrations and reflections from static reflectors such as walls. The classifier provides a high accuracy of 99.4%, thus facilitating elimination of static reflectors like walls, furniture etc. Once the peak corresponding to the reflection from the phone has been identified, the *phase* of this peak is used for reconstruction of the earpiece audio as well as performing speech classification tasks (more details in Appendix B). Given that the receiver has four receiver antennas, the phase values are averaged across all four antennas to minimize the Gaussian noise in the extracted audio.

B. Statistical error correction:

The vibrations induced on the body of the phone by the earpiece is, by nature, of a very low magnitude. As a consequence of this, the variation in the phase of the range-FFT peaks is also very low and noise can supersede the magnitude of the phase changes that are useful for vibration detection.

The mmWave radio transmits and receives chirps in a discontinuous manner in the form of frames. We observe that at the beginning of every frame, there is a spike in the magnitude of the phase values (as shown in Fig. 8 (a)). Additionally, there is also a continuous noise component that fluctuates more smoothly with time. In order to eliminate this, we assume that within each frame of chirps, a smooth enveloping component exists and we eliminate it.

This is done by estimating the fluctuation within a frame using a polynomial of degree 2. To avoid the effect of the spikes at the beginning of each frame, we also eliminate the first two data points received in a frame. Since the frame size in *mmSpy* is 128, we effectively work with 126 points within each frame to offset the fluctuations.

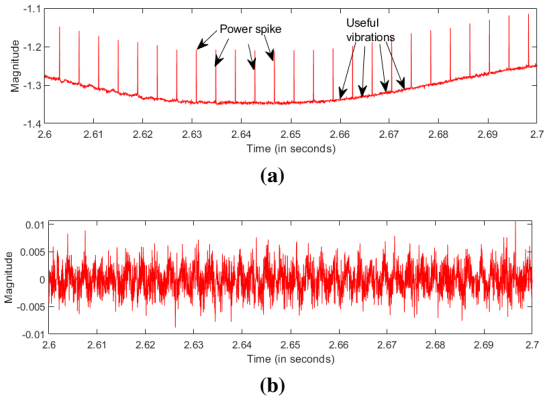


Fig. 8: Statistical error correction: (a) Noisy phase data (b) Phase data after elimination of noise.

If a frame is represented as (X, Y), where X is the index of chirps and Y is the magnitude of the phase extracted from the chirps, then the smooth fluctuation is estimated using the following model:

$$\hat{Y} = a_0 X^2 + a_1 X + c \tag{4}$$

where X^2 refers to each element of X being raised to a power of 2. This is similar to a linear regression of order 2 on (X, Y). Here, the parameters a_0, a_1 and c are the estimated parameters in the polynomial fitting model. Once $\hat{Y}$ has been estimated, the corrected frame is obtained by subtracting $\hat{Y}$ from Y:

$$Y' = Y - \hat{Y} \tag{5}$$

where Y' is the corrected frame.

The effect of the error correction is demonstrated in Fig. 8. The spikes are eliminated, and the signal is zero centered, as we expect audio signals to be, due to the elimination of the fluctuating components.

C. Preprocessing and Signal Filtering

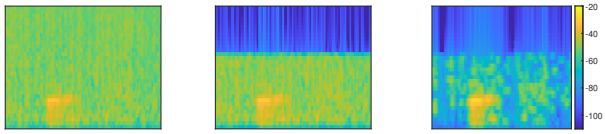
We perform a number of preprocessing and filtering techniques on the extracted raw audio from the radar as outlines below.

Bandpass Filtering: *mmSpy* tracks sound via material vibrations. It is known from literature that materials attenuate vibrations at higher frequencies [41], [86], [85]. Therefore, the spectrum measured at higher frequencies mostly consists of noise which can be eliminated by applying a low pass filter at 2000 Hz [73]. Also, the fundamental frequency of the voiced speech of a typical adult male will vary from 85 to 180 Hz, and that of an adult female from 165 to 255 Hz [87], [34]. Thus, we apply a high pass filter at 80 Hz to eliminate the DC offsets and low frequency noise without affecting speech recognition.

Spectral Subtraction: We perform background noise elimination using spectral subtraction techniques popular in speech processing [38]. At a high level, the average signal spectrum and the average noise spectrum are first estimated and then

subtracted from each other, which is shown to eliminate additive stationary noise [88], [85], [78].

Fig. 9 depicts an audio signal before and after preprocessing techniques. Evidently, the voice part of the signal has been enhanced in comparison to hardware noise.



(a) (b) (c)

Fig. 9: Preprocessing and noise subtraction (the word spoken is "two") – (a) Raw signal (b) After bandpass filtering (c) After spectral subtraction.

D. Synthetic training data generation:

Speech processing algorithms have to be robust to speaker patterns, dialects, gender, etc. However, developing robust models require large scale training data accumulated over years across diverse users. While large scale datasets are available for speech and vision domains, the unfortunate lack of availability of similar radar datasets makes it challenging to develop robust models. Towards minimizing the overhead of training data collection, *mmSpy* generates synthetic training data to bootstrap the training of ML models.

The synthetic data is created from the existing datasets (AudioMNIST [36] and Speech Commands [91]) using two main operations: scaling and noising. In order to scale the existing datasets, radar samples were collected and moments of speech were identified. Whenever words are being spoken, the phase magnitudes vary in the range of -0.024 and 0.024 (after statistical error correction). We assume that the noise in the sensor data is normally distributed and estimate the parameters of the noise distribution from samples where no sounds are played on the phone. The mean and the standard deviation of the noise distribution are 0 and 0.0035 respectively. Thus, the audio datasets were scaled in the range of [-0.024, 0.024] and random noise sequences with a normal distribution $N(0, 0.0035)$ were added to create synthetic training data.

Although the synthetic data attempts to match the distribution of real radar data well at a high level, there will still be residual differences in distribution. Towards eliminating such differences, *mmSpy* later performs *domain adaptation* based on a small set of real training data, thus achieving a sweet-spot in the trade-off between training data overhead and robustness of the ML model.

E. Audio Reconstruction:

Towards reconstruction of the original audio from the noisy radar data, we design a redundant convolutional encoder decoder (RCED) architecture [79] as illustrated in Fig. 10. The audio and radar samples are downsampled to a sampling rate of 8kHz – this sampling rate is adequate to capture audible frequencies from human speech. The input X_i to the network are the mel-spectrograms of 1-second audio samples. The dimensions of the spectrograms are 128×81. We refer to each column of this input matrix as a *segment*. Towards generating an enhanced *segment* at time t, the network accepts input *segments* from times t to $t-7$, thus accepting an input of size 128×8, and producing an output of size 128×1. This allows the network to exploit temporal locality in performing the enhancement. Also, the network consists of skip connections that help in training and convergence [46], [61], [29]. The network outputs one *segment* at a time, and after passing through the entire input, it produces an output matrix O of size 128×73 [3]. After performing the enhancement, we exploit masking techniques [51] as an alternative enhancement option. While the enhanced output O eliminates noise, it may distort the voiced segments of the audio as well. The masking can potentially help eliminate such distortions. After performing the enhancement with RCED network, the output spectrogram O is divided into 8 evenly spaced frequency ranges in the mel scale. Within each frequency range, a threshold is decided adaptively based on Otsu's method [67]. Based on the thresholding, a 0/1 binary mask is computed from spectrogram O. The 0/1 masked values are further smoothed based on Gaussian blurring [72] with a kernel of size 3×3 resulting in a masking matrix M with values between 0 and 1. The enhanced output O' is then calculated as $O' = I \odot M$. We compare the performance of both O, and O' in Section V.

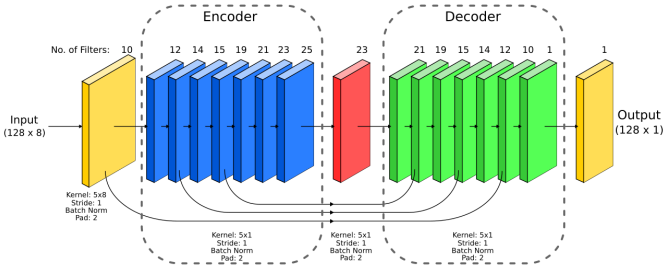


Fig. 10: Architecture of audio reconstruction network.

Loss Functions and Optimization: The loss function used to train the audio reconstruction model is the mean-squared error (MSE) loss function. If X_i and Z_i are the i^{th} input and output spectrograms respectively, the MSE is defined as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n} ||X_i - Z_i||_2^2 \tag{6}$$

Since the inputs and outputs are spectrograms, the MSE loss function is used to make the input and output spectrograms resemble each other as much as possible.

The Adam optimizer was used to minimize the loss function with a learning rate of 0.001 and L2 regularization loss with weight decay of 10^{-5}.

[3] Due to the overlapping windows used in creating spectrograms, this creates a truncation of approximately 5.48% of the final time domain output with respect to the size of the input, which we ignore in this paper.

Hyperparameter Selection: The hyperparameters were the number of skip connections between the encoder and the decoder sections, the learning rate, the L2 regularization factor. The optimal set of hyperparameters were selected using a grid search: all combinations of skip connections (that were dimensionally compatible) between the encoder and decoder were tried, the learning rate and L2 regularization rate were varied in the set $\{0.001, 0.005, 0.01, 0.05\}$.

Bootstrapping the Training: The synthetic data generated as described in Section IV-D and the true data samples corresponding to each synthetic sample were treated as the input-output pairs (X_i, Z_i), to train a *synthetic model*. The goal of the *audio reconstruction model* is thus to denoise and appropriately scale the noisy inputs to resemble the true outputs.

Domain Adaptation: The *synthetic model* is adapted using a small sensor data sample (5% of the size of the synthetic dataset for Audio MNIST, and speech commands). We perform fine-tuning of the model where we simply update the weights of all parameters with real sensor data. The amount of domain adaptation data was sufficient to achieve convergence even though none of the layers were frozen. Based on a cross-validation approach, we verified that such a fine-tuning approach provided us best results for this architecture instead of performing domain adaptation only on a few layers and freezing the rest.

F. Speech Recognition via Classification:

Towards performing speech recognition, we develop models based on convolutional neural networks that take spectrograms as inputs and estimate class labels corresponding to digits and speech-commands as outputs. The high level architecture of the classification model is depicted in Fig. 11. Similar to the network for audio reconstruction, skip connections are exploited for benefits in training and convergence.

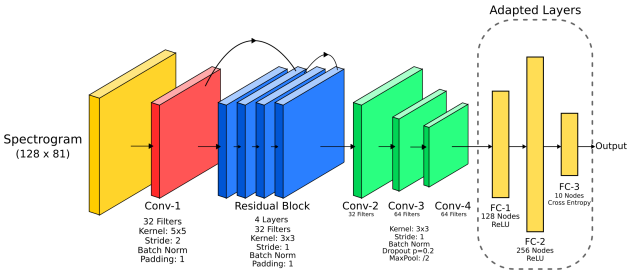


Fig. 11: Architecture of speech classification network.

Loss functions and optimizations: The model is trained using a cross-entropy loss function. The cross entropy loss function is commonly used in classification problems. It is given as:

$$\text{CE} = -\sum_{i=1}^{N}\sum_{c=1}^{M} y_{i,c} log(p_{i,c}) \tag{7}$$

where N is the number of data samples, M is the number of classes, $y_{i,c}$ is 1 if sample i belongs to class c and 0 otherwise, and p is the predicted probability that sample i is of class c.

Bootstrapping the Training: The synthetic data that was generated as described in Section IV-D and the labels corresponding to each synthetic sample were treated as the input-output pairs (X_i, y_i), to train a *synthetic model*. The goal of the classifier is to best estimate the word that is being spoken, given the spectrogram of the vibration (sensed using the mmWave radar) as the input.

Hyperparameter Selection: The hyperparameters include learning rate, L2 regularization factors, kernel sizes for convolutional layers, dropout rates, the number of resnet blocks, and the number of nodes per fully connected layer. The above parameters were varied using a grid search as follows: learning rate in the set of $\{0.001, 0.005, 0.01, 0.05\}$, square kernel sizes $\{3, 5, 7\}$, number of resnet blocks $1-3$, number of filters per convolutional layer in the resent blocks as $\{64, 128, 256\}$, number of convolutional filters in the deep convolutional layers as $\{64, 128, 256, 512\}$, the number of nodes per fully connected top layer as $\{128, 256, 512\}$. We use randomized cross-validation to tune the hyperparameters for the model, and run multiple cross-validation programs on a campus GPU cluster concurrently.

Domain Adaptation: In order to adapt the *synthetic model* to the sensor data, the last layers (indicated in Fig. 11) are retrained using sensor data as inputs and the corresponding true data samples as outputs. All the layers excepting the last few layers are frozen so that their weights do not change when the model is adapted. This is done so that the model can learn the same representation as the *synthetic model* by relearning only the last layers. Such a strategy is popular in other domain adaptation and transfer-learning applications [60], [74], [44]. Approximately 5% of synthetic data is used for domain adaptation for both AudioMNIST and speech commands datasets.

V. Evaluation

Implementation: The experimental setup is depicted in Fig. 12. *mmSpy*'s frontend includes Texas Instruments *AWR1843BOOST* [8] and *IWR6843ISK* [12] mmWave radars as introduced in Section II operating in the spectrum of 77 GHz and 60 GHz respectively. Operating with a FMCW bandwidth of $1798.92GHz$ (Appendix B), we use the DCA1000EVM [17] platform to extract samples at 10 Msps, and obtain reflections from the phone vibrations. We use Samsung Galaxy S20 (S20) and Google Pixel 4a (Pixel) phone models in our evaluation. The phases of the reflections from phone vibrations are used to extract audio content as well as train the ML models in *mmSpy* for speech recognition. We capture the reflections from the back of the phone opposite to the side of the earpiece. The ML model is implemented with PyTorch [80] packages and the training is performed on a desktop with Intel i7-8700K CPU, 16GB RAM memory, and NVIDIA Quadro RTX 8000 GPU.

Datasets: We validate the attack capability of *mmSpy* with two diverse speech recognition tasks: (i) We use the AudioMNIST dataset for validating a task in digit recognition. AudioMNIST [36] dataset consists of 30000 audio samples of spoken digits

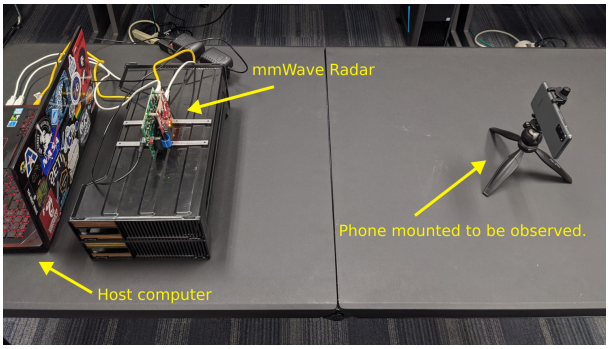


Fig. 12: Experimental setup in *mmSpy*. Off the shelf mmWave radar device is used to detect earpiece vibrations from a smartphone.

(0-9) of 60 different speakers consisting of 48 males and 12 females from all age groups. Each audio sample is less than one second long, captured in a controlled environment at a sampling rate of 8kHz. The AudioMNIST dataset is a popular benchmark for testing several techniques in the literature of speech recognition including an attack on the accelerometer sensor [33], [48]. (ii) Towards validating a task in recognition of words, we use the speech command dataset [91]. The dataset consists of 38546 samples of the following speech commands: ['down', 'go', 'left', 'no', 'off', 'on', 'right', 'stop', 'up', 'yes']. This dataset is completely anonymous and does not come with any information about age groups, genders, etc. Additionally, this dataset was crowdsourced and prepared so it includes samples from phone, laptop and tablet microphones. Each sample is converted to a 16kHz WAV file and is 1-second long.

Training Data: About 90% of samples from the datasets described above were converted into synthetic radar data (as discussed in Section IV-D) to bootstrap the training process. This includes data from all speakers, regardless of gender, age, etc. A *synthetic model* is first created, which is later adapted with small sets of real sensor data as elaborated next.

Data for Domain Adaptation: The data used for domain adaptation is approximately 5% of the size of the synthetic training data. We play the audio samples corresponding to this data on the smartphone and record the radar measurements. This generates labelled training dataset with radar recordings and their respective audio classes. We used two phones of each of these models – Google Pixel 4a and Samsung S20 – a total of four smartphones. This allows us to perform the domain adaptation and testing on different phones. We use an app, Stealth Audio Player [18] that plays the audio contents of the domain adaptation data on the smartphone's earpiece.

Test Data: In order to test the model, separate training, validation, and test sets are collected. Testing is done on a different smartphone (victim's phone) than the one from which training data for domain adaptation was generated (attacker's phone). The domain adaptation dataset was roughly split into 80:20 ratio for cross-validation while ensuring that the train and test data comes from different phones. There are no samples in common between training, testing, and the domain adaptation dataset. Similar to the domain adaptation above, we use Stealth Audio Player for playing audio contents of the test data on the earpiece device.

Metrics of Evaluation: We consider the following two metrics of evaluation. (i) For the audio reconstruction model, we report the *reconstruction loss* between the recovered audio from the radar and the original audio played on the phone. A MSE error is used to quantify this. (ii) For speech recognition with digits and speech commands dataset, we report the *classification accuracy* (or simply the *accuracy*), which is the ratio of number of correct classifications over the total number of test cases. In addition to the above, we provide qualitative results such as spectrograms and an audio demo. We now present results from a systematic measurement study.

Terminology: Since we consider multiple phones, multiple datasets, and multiple frequency bands in this paper, the number of different combinations can be exhaustive. Therefore, we choose the following subset of combinations that provide a good representation of the variation across different factors. Accordingly, we use the following terminology to represent these cases. (a) "S20 (77 GHz)" – results from Samsung S20 at 77 GHz with AudioMNIST. (b) "S20 (60 GHz)" – results from Samsung S20 at 60 GHz with AudioMNIST. (c) "Pixel (77 GHz)" – results from Google Pixel 4a at 77 GHz with AudioMNIST. (d) S20 (Sp, 77 GHz)" – results from Samsung S20 at 77 GHz with Speech commands dataset.

Qualitative Reconstruction Results: Figures 13, 14 depict the spectrograms from a qualitative reconstruction of the audio. The y-axis of the spectrograms varies from 0 to 4KHz. Representative samples from each class are presented for both AudioMNIST and speech commands datasets. The raw capture from the sensor (top row), as well as *mmSpy*'s reconstruction of the original audio from the raw sensor data is shown. Second row shows the output before masking whereas the third row shows the output after masking (discussed in Section IV-E). The bottom row shows the spectrograms of ground truth audio. Evidently, *mmSpy*'s reconstruction agrees well visually with the ground truth. While the raw sensor data looks mostly noisy, *mmSpy*'s audio reconstruction is able to highlight the key spectro-temporal trends in the audio resulting in a good recovery of the original audio.

Audio Demo: Sample audio files from the raw sensor, *mmSpy*'s reconstruction from the raw sensor data, as well as the ground truth is included in the following anonymous url [5]. Headphones are recommended for listening. While the raw sensor data is incomprehensible, evidently, an adversary can roughly decipher the contents of the ground truth audio from *mmSpy*'s reconstruction. *mmSpy* uses Griffin-Lim algorithm for reconstructing audio from spectrograms [52].

Quantitative Reconstruction Results: Fig. 15 depicts the MSE error of audio reconstruction. The spectrograms are normalized within a range of [-1,1] for a uniform comparison. As depicted, the difference in MSE between the enhanced and the true audio is lower than the MSE between the raw sensor data and the true audio. Fig 15(b) shows the MSE averaged over 1-6 ft for various settings. This shows the effectiveness

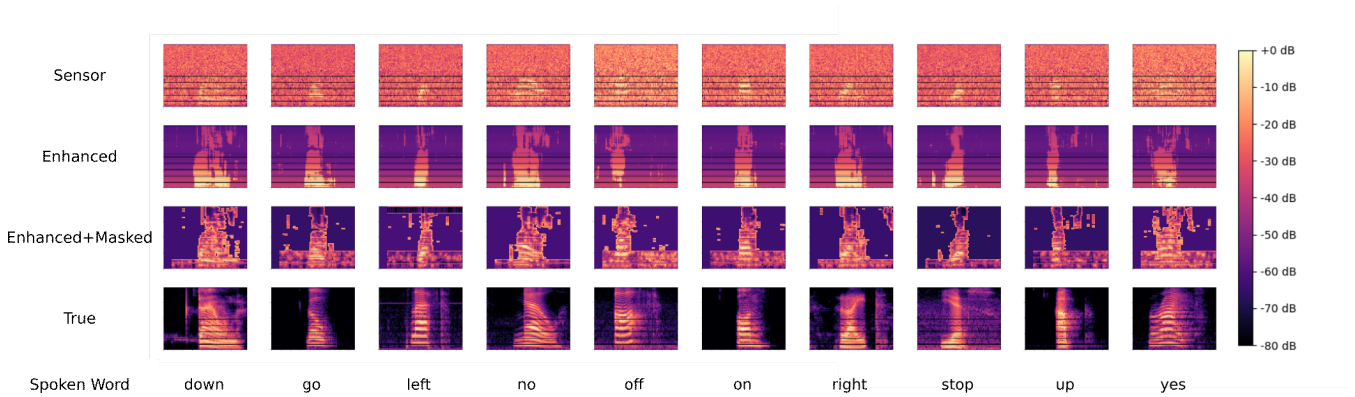


Fig. 13: Qualitative Results (at 3ft range): Speech commands. While the raw sensor data (top row) is noisy, *mmSpy*'s reconstruction (second row) is able to extract the key spectro-temporal trends in the noisy sensor data. The masking ideas (third row) allows focusing on distortion free voiced components from the input. The enhanced outputs looks visually similar to the true audio spectrograms (bottom row).

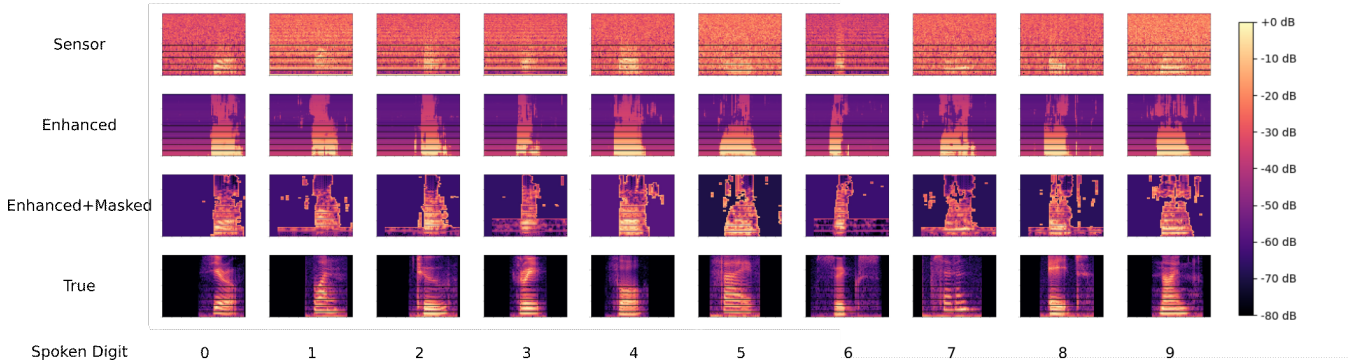
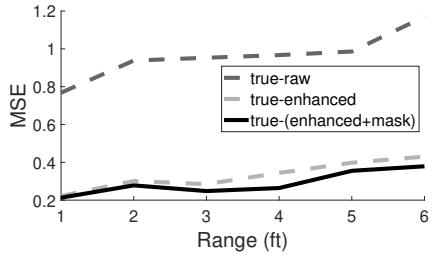
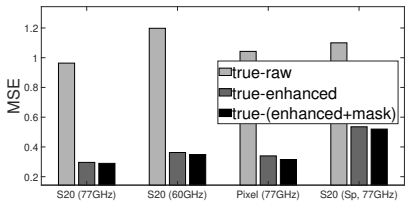


Fig. 14: Qualitative Results (at 3ft range): Audio MNIST. Similar to speech commands, the *audio reconstruction* model is able to extract the key spectro-temporal trends from noisy sensor data.

of the ML algorithms in *mmSpy* in reducing the MSE.



(a)



(b)

Fig. 15: Reconstruction error (a) MSE vs Range (S20, 77 GHz) (b) Average MSE across settings.

Power vs Range: Fig. 16 depicts the power levels of sound vibrations as a function of the distance of the phone from the radar device. The power levels are measured in dB in reference to the noise power levels when there is no sound being played on the earpiece device. As expected, the power levels drops linearly as a function of distance. At a distance of $6ft$ ft, the power starts getting closer to noise levels. Beyond this, accurate detection of the phone reflection becomes hard. The variation is consistent for different phone models (Pixel vs. S20) as well as different frequencies (77 vs. 60 GHz). The power levels with speech commands is slightly lower than AudioMNIST mainly due to a corresponding lower quality (volume) of the data in comparison to AudioMNIST data.

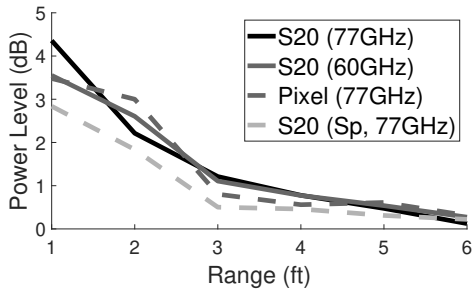


Fig. 16: Power Levels vs Range

Speech Recognition Accuracy vs Range: Table. I depicts the accuracy of *mmSpy* as a function of range for different phone models and frequency range. Evidently, the accuracy only degrades gracefully over distances upto 6 ft which suggests the potential for a successful attack under conditions identified in the threat model in Section III. The performance is consistent across multiple phone models and frequency bands.

The performance with speech commands is slightly lower than

Setting	Distance					
	1ft	2ft	3ft	4ft	5ft	6ft
S20 (77 GHz)	83.33	73.10	65.09	60.66	50.14	47.99
S20 (60 GHz)	78.35	70.05	62.37	60.11	50.91	49.92
Pixel (77 GHz)	80.11	70.94	66.30	58.33	49.09	46.60
S20 (Sp, 77 GHz)	69.37	63.81	60.74	56.70	48.62	44.56

TABLE I: Accuracy vs distance under different settings

the performance with AudioMNIST data which follows the trend in power levels observed in Fig 16. The confusion matrices (Fig. 29), precision, and recall values (Table III) for a representative setting is depicted in the appendix.

Accuracy vs Users: Fig. 17 shows the accuracy as a function of different users. The results are averaged over the entire range of $1-6ft$ where the power level related to noise varies between $4.4-0.2dB$. These results are only based on AudioMNIST dataset since the speech command dataset is a crowd-sourced dataset that does not have classes organized by users. Given that the model has been trained from a diverse

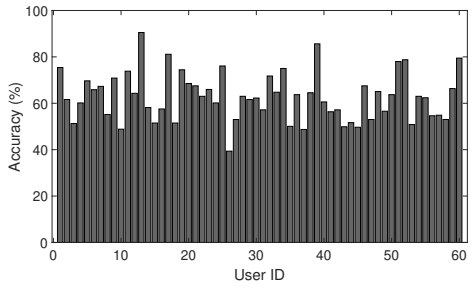


Fig. 17: Accuracy vs Users

distribution of users including males and females, the model is overall robust across a variety of users. We notice that the variation in accuracy across users is roughly correlated with the volume of their voice. The accuracy is also consistent across different genders with a 66.39% and 63.02% accuracy for male and female users respectively.

Performance in the 60 GHz spectrum: Table I also depicts the performance at 60 GHz frequencies in comparison with the 77 GHz spectrum. Evidently, the performance is consistent across both frequency spectrums. This is mainly because the propagation path loss does not change much between the two frequencies. Therefore, we observe a similar trend in SNR as well as the accuracies across both spectrums.

The Role of Domain Adaptation: Fig. 18 depicts the breakup of the gain in accuracy due to domain adaptation. The results are averaged over the entire range of $1-6ft$ where the power level varies between $4.4-0.2dB$. *mmSpy* trained with synthetic radar data helps bootstrap the process of training. While the average accuracies ($\approx 30\%$) with synthetic data is a modest start, *mmSpy* boosts the accuracy by adapting the model with small scale training data from real sensor. Evidently, with only 5% of real sensor data in comparison with original source of training dataset, the performance of *mmSpy* is substantially enhanced resulting in accuracy levels of $58-69\%$.

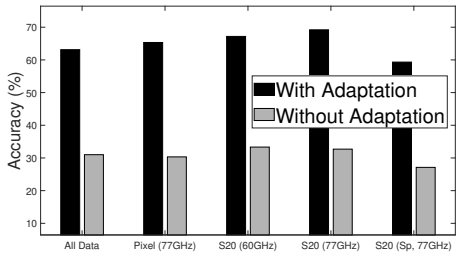


Fig. 18: While synthetic data bootstraps the training, domain adaptation substantially boosts the accuracy with small real training data.

Earpiece Signal Levels and Performance under Noisy Setting: Fig. 19 depicts earpiece power levels measured by a high fidelity microphone under two settings: (i) Ambient noise in an indoor lab, approximately estimated at $32dB$ with respect to a complete silence. (ii) Loudspeaker playing white noise with approximately $58dB$ relative to silence, thus simulating a crowded setting. The measurements were conducted using a high fidelity microphone model Zoom H1 [27]. For each case, we report the overall power levels (earpiece audio + ambient/external noise) in comparison with ambient/external noise levels when the earpiece is silent. Evidently, the earpiece

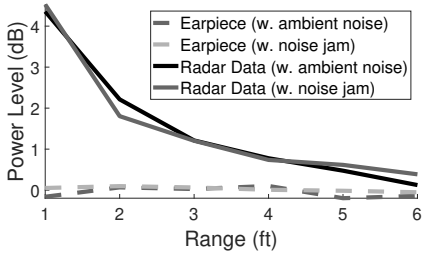


Fig. 19: Earpiece vs radar power levels in normal and noisy setting

audio is feeble and becomes easily buried under the noise floor. The power levels when the earpiece is playing is close to 0dB relative to ambient power levels, thus it is hard to distinguish whether the earpiece is playing sound or silent. The figure also depicts the performance of radar under both conditions. Evidently, the existence of external noise does not interfere with the signal strength of detection by the radar since the radar picks up the vibrations directly from the source of the vibration. In addition, Fig 20 shows samples of spectrograms from testing in the noisy environment with the loudspeaker. The microphone spectrum mainly consists of white noise

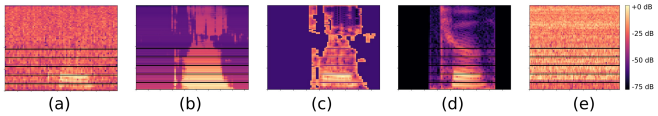


Fig. 20: Audio spectrograms (range = 3ft) from (a) Raw radar data (b) Reconstructed audio by *mmSpy* before masking (b) Reconstructed audio by *mmSpy* after masking (c) Ground truth (d) Microphone co-located with the radar. *mmSpy*'s spectra closely match the ground truth whereas a co-located microphone only detects noise.

whereas the detection from *mmSpy* matches closely with true audio. *mmSpy* detects vibrations from the source (earpiece), thus free of interference from ambient sound.

Robustness to Electromagnetic Interference: We evaluate the robustness with respect to four kinds of interference (i) Interference in the microwave spectrum as generated by software defined radios USRP N210 [24] in 2.4 GHz spectrum with a bandwidth of 5 MHz. This can emulate interference due to a number of real world electronic devices such as microwave Owens, Bluetooth streaming, Zigbee sensors etc. (ii) Interference in the 2.4 GHz and 5 GHz spectrum using off the shelf WiFi hardware (MacBook Pro 15) that generates video traffic. (iii) Interference in the mmWave spectrum using network devices based on 802.11ad protocol with Netgear XR700 router [26] and a MG360 network adapter [14] that generate video traffic. (iv) Interference in the mmWave spectrum caused by other radar devices (IWR6843ISK). In all cases, the interferer was co-located with the radar so as to measure the performance under the most challenging case. Fig. 21 summarizes the results. The results are taken from IWR6843ISK radar placed at a distance of 3ft from the phone ("S20 (60 GHz) setting"). As expected, the microwave spectrum effectively does not have any influence on the mmWave radars since they operate in different frequency bands. On the other hand, even though the mmWave interference can happen in the same frequency band as the radar, our experiments reveal that this does not affect the accuracy in any significant way. This is because of the following reasons (i) The networking protocols use traditional modulation schemes such as OFDM [65], whereas radar uses FMCW. Because of the difference in modulation, the OFDM or other non-FMCW signals will have less interference on a FMCW radar that primarily latches onto *chirps*. (ii) Another FMCW radar ceases to have any interference. The lack of clock synchronization will create an interference peak at the radar at a different position than the reflection from phone. This is automatically eliminated by the static multipath elimination algorithms in *mmSpy*. Moreover, automatic filtering at the hardware level can typically happen even for a small clock offset. This observation is consistent with the documentation by Texas Instruments [13].

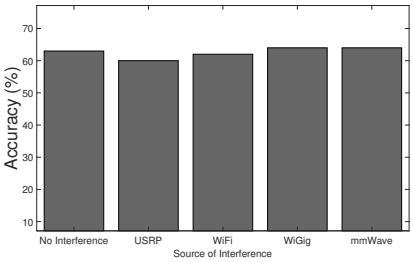


Fig. 21: Accuracy under various interference settings (range = 3ft)

Train/Test split across Phone Models and Frequency Bands: We discuss the feasibility of training and testing on different brands of phones in Appendix A.

Effect of Hand Coverage of Phone: We have been able to extend the evaluation to include humans holding the phone in the hand (as if engaged in a casual conversation) as depicted in Figure 22. The distance between the phone and the radar is approximately 3ft. Our results indicate that a sufficient amount of information still exists despite two interfering factors: (i)

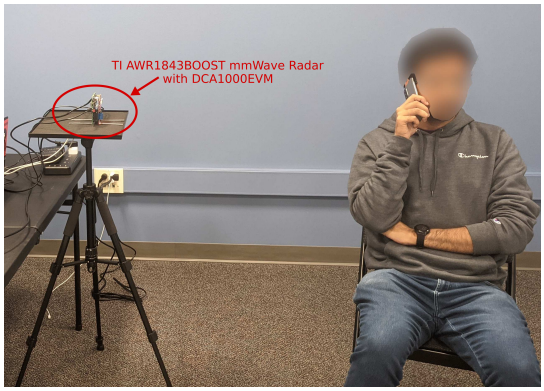
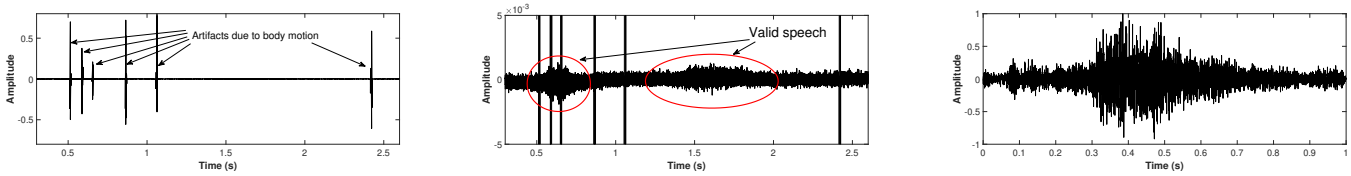


Fig. 22: A test subject holding the phone at a distance of 3 ft from the radar device.

Partial coverage of the body of the phone by the human hand. (ii) Any micro-motion (due to breathing, heartbeats, muscular twitches, etc.) in the human body which interferes with the audio sensing. Figure 23a shows the raw audio captured when the phone is held in the hand, with the audio content on the phone being "four". The artifacts due to body motion and vibrations are evident from the figure. Figure 23b shows the zoomed-in version of Figure 23a where the audio contents are now visible. We apply a simple threshold to eliminate the body motion artifact (caused due to micro-motions), as well as interpolation to fill out the gap after eliminating the body motion artifacts. The recovered audio is depicted in Figure 23c. We process the recovered audio with the classification model presented in Figure 11 (from the paper submission). The accuracy under "S20 (77 GHz)" setting (the dataset being AudioMNIST) is 58.8%. In contrast, when the phone was held on a tripod, the corresponding accuracy at the same distance was 65.09%. Similarly, for the case of "S20 (Sp, 77GHz)" (the dataset being Speech Commands), the accuracy with hand coverage is "52.74%", whereas when the phone was held on a tripod, the corresponding accuracy was "60.74%".

Experiments on Longer Speech Sentences including Songs and Music: We have been able to capture multi-word sentences, and even music, in order to demonstrate the capabilities of *mmSpy*. The performance of the audio reconstruction model in Fig. 10 is independent of the length of the audio. Therefore, we can readily use the model to extract audio even if the audio includes multiple words or sentences. We have used the network to extract audio from actual sentences including speech, music, and song. A few examples of spectrograms as well as the corresponding sound recordings (headphones are recommended to listen to the recordings) are included in this document, for three categories: (i) Speech – "I have a dream .. " speech by Dr. Martin Luther King Jr. in Figure 26 (ii) Song with background music – "Twinkle Twinkle little star .. " in Figure 27, and (iii) Music (Turkish March) in Figure 28. We believe the speech content is evident from the recordings.

Cost of Model Training: Training the classifier using synthetic data takes 10.29 minutes on average. Since the domain adaptation is done on a smaller set of real examples, the adaptation for the classifier takes 43.8 seconds on average.



(a) (b) (c)

Fig. 23: Hand Coverage Experiments (a) Body motion artifacts can be seen in the raw audio (b) Zoomed in version of the raw audio. The recording is here [7] (c) The recovered audio after body motion filtering. The recording is here: [6].

After the network is adapted, the inference per sample takes 15.3ms on average. Training the enhancer took longer since each sample is converted into multiple inputs – the training time using synthetic data on an average is 17.2 minutes. The adaptation of the enhancer took longer as well – 2.17 minutes on average, and finally, the inference per spectrogram takes 54.7ms. All evaluation was done on the desktop whose configuration is specified in the implementation subsection.

VI. Related Work

Sensing Applications with RF: There is a lot of recent interest in using WiFi-like communication devices for RADAR-like sensing applications in addition to more conventional applications in communication and networking. LiquID [45] identifies permittivties of liquid materials by measuring the slow-down and attenuation of UWB RF signals. RF-EATS [55] can sense food materials in containers by measuring reflection of backscattered RFID stickers attached on the container. RF-avatar [101] shows capabilities of beyond-the-wall 3D imaging using RF signals. While the above works use microwave frequencies, with the proliferation of 5G, there has been a lot of recent interest in using mmWave frequencies for sensing applications. mSense [93] can classify upto 21 liquids by measuring reflection of mmWave signals. mmVib can detect vibrations for classifying machines as well as monitoring their health in a number of industrial IoT applications [62]. Pointillism [35] can detect objects such as cars for applications in autonomous driving. Osprey [81] uses mmWave technologies for geometry sensing. Osprey estimates the depth of tire by utilizing concepts of synthetic aperture radars to create an image of the tire thread being placed over tires of cars. By identifying anomalies in the tire image, debris, and the wear and tear is detected. In contrast to the above applications *mmSpy* exploits the high precision sensing capabilities of the small wavelength of mmWave spectrum for exposing a security vulnerability. WaveSpy [66] spies apps running on a system by analyzing reflections from its screen. Material properties such as permittivity change with on-screen color patterns used by apps. This manifests as SNR/phase changes in reflected mmWave signals. In contrast to detecting material properties, *mmSpy* analyzes vibrations of a known material.

Speech Sensing with RF: Acoustic eavesdropping of loudspeakers have been shown in [92], where phase variations of radio frequency (RF) reflections in the microwave frequencies collected from a large antenna array are exploited to detect digits. In a similar spirit, UWHear [90] uses high resolution UltraWideBand (UWB) RF reflections for detecting multiple speakers in the environment, and shows the feasibility on a problem on digit classification. WaveEar [95] can detect speech signals using custom mmWave hardware based on reflection from a human throat. More recently, RadioMic [77] uses mmWave to detect speech signals from loudspeakers, humans, and objects. In contrast to these works, *mmSpy* shows the feasibility of eavesdropping earpiece devices used in phone calls. While loudspeaker or human throat vibrations can be stronger, thus can also be eavesdropped by a co-located microphone, the earpiece vibrations are very minute and inaudible to a microphone co-located with the attacking radar device. Nevertheless, *mmSpy* exploits small wavelength of mmWave signals to show the capability of audio reconstruction as well as speech classification.

Eavesdropping with cameras and lasers: Works in [41], [42] detect sounds played in a room using camera. In particular, sound waves induce vibration in objects (paper bags, bottles etc). By capturing such vibration patterns using a high speed camera, feasibility of recovering sound is shown even from the outside of a sound-proof room. In contrast to the above works, which detect stronger vibrations from loudspeakers, *mmSpy* detects minute vibrations from an earpiece device. Similarly, laser microphones have been popularly used for eavesdropping in a passive manner. However mmWave antennas are much smaller in size in comparison to laser microphones thus making them easier to conceal. Additionally, the presence of laser microphones is detectable [4], while mmWave signals can conceal themselves within ambient mmWave signals. Given that mmWave based 5G is a popular communication technology, this allows the adversary to conceal themselves among ambient mmWave signals. Lamphone [73] can eavesdrop acoustic vibrations that are are already in the air by analyzing vibrations of a light bulb. However, the sound is also audible to a colocated microphone near the bulb. In contrast, *mmSpy* detects weaker sound sources by picking up vibrations directly from the source, even if it us inaudible to a colocated microphone near the mmWave hardware.

Motion Sensor based Attacks: Gyrophone [69] detects the speech content from an external loudspeaker (subwoofer) using smartphone gyroscope sensors placed on the same surface (for example, shared table). Classification of 11 digits (0-9 and "oh") is shown feasible. Speechless [30] shows the sensitivity of smartphone accelerometers to eavesdrop on speech content from a loudspeaker source sharing the same surface as the phone (subwoofers, laptops etc). Spearphone

[31] performs gender and speaker classification, and detection of keywords by spying on smartphone speech content with builtin accelerometers. AccelEve [33] proposes an attack in a similar setting to Spearphone. Apart from gender and speaker classification, they perform digit and alphabet classification. AccelWord [100] shows the feasibility of detecting the wakeup keywords of voice commands such as "Okay Google", and "Hi Galaxy" using accelerometers. PitchIn [59] shows the feasibility of eavesdropping ambient speech by fusing data from multiple non-acoustic sensors (accelerometers, gyroscope, geophone etc) with low sampling rate. In contrast to such works that exploit motion sensors, *mmSpy* launches a remote attack with mmWave devices.

Attacks on mobile sensors: Ghosttalk [64] shows the capability of injecting fake data into analog sensors by directly inducing voltages into the circuitry by an external RF excitation, thus critically compromising IoT systems including heart monitors. DolphinAttack [98] shows the feasibility of injecting inaudible voice commands to attack voice assistants. Mole [89] uses a smartwatch accelerometer to spy on the contents of a user's typing. S3 [49] detects drawings on a tablet using an apple pencil by exploiting variations in magnetic fields sensed by the magnetometer. Accelerometer sensors are also known to reveal passwords as entered on the touchscreen of a phone [76]. The smartphone magnetometers are even shown to be capable of identifying the operating systems and the pattern of applications in a nearby desktop by monitoring the spinning of hard-drives which are made of magnetic materials [37]. In contrast, *mmSpy* performs an attack on spying speech contents based on radar reflections.

Domain adaptation: Transfer-learning based domain adaptation is popular in vision and speech processing. For example, AlexNet model [63] pretrained on ImageNet database [43] was fine-tuned for classifying images in the medical domain[103], remote-sensing [60] and breast-cancer [74]. Similarly, a pre-trained BERT language model [44] was fine-tuned for tasks such as text-summarizing [99], question answering [82] etc. This significantly reduces the burden of training for a new task. In a similar spirit, we use pre-trained model from synthetic radar data. While this provides a good enough *synthetic model* to begin with, we adapt the model with real radar data. Noted in Section IV, our domain adaptation trains only a few layers such that it significantly decreases the overhead of training.

VII. Discussion, Limitations, and Future Work

Eavesdropping a User under Mobility: Results in Fig. 23 show that enough information for the attack exists despite partial coverage of the phone by the human hand. However, not being able to attack a user who is in motion (such as walking) is a limitation of the current work. While sufficient vibration information still exists, the motion of the user might create interference which needs to be eliminated. We have some preliminary ideas for canceling body motion based on emerging recent works. Wistress [56] uses self-similarity matrices to identify and cancel out artefacts that are caused by small muscular movements to extract heartbeat signals. We will explore such opportunities in the future.

Relevance of the Attack in Context of 5G Applications: Table II outlines applications (both current and future) that reply on 5G technology. We believe the attack is relevant in the context of applications outlined in the table.

Application	Frequency Band
Autonomous Driving	77-81GHz [50], [68], 76GHz [102], [15]
Industrial IoT	77-81GHz [62]
Healthcare	76GHz [96], 77-81GHz [56]
5G Communication	60GHz [75], [19], [2]
Augmented and Virtual Reality	77-81GHz [97], [54], 60GHz ([47], Google Soli)
Remote Sensing	60GHz [94]
Smart Cities	[2], [3], [1]

TABLE II: Use of mmWave bands in various applications.

Defense: The vibration sensor on the phone can be used to produce noisy vibrations [84] such that the accuracy of vibration detection in *mmSpy* using the phases of the reflections is reduced. Similar to white box adversarial attacks on machine learning models [39], we can generate minimal noise using vibrations that is enough to confuse *mmSpy*'s models, while still having negligible impact on user experience. Another possible defense against *mmSpy* is to surround the end of the earpiece that is not facing the ears with a vibration dampening material. For example, materials such as q-pads, or borosilicate paints are commonly used in the music industry to eliminate unwanted vibrations [25]. Evaluation of the above ideas for defense would be a part of our future work.

Automatic Speech Recognition: While *mmSpy* demonstrates the feasibility of the attack on isolated speech recognition, we plan to extend to recognition of continuous speech. Automatic speech recognition (ASR) models based on LSTM, attentions, and language modeling [40] are popular in continuous speech recognition where the boundaries between successive words can be blurred. While training such deep learning based models requires an extensive amount of datasets, we plan to adopt a procedure similar to synthetic training data proposed here to bootstrap the training process.

VIII. Conclusion

This paper shows the feasibility of eavesdropping phone calls by detecting minute vibrations produced by the earpiece device used in phone calls using mmWave radars. While the sensor data is very noisy, *mmSpy* proposes a range of techniques from statistical noise correction, machine learning based modeling, as well as domain adaptation to develop robust models for speech recognition with low overhead of training. Extensive measurements demonstrate the feasibility of the attack. The proliferation of off-the-shelf mmWave devices both for 5G networking as well as in sensing applications makes this attack of critical concern in the context of speech privacy.

IX. Acknowledgement

We are grateful to the reviewers and the anonymous shepherd for feedback. This research was partially supported by NSF grants CNS-2008384, and CNS-1956276. We also thank Ankush Mishra for help with some of the experiments, and Trent Jaeger for feedback on the paper draft.

REFERENCES

[1] 5g drones: everything you need to know. https://www.5gradar.com/news/5g-drones-take-to-the-skies.

[2] 5g use cases: 31 examples that showcase what 5g is capable of. https://www.5gradar.com/features/what-is-5g-these-use-cases-reveal-all.

[3] Alba iulia smart city pilot project. https://sustainablecities.eu/transformative-actions-database/?c=search&action_id=ixgrulm3.

[4] An/avr-2 laser warning system. https://fas.org/man/dod-101/sys/ac/equip/an-avr-2.htm.

[5] Audio demo of mmspy. https://www.dropbox.com/sh/jppc7pg1881y51a/AAD9cHvCRWtXt__MFCko96eha?dl=0.

[6] Audio recording after compensation of body signals. https://www.dropbox.com/s/3m9xop64ja59jz0/4_23_43.wav?dl=0.

[7] Audio recording polluted by body signals. https://www.dropbox.com/s/hyc4xr62e4av5uy/main_body_vib.wav?dl=0.

[8] Awr1843 single-chip 76-ghz to 81-ghz automotive radar sensor evaluation module. https://www.ti.com/store/ti/en/p/product/?p=AWR1843BOOST.

[9] Boosting smart manufacturing with 5g wireless connectivity. https://www.ericsson.com/en/reports-and-papers/ericsson-technology-review/articles/boosting-smart-manufacturing-with-5g-wireless-connectivity.

[10] I have a dream speech, enhanced recording. https://www.dropbox.com/s/ndwff7slqwx55ig/i_have_a_dream.wav?dl=0.

[11] I have a dream speech, raw recording. https://www.dropbox.com/s/oyiuv4ne4jzp6fv/i_have_a_dream.wav?dl=0.

[12] Iwr6843isk. https://www.ti.com/tool/IWR6843ISK.

[13] Managing interference in fmcw radar systems. https://training.ti.com/managing-interference-fmcw-radar-systems.

[14] Mg360 millitronic. https://millitronic.com.tw/mg360/.

[15] O2 partners with trl testbed to bring 5g-powered driverless cars to london. https://www.5gradar.com/news/o2-partners-with-trl-testbed-to-bring-5g-powered-driverless-cars-to-london.

[16] Programming chirp parameters in ti radar devices. https://www.ti.com/lit/an/swra553a/swra553a.pdf?ts=1618486896136&ref_url=https%253A%252F%252Fwww.google.com%252F.

[17] Real-time data-capture adapter for radar sensing evaluation module. https://www.ti.com/tool/DCA1000EVM.

[18] Stealth audio player - play audio through earpiece. https://play.google.com/store/apps/details?id=com.appsbyusman.stealthaudioplayer&hl=en_US&gl=US.

[19] Top use cases for 5g technology. https://www.intel.com/content/www/us/en/wireless-network/5g-use-cases-applications.html.

[20] Turkish march, enhanced recording. https://www.dropbox.com/s/x63by49yfxoys1h/turkish_march.wav?dl=0.

[21] Turkish march, raw recording. https://www.dropbox.com/s/zv0phre4fn3pr62/turkish_march.wav?dl=0.

[22] Twinkle twinkle little star, enhanced recording. https://www.dropbox.com/s/o375a03z31eimwb/twinkle_twinkle_little_star.wav?dl=0.

[23] Twinkle twinkle little star, raw recording. https://www.dropbox.com/s/4i1wtkdyg8kl42k/twinkle_twinkle_little_star.wav?dl=0.

[24] Usrp n210 software defined radio (sdr). https://www.ettus.com/all-products/un210-kit/.

[25] Vibration: Origins, effects, solutions. https://www.gcaudio.com/tips-tricks/vibration-origins-effects-solutions/.

[26] Xr700 — nighthawk pro gaming router. https://www.netgear.com/support/product/XR700.aspx.

[27] Zoom zh1 h1 handy portable digital recorder. https://www.amazon.com/Zoom-ZH1-Portable-Digital-Recorder/dp/B003QKBVYK.

[28] What is 5g: Everything you need to know about 5g: 5g faq, May 2021. https://www.qualcomm.com/5g/what-is-5g.

[29] ADALOGLOU, N. Intuitive explanation of skip connections in deep learning. *https://theaisummer.com/* (2020).

[30] ANAND, S. A., ET AL. Speechless: Analyzing the threat to speech privacy from smartphone motion sensors. In *2018 IEEE Symposium on Security and Privacy (SP)* (2018), IEEE, pp. 1000–1017.

[31] ANAND, S. A., ET AL. Spearphone: a lightweight speech privacy exploit via accelerometer-sensed reverberations from smartphone loudspeakers. In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks* (2021), pp. 288–299.

[32] ASHWINI, A. Everything you need to know about iot prototyping, Mar 2020. https://medium.com/swlh/everything-you-need-to-know-about-iot-prototyping-e4ad2739bc6a.

[33] BA, Z., ET AL. Learning-based practical smartphone eavesdropping with built-in accelerometer. In *Proceedings of the Network and Distributed Systems Security (NDSS) Symposium* (2020), pp. 23–26.

[34] BAKEN, R. J., ET AL. *Clinical measurement of speech and voice*. Cengage Learning, 2000.

[35] BANSAL, K., ET AL. Pointillism: accurate 3d bounding box estimation with multi-radars. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems* (2020), pp. 340–353.

[36] BECKER, S., ET AL. Interpreting and explaining deep neural networks for classification of audio signals. *CoRR abs/1807.03418* (2018).

[37] BIEDERMANN, S., ET AL. Hard drive side-channel attacks using smartphone magnetic field sensors. In *International Conference on Financial Cryptography and Data Security* (2015), Springer, pp. 489–496.

[38] BOLL, S. Suppression of acoustic noise in speech using spectral subtraction. *IEEE Transactions on acoustics, speech, and signal processing 27*, 2 (1979), 113–120.

[39] CARLINI, N., ET AL. Audio adversarial examples: Targeted attacks on speech-to-text. In *2018 IEEE Security and Privacy Workshops (SPW)* (2018), IEEE, pp. 1–7.

[40] CHAN, W., ET AL. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2016), IEEE, pp. 4960–4964.

[41] DAVIS, A., ET AL. The visual microphone: Passive recovery of sound from video.

[42] DAVIS, A., ET AL. Visual vibrometry: Estimating material properties from small motion in video. In *Proceedings of the ieee conference on computer vision and pattern recognition* (2015), pp. 5335–5343.

[43] DENG, J., ET AL. Imagenet: A large-scale hierarchical image database. In *IEEE CVPR* (2009).

[44] DEVLIN, J., ET AL. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[45] DHEKNE, A., ET AL. Liquid: A wireless liquid identifier. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services* (2018), pp. 442–454.

[46] DROZDZAL, M., ET AL. The importance of skip connections in biomedical image segmentation. In *Deep learning and data labeling for medical applications*. Springer, 2016, pp. 179–187.

[47] ELBAMBY, M. S., ET AL. Edge computing meets millimeter-wave enabled vr: Paving the way to cutting the cord. In *2018 IEEE Wireless Communications and Networking Conference (WCNC)* (2018), IEEE, pp. 1–6.

[48] ERICSSON, D., ET AL. Adversarial representation learning for private speech generation. *arXiv preprint arXiv:2006.09114* (2020).

[49] FARRUKH, H., ET AL. S3: Side-channel attack on stylus pencil through sensors. *arXiv preprint arXiv:2103.05840* (2021).

[50] GARCIA, K., ET AL. Robust traffic and intersection monitoring using millimeter wave sensors.

[51] GERKMANN, T., ET AL. Spectral masking and filtering, 2018.

[52] GRIFFIN, D., ET AL. Signal estimation from modified short-time fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing 32*, 2 (1984), 236–243.

[53] GUAN, J., ET AL. High resolution millimeter wave imaging for self-driving cars. *arXiv preprint arXiv:1912.09579* (2019).

[54] GUAN, J., ET AL. Through fog high-resolution imaging using millimeter wave radar. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), pp. 11461–11470.

[55] HA, U., ET AL. Food and liquid sensing in practical environments using rfids. In *17th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 20)* (2020), pp. 1083–1100.

[56] HA, U., ET AL. Wistress: Contactless stress monitoring using wireless signals.

[57] HABER, E. The wiretapping of things. *UC Davis L. Rev. 53* (2019), 733.

[58] HACKETT, T. M., ET AL. *Procedia Engineering 159* (2016), 158–166.

[59] HAN, J., ET AL. Pitchln: eavesdropping via intelligible speech reconstruction using non-acoustic sensor fusion. In *Proceedings of the 16th ACM/IEEE International Conference on Information Processing in Sensor Networks* (2017), pp. 181–192.

[60] HAN, X., ET AL. Pre-trained alexnet architecture with pyramid pooling and supervision for high spatial resolution remote sensing image scene classification. *Remote Sensing* (2017).

[61] HE, K., ET AL. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778.

[62] JIANG, C., ET AL. mmvib: micrometer-level vibration measurement with mmwave radar. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking* (2020), pp. 1–13.
[63] KRIZHEVSKY, A., ET AL. Imagenet classification with deep convolutional neural networks. In *NIPS* (2012).
[64] KUNE, D. F., ET AL. Ghost talk: Mitigating emi signal injection attacks against analog sensors. In *2013 IEEE Symposium on Security and Privacy* (2013), IEEE, pp. 145–159.
[65] LI, Y. G., ET AL. *Orthogonal frequency division multiplexing for wireless communications*. Springer Science & Business Media, 2006.
[66] LI, Z., ET AL. WaveSpy: Remote and through-wall screen attack via mmWave sensing. In *2020 IEEE Symposium on Security and Privacy (SP)* (May 2020), IEEE.
[67] LIU, D., ET AL. Otsu method and k-means. In *2009 Ninth International Conference on Hybrid Intelligent Systems* (2009), vol. 1, IEEE, pp. 344–349.
[68] LU, C. X., ET AL. Milliego: Single-chip mmwave radar aided egomotion estimation via deep sensor fusion. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems* (New York, NY, USA, 2020), SenSys '20, Association for Computing Machinery, p. 109–122.
[69] MICHALEVSKY, Y., ET AL. Gyrophone: Recognizing speech from gyroscope signals. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)* (2014), pp. 1053–1067.
[70] MOHAMMADI, R. The future of 5g with sdr, Mar 2021.
[71] MURO, B. These cots sdr system solutions focus on 5g, Aug 2019.
[72] NAJMAN, L., ET AL. *Mathematical morphology: from theory to applications*. John Wiley & Sons, 2013.
[73] NASSI, B., ET AL. Lamphone: Real-time passive sound recovery from light bulb vibrations. Cryptology ePrint Archive, Report 2020/708, 2020. https://eprint.iacr.org/2020/708.
[74] NAWAZ, W., ET AL. Classification of breast cancer histology images using alexnet. In *International conference image analysis and recognition* (2018), Springer.
[75] NITSCHE, T., ET AL. Ieee 802.11ad: directional 60 ghz communication for multi-gigabit-per-second wi-fi [invited paper]. *IEEE Communications Magazine 52*, 12 (2014), 132–141.
[76] OWUSU, E., ET AL. Accessory: password inference using accelerometers on smartphones. In *proceedings of the twelfth workshop on mobile computing systems & applications* (2012), pp. 1–6.
[77] OZTURK, M. Z., ET AL. Radiomic: Sound sensing via mmwave signals. *arXiv preprint arXiv:2108.03164* (2021).
[78] PALIWAL, K., ET AL. Single-channel speech enhancement using spectral subtraction in the short-time modulation domain. *Speech communication 52*, 5 (2010), 450–475.
[79] PARK, S. R., ET AL. A fully convolutional neural network for speech enhancement. In *Proc. Interspeech 2017* (2017), pp. 1993–1997.
[80] PASZKE, A., ET AL. Automatic differentiation in pytorch.
[81] PRABHAKARA, A., ET AL. Osprey. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services* (June 2020), ACM.
[82] QU, C., ET AL. Bert with history answer embedding for conversational question answering. In *ACM SIGIR Conference on Research and Development in Information Retrieval* (2019).
[83] RAO, S. Introduction to mmwave sensing: Fmcw radars. *Texas Instruments (TI) mmWave Training Series* (2017).
[84] ROY, N., ET AL. Ripple: Communicating through physical vibration. In *12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15)* (2015), pp. 265–278.
[85] SAMI, S., ET AL. Spying with your robot vacuum cleaner: eavesdropping via lidar sensors. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems* (2020), pp. 354–367.
[86] TAGLIASACCHI, M., ET AL. Seanet: A multi-modal speech enhancement network. *arXiv preprint arXiv:2009.02095* (2020).
[87] TITZE, I. R., ET AL. Principles of voice production, 1998.
[88] UPADHYAY, N., ET AL. Speech enhancement using spectral subtraction-type algorithms: A comparison and simulation study. *Procedia Computer Science 54* (2015), 574–584.
[89] WANG, H., ET AL. Mole: Motion leaks through smartwatch sensors. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking* (2015), pp. 155–166.
[90] WANG, Z., ET AL. Uwhear: through-wall extraction and separation of audio vibrations using wireless signals. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems* (2020), pp. 1–14.
[91] WARDEN, P. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. *ArXiv e-prints* (Apr. 2018).
[92] WEI, T., ET AL. Acoustic eavesdropping through wireless vibrometry. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking* (2015), pp. 130–141.
[93] WU, C., ET AL. msense: Towards mobile material sensing with a single millimeter-wave radio. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies 4*, 3 (2020), 1–20.
[94] WU, C., ET AL. Msense: Towards mobile material sensing with a single millimeter-wave radio. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 4*, 3 (Sept. 2020).
[95] XU, C., ET AL. Waveear: Exploring a mmwave-based noise-resistant speech sensing for voice-user interface. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services* (2019), pp. 14–26.
[96] XU, C., ET AL. Cardiacwave: A mmwave-based scheme of non-contact and high-definition heart activity computing. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 5*, 3 (Sept. 2021).
[97] XUE, H., ET AL. Mmmesh: Towards 3d real-time dynamic human mesh construction using millimeter-wave. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services* (New York, NY, USA, 2021), MobiSys '21, Association for Computing Machinery, p. 269–282.
[98] ZHANG, G., ET AL. Dolphinattack: Inaudible voice commands. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (2017), pp. 103–117.
[99] ZHANG, H., ET AL. Pretraining-based natural language generation for text summarization. *arXiv preprint arXiv:1902.09243* (2019).
[100] ZHANG, L., ET AL. Accelword: Energy efficient hotword detection through accelerometer. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services* (2015), pp. 301–315.
[101] ZHAO, M., ET AL. Through-wall human mesh recovery using radio signals. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 10113–10122.
[102] ZHENG, W., ET AL. 5g v2x communication at millimeter wave: rate maps and use cases. In *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)* (2020), IEEE, pp. 1–5.
[103] ZHOU, Z., ET AL. Fine-tuning convolutional neural networks for biomedical image analysis: actively and incrementally. In *IEEE CVPR* (2017).

APPENDIX

A. Train/Test split across Phone Models and Frequency Bands

Fig. 24 depicts the accuracy in *mmSpy* across several combination of train/test split were explored where training and test data come from different phone models. The y-labels indicate the source of training data whereas the labels on each bar indicate the source of test data. The results include data averaged from 1-6 ft for AudioMNIST dataset. Note that we do not assume access to training data from the victim's phone in any of the above cases. For cases where training and testing data is coming from the same phone model, they are generated from two different phones of the same model. Evidently the accuracy levels with train/test data split across different phone models are lower than the overall accuracy levels where training data incorporates data from the same phone model. We hypothesize the difference comes because of the difference in material properties among smartphones which affect the properties of acoustic vibrations. Nevertheless the accuracy levels are still substantially higher than random guessing (10%) which can result in leakage of information. Training and test split across 60 and 77 GHz spectrum also shows a similar trend because the phase variations are a function of the wavelength. Fig. 25 further depicts cases where training data is derived from two settings and tested on a third different setting. These accuracies are higher in comparison to cases where training data is derived from only a single setting. This indicates that accumulating more training data

from diverse phone models can improve the robustness wh testing is conducted on a new phone model not included the training dataset.

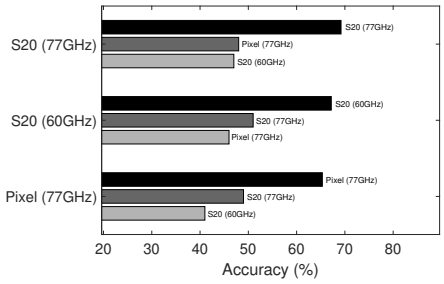


Fig. 24: Train/Test split across different phones and spectrum.

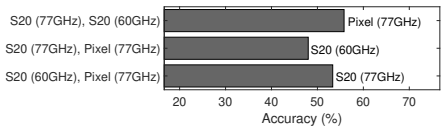
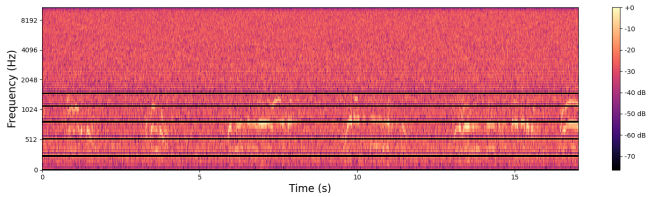
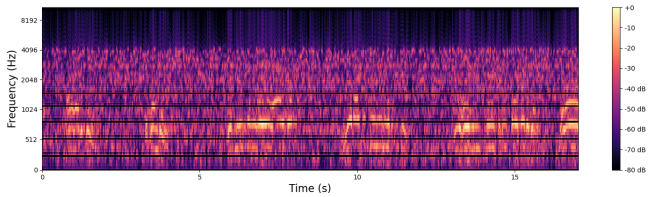


Fig. 25: Train/Test split across different phones and spectrum.



(a) Raw audio spectrogram (distance = 3ft, Recording [1



(b) Enhanced audio spectrogram (distance = 3ft, Recording [10])

Fig. 26: "I Have A Dream" – speech by Dr. Martin Luther King Jr.

(a) AudioMNIST

Class	Precision	Recall	F_1-Score
0	0.79	0.83	0.83
1	0.65	0.66	0.66
2	0.6	0.51	0.51
3	0.49	0.55	0.55
4	0.72	0.71	0.71
5	0.76	0.68	0.68
6	0.82	0.78	0.78
7	0.75	0.88	0.88
8	0.66	0.72	0.72
9	0.69	0.61	0.61

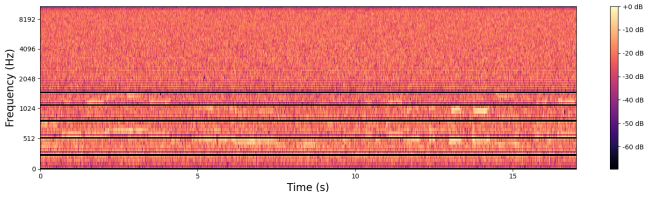
(b) Speech Commands

Class	Precision	Recall	F_1-Score
yes	0.67	0.68	0.68
no	0.52	0.66	0.66
go	0.74	0.63	0.63
stop	0.47	0.46	0.46
on	0.64	0.62	0.62
off	0.51	0.47	0.47
left	0.44	0.54	0.54
right	0.53	0.56	0.56
up	0.51	0.45	0.45
down	0.51	0.48	0.48

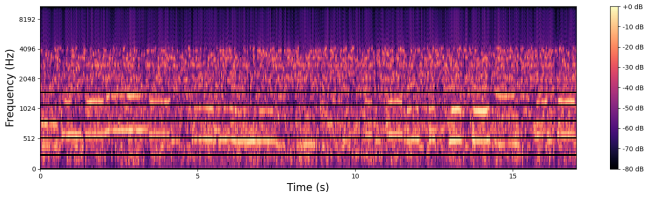
TABLE III: Descriptive statistics for S20 at 77 GHz with data from 1-6 ft distance combined.

B. System Parameters

Expanding on the high level overview in Section II-C, we provide a detailed description of the system parameters

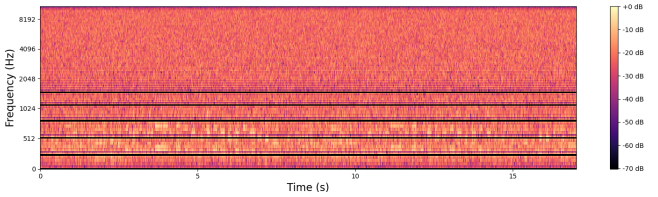


(a) Raw audio spectrogram (distance = 3ft, Recording [23])

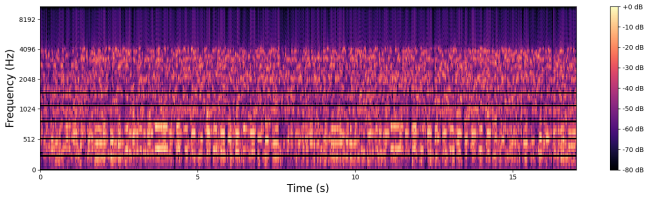


(b) Enhanced audio spectrogram (distance = 3ft, Recording [22])

Fig. 27: "Twinkle Twinkle Little Star" – sung by a child, with light music in the background.



(a) Raw audio spectrogram (distance = 3ft, Recording [21])

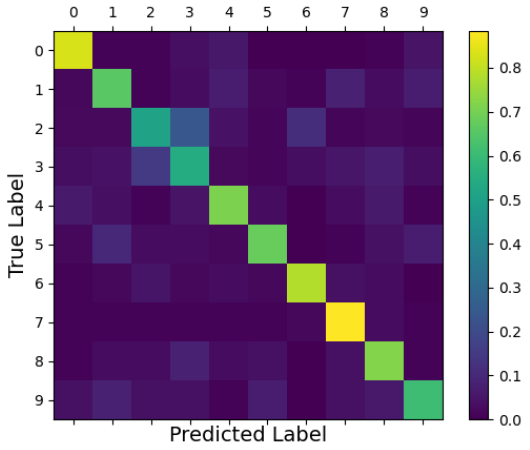


(b) Enhanced audio spectrogram (distance = 3ft, Recording [20])

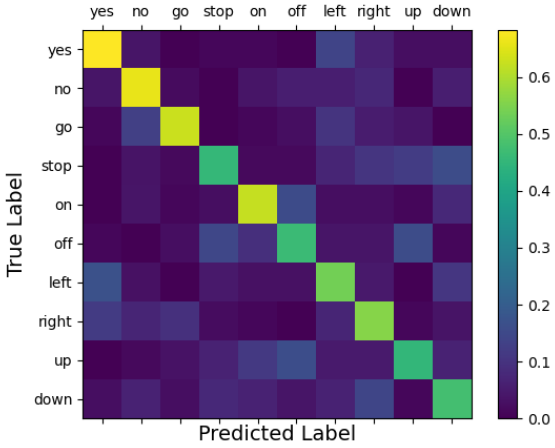
Fig. 28: "Rondo Alla Turca" ("Turkish March") – composition by Mozart.

in this section. From each chirp, we select the FFT-peak corresponding to the reflection from a phone and extract the phase. Thus, every chirp results in a single sample that can be converted to audio. To be more specific, the variation in phase of the FFT-peak from the reflection of the phone is the raw audio signal extracted from the radar. We term the frequency of chirp transmission as the *phase sampling frequency* since the phase of the FFT peak is selected and converted to audio. The necessary chirp parameters that determine the phase sampling frequency are:

1 *Start Frequency:* The starting frequency is the initial frequency at which the radar starts emitting a signal. In our system, the starting frequency is set as 77GHz.
2 *Frequency Slope (MHz/μs):* The TI *AWR1843BOOST* can modulate the chirp frequency linearly. The frequency slope determines the rate at which the frequency changes. In mmSpy, we set the frequency slope as 30MHz/μs. There are



(a) AudioMNIST



(b) Speech Commands

Fig. 29: Confusion Matrices for S-20 at 77 GHz with data from 1-6 ft combined.

practical limitations with setting a slope higher than this. For chirps with a small cycle time, if the slope is really high, the received signals become noisy as the system requires some time to cool down when transitioning quickly from the highest to the lowest frequency.

3 *Idle Time:* At the end of each chirp, the system is required to stay idle for sometime in order to avoid noise due to heat. The idle time in mmSpy is a low 20μs.

4 *TX Start Time:* The TX start time determines the time the transmitter takes to begin transmitting with respect to the start of a cycle. In our system, this is set as a low 10μs.

5 *ADC Start Time:* The ADC start time determines the time the ADC (at the RX) takes to begin converting received chirps. Note that the TX and ADC begin at the same instance of time but the TX start time can be earlier than the ADC start time. However, in *mmSpy*, the TX start time and the ADC start time are both set to 10μs in order to avoid further overhead and time delays and maximize the resolution of sensing.

6 *ADC Samples:* The number of ADC samples collected at the receiver is determined by this parameter. The ADC begins sampling at the the ADC start time and ends when this fixed number of samples have been collected. In *mmSpy*, the number of ADC samples is set to 256.

7 *Sample Rate:* The sample rate determines the rate at which discrete ADC samples are collected at the receiver. The TI *AWR1843BOOST* has an upper limit of 25Msps for real-valued data, and 12.5Msps for complex in-phase/quadrature (IQ) data. *mmSpy* collects IQ data and the sample rate is set as 10Msps.

8 *Ramp End Time:* The ramp end time determines the duration for which a chirp is emitted. It also determines the bandwidth of the transmitted signal, and consequently, the maximum range that can be detected. The ramp end time was set as 60μs.

9 *Chirp Cycle Time:* This is the sum of the idle time and the ramp up time. One value of phase is extracted per cycle.

The important parameters for the frames are:

- *No. of Chirp Loops:* The number of chirp loops determines the number of chirps within a frame. It is set as 128 in our system.
- *No. of Frames:* The number of frames that are transmitted and received is set as 800.
- *Periodicity (ms):* The periodicity of a frame is the total duration over which a frame is transmitted and received. In *mmSpy*, this is set as 10.64 milliseconds.
- *Duty Cycle (%):* The duty cycle of a frame is the amount of time for which frames are actively transmitted and received. The TI *AWR1843BOOSTe* requires that the device not be used on a 100% duty cycle, so as to allow it to cool down between frames. We use a duty cycle of 96.2%.

A few key considerations are made in setting the periodicity. First, the lower the periodicity, the better, as that allows us to capture more chirps within a given time period. Second, since there is a discontinuity between frames in order to allow the device to settle down (duty cycle is < 100). We decide to pad phase values due to such discontinuity between two frames with zeros. So the periodicity should be set such that the device is able to function at the assigned periodicity and that the number of zeros padded between frames is a discrete number. Additionally, we decided to use the same radar parameters for both the AWR1843 and IWR6843ISK, and we find that the settings we have mentioned work for both. We set the periodicity to 10.64ms as it is the smallest periodicity for which a discrete number of zeros can be padded between frames captured from both radars. The below equations further elaborate on the interrelationships between various system parameters as well as the computation of the final sampling rate based on the chosen parameter setting.

$$t_{\text{ramp}} = t_{\text{adc-start}} + t_{\text{adc-sampling}} + t_{\text{misc}} \tag{8}$$

where t_{misc} is the time spent transmitting at the end of a chirp cycle that is not sampled by the receiver.

$$t_{\text{cycle}} = t_{\text{idle}} + t_{\text{ramp}} \tag{9}$$

In mmWave studio, we can specify the values of t_{ramp} and t_{idle}.
Let the number of chirps be given by n_{chirps} and the number of zero skips between two frames be given by n_{skips}. Thus, the frame periodicity (p) is given by:

$$p = (n_{\text{chirps}} + n_{\text{skips}}) \times t_{\text{cycle}} \tag{10}$$

In mmSpy, we set $n_{\text{chirps}} = 128$ and $n_{\text{skips}} = 5$. This implies,

$$p = (128 + 5) \times t_{\text{cycle}} = 133 \cdot t_{\text{cycle}} \tag{11}$$

Since one phase value is extracted from each chirp cycle, the sampling rate F_s is given as:

$$F_s = \frac{1}{t_{\text{cycle}}} \tag{12}$$

We need to set t_{ramp} and t_{idle} such that the radar can successfully transmit and receive. In mmSpy, we set the values as $t_{\text{ramp}} = 60\mu s$ and $t_{\text{idle}} = 20\mu s$. Thus, $t_{\text{cycle}} = 80\mu s$.
This implies,

$$F_s = \frac{1}{t_{\text{cycle}}} = 12500Hz = 12.5kHz \tag{13}$$

and

$$p = 133 \cdot 80\mu s = 10.64ms \tag{14}$$

Thus, the values of frame periodicity is set as 10.64ms in *mmSpy*, ramp end time is set as $60\mu s$ and idle time is set as $20\mu s$. This results in a total bandwidth of 1798.92MHz. For *mmSpy*, we downsample the audio from 12.5kHz to 8kHz. Based on the Nyquist sampling theorem, the highest frequency audio signal that can thus be captured is 4kHz, which is adequate for speech recognition tasks.

C. Size of Attack Equipment

The current experimental setup is bulky. However, we note that the actual mmWave chip as highlighted in Figure 31 is only 2cm×2cm in size, and the dimensions of the antenna is 2.5cm×3cm. This can be integrated into a concealed PCB to enforce wireless wiretapping [57] and the raw data can be streamed to a smartphone with powerful GPU via 5G communication which can support Gbps data rates [28] sufficient for streaming the raw data at 25 MHz sampling rate (same as the sampling rate of our data acquisition device DCA1000). The development board shown in the figure is only used in the 'prototypying phase' as this is the standard procedure in many IoT applications to extensively test the prototype before rolling out on a compact PCB [32]. Our future work will include testing the feasibility of such a fabrication to create a smaller attack device.

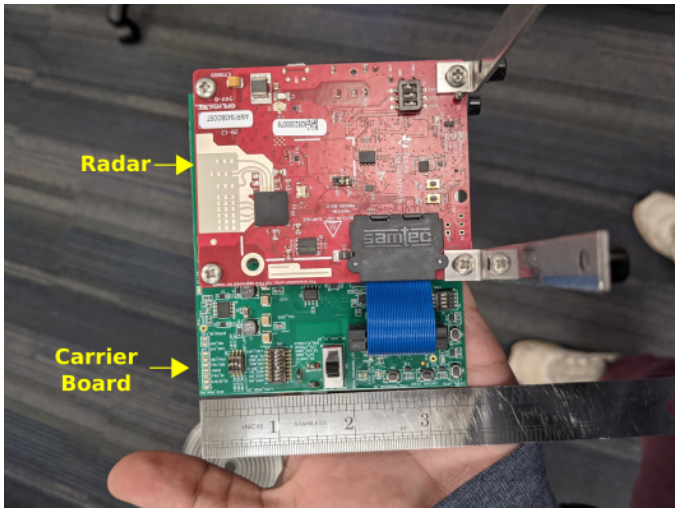


Fig. 30: Scale of development arrangement.

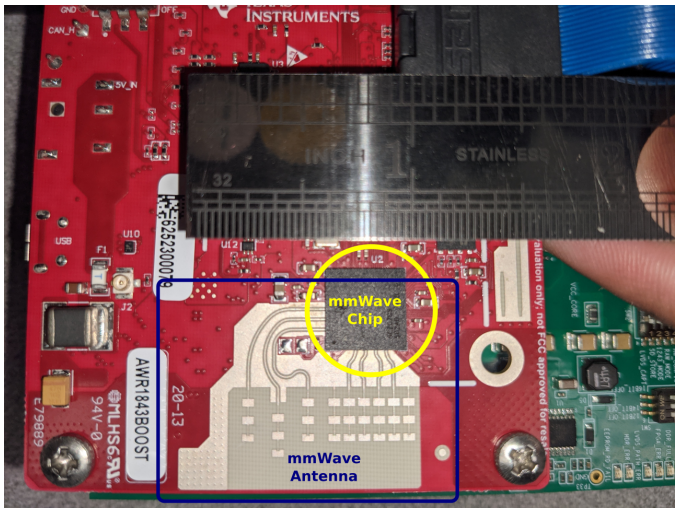


Fig. 31: Scale of chip and antenna.

D. Attack with 5G routers

At a high level, if an adversary gains access to the physical layer of a 5G router, or use a software-defined 5G radios which are becoming popular [70], [71], then they can modulate emitted radio waves. In such a scenario, it might be possible to modulate and mix sent and received signals to replicate an attack like mmSpy. At this point in time, this is a pre-emptive estimate of the possibility of such an attack; one of the reasons is that 5G has not yet fully proliferated as a popular technology. One of the key aspects of mmSpy is the short wavelength of the carrier wave used (which is of the order of millimeters), which makes tiny phase changes easy to detect: and this is a commonality that automotive radars share with commercial 5G appliances. We already have various hardware and open source software tools that can allow end users to build custom WiFi hardware [58]. We believe that with time, it will be possible to build adversarially-capable 5G hardware based on the same prototyping tools that enable their usual functionality.