Tracking Keystrokes Using Wireless Signals

Bo Chen†, Vivek Yenamandra† and Kannan Srinivasan
Department of Computer Science and Engineering
The Ohio State University, Columbus, OH 43210
{chebo, yenamand, kannan}@cse.ohio-state.edu
†Co-primary Authors

Abstract

We implement a passive remote keystroke detection mechanism using only changes in the wireless channel. The detection algorithm does not require the user to wear any active devices nor does it require a change in the user's wireless transmission technique. The receiver system is implemented with five antennas.We cancel the signals received on multiple antennas. The key insight to realizing a fine-grained localization system is to exploit the extremely high sensitivity of cancellation performance (interference cancellation in full-duplex for example) to exact amplitude and phase matching. The receiver design introduces a delay mismatch between the received signal streams to guarantee imperfect cancellation across the transmission bandwidth except at one in-band frequency resulting in a trough in the cancellation spectrum. We detect keystrokes by forming an array of observed trough frequency across different antenna pairs.

We implement our receiver system on the NI-based SDR platform. With full-training the receiver detects a keystroke within one key offset with an accuracy of 91.8%. With a short $\approx$ 10 character training input, this accuracy drops to 85%.

1. INTRODUCTION

The objective of this paper is to determine if it is feasible for a nearby receiver to detect and identify the keystrokes of a user typing on a wireless device by simply observing the resultant variations in the wireless channel. We assume a continuously transmitting wireless user device with a keyboard interface and place our receiver (hacker) in the vicinity of the user's device. The key challenge is that the hacker placed in the vicinity of the user's mobile device (say 5 meters), must still be capable of detecting keystrokes separated by only 2 cms. This accuracy is an order of magnitude higher than existing state-of-the-art systems [18]. Unlike fine-grained RFID tag localization, we assume the user does not wear any active tag/device to assist with localization of the finger presses [14]. Further, we consider a scenario where the hacker has to deal with limited training data since the user does not actively type anything to assist with keystroke detection [16]. Finally, we assume that the hacker is equipped with fundamental hardware capabilities comparable to existing wireless systems proposed in literature. The goal of this paper is to investigate the possibility of a constrained fine-grained wireless localization/detection system and its potential implications on issues of security/privacy. We present the design of a multi-antenna receiver system that enables the hacker to detect keystrokes by simply observing its effect on the wireless channel between the user and the hacker. We hope the findings of this work will serve as a catalyst to re-evaluate user interface designs for passing sensitive data considering the availability of increasingly sophisticated passive localization and gesture recognition wireless systems.

We have the user type on a keyboard interface of a device that is continuously transmitting wireless signals. We *cancel* the signals received on multiple antennas at the receiver. Consider two such antennas. We cancel the signal received on one antenna with the copy of it observed on the second antenna. In order to perfectly cancel the signals received on two antennas, we need to match both the amplitude and the phase of the signals. The cancellation performance is highly sensitive to matching the signals accurately in both amplitude and phase. To achieve uniform cancellation across the entire transmission bandwidth we need to match these signals across all transmission frequencies. We exploit this sensitivity of cancellation performance to map variation in cancellation performance across frequency to detecting and localizing keystrokes.

First, the receiver compensates for the difference in the channel between the user and each antenna. Second, we introduce a delay difference between the two receiver channels (from each antenna) to *introduce* a phase mismatch between the two signals. This phase mismatch causes degradation of cancellation performance across the transmission bandwidth except at specific frequencies where the phase mismatch between the two paths is an integral multiple of 2π radians. At these frequencies, the signal cancels *perfectly* while the cancellation performance at adjacent frequencies is still degraded because of phase mismatch. This results in *troughs* at specific frequencies when observing the power spectrum of the cancelled signal. We envision the user's finger when typing as the source of one multipath. When the user presses a different key, the source of this multipath changes resulting in the change of the delay difference between the two channels. This change in delay difference and the corresponding change in phase mismatch causes a change in frequency at

MobiSys'15, May 18–22, 2015, Florence, Italy.

http://dx.doi.org/10.1145/2742647.2742673.

which the cancellation is perfect and the resultant trough is observed. We map different keystrokes to the variation in the trough of the cancelled signal's power spectrum.

Ideally, a hacker system is constrained by limited-to-no training information. One can imagine getting short, limited training information by providing a hotspot high quality connection to the user and having the user type a set of keys to gain access. Thus, we assume such limited training data may be available at the beginning. However, since in our setting, the user is not assisting with detection, the user is not obligated to repeat key presses to provide large training data to assist with subsequent classification. Therefore, we use techniques that extrapolate key press identification beyond the keys used for training.

To our knowledge, this is the first RF keystroke detection and identification system. We develop new techniques to push the state-of-the-art gesture recognition and localization techniques to enable fine-grained keystroke detection with limited "training." The experiments in this paper rely on continuous wireless transmissions from the user device to enable keystroke detection and identification at the receiver. However, as discussed in Sec. 8, we believe with some implementation changes, the proposed receiver design itself should fundamentally be able to detect and identify keystrokes even in the presence of typical packetized Wi-Fi transmissions from the user device. We believe the findings of this paper presents the feasibility of potentially spying on keystrokes on a wireless device by an anonymous receiver, thus, exposing a potential threat.

We prototype our receiver on the NI based SDR platform. The two receive chains use NI-5791 radio front-ends, Virtex-5 based FPGAs for baseband signal processing and PXIe-8133 controller for real-time detection of keystrokes. Our receiver design only relies on the receive chains receiving the same signal from the transmitter and is agnostic to the underlying PHY layer design of the wireless transmission. With full-training the receiver detects a keystroke within one key offset with an accuracy of 91.8%. With a short $\approx$ 10 character training input, this accuracy drops to 85%.

The rest of this paper is organized as follows. In Sec. 2 we survey prior work and elucidate their shortcomings for our scenario, Sec. 4 details the design decisions of our receiver system from the physical signal perspective, Sec. 5 discusses the training and clustering algorithms we adopt for our system, Sec. 7 discusses the details of prototype implementation and system evaluation, Sec.8 discusses the current implementation limitations and finally Sec. 9 concludes the paper.

2. RELATED WORK

Researchers have been developing systems for indoor localization and gesture recognition. This section elaborates on the shortcomings of existing work to our setting.

Indoor Localization: Most of the work in the field of indoor localization involved adapting principles of RADAR based target finding to the RF based approach in indoor environments. RADAR [4] was one of the pioneering works in the field of indoor wireless localization. In this work, the authors aimed at localizing users inside buildings based on RSSI measurements at multiple access points and propagation models. Since then researchers have studied localization problems based on different types of wireless systems for various applications. A stream of work aimed at localizing transmitting nodes such as PinIt [14], PinPoint [9] etc. These systems rely on transmissions from nodes for localizing them. These techniques are referred to as active localization techniques where the readers perform signal processing techniques on the signals transmitted by the targeted device. Alternatively, researchers have studied using WiFi transmissions for localization and activity/gesture recognition. Arraytrack [18] is an example of an active localization system that uses a multi-antenna receiver to detect the angle-of-arrival (AoA) of the targeted device by observing the signal from that device on the multiple antennas at the receiver. Our work differs from this line of work. We seek to localize the finger and detect the key press where the user is not wearing any active tags. Our system is a passive localization and gesture recognition system.

An alternative line of work has considered passive localization and activity/gesture recognition systems. These systems observe the effect of the targeted device on the established wireless channel. Most of these designs require both a transmitter and a receiver. As part of the design, these techniques establish a channel between the transmitter and the receiver. The design typically involves changes in the transmitter to assist the receiver with localizing the passive targeted device. For example, in Wi-Vi [3], two transmit antennas beamform such that they null their signal at the receive antenna. This nulling phenomena relies on the channel. Changes in the channel due to movement of the passive targeted device causes change in the nulling performance which is mapped to the location of the target. 3D Body tracking [2] uses a continuous wave transmitter to help with tracking using time difference of arrival approaches. In Wi-Hear [13], the authors beamform the transmitter towards the mouth of the user to detect changes in channel caused by movement of the mouth at the receiver. Our system differs from this line of work since we use a single node to detect wireless keystrokes. We do not use an additional transmitter to beamform the signal in the direction of the targeted device. The rationale behind such a constraint is we envision the hacker to be stealth and not have a discernible placement of multiple nodes or antennas around the targeted user to detect their keystrokes. More recent works exploit synchronization between multiple far-apart nodes to achieve fine-grained localization [19] using triangulation techniques. Our system, however, operates by itself since having a spatially-extravagant setup itself could trigger suspicion.

Some researchers have observed changes in wireless channel between commodity radios due to the presence and movement of humans. For instance, E-eyes [17] localizes a person to a room in a house by measuring the change in the channel between a fixed WiFi device (say a TV) and the AP. The authors study temporal variations of the channel to provide broad activity classification. This approach would seem obvious for our setting. The user is on a wireless mobile device that is transmitting. The hacker device can in theory, be a receiver that continuously measures the channel between itself and the user and employ signal processing techniques on the observed channel measurements to detect and differentiate different keystrokes by measuring changes in the phase and/or magnitude of the channel. We implemented a receiver system with this objective. We continuously measure the phase of the channel and seek to find a pattern in phase variation for the same repeated action (example: pressing the same key.) However, as illustrated in Fig. 1, the vari-

ation in absolute phase and magnitude of the channel is so small that these changes are buried within the noise. The finite resolution of off-the-shelf ADCs limits the efficacy of this procedure. Thus, existing signal processing techniques on absolute channel measurements does not provide information specific to our setting.

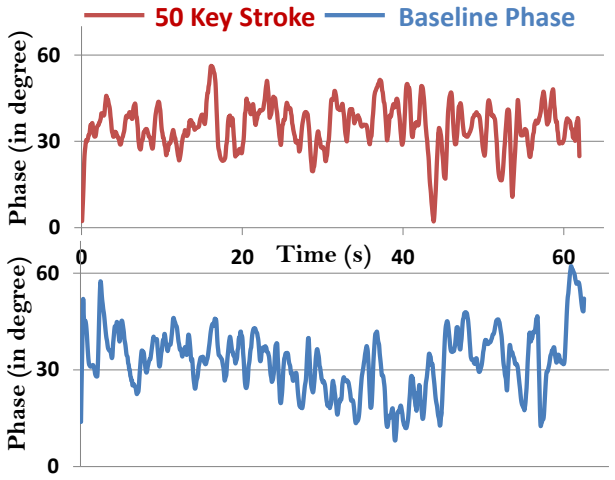


Figure 1: Phase Variation in Channel: In the presence and absence of repeated key stroke, 'A'. The absolute phase variation is minimal causing the signal to be buried within noise

Gesture recognition: In addition to indoor localization, researchers have also studied the problem of gesture recognition. The problem of detecting keystrokes is not merely a localization problem. The hacker must be capable of differentiating between a user simply moving his fingers along the keyboard and actually pressing a particular key. Here, the act of key press itself is the activity that the hacker must be capable of detecting. While E-eyes [17] classifies between different activities while providing some location information, the classification and localization is very coarse-grained for our application. There are finer gesture recognition systems such as WiSee [11], AllSee [10] etc. WiSee detects a gesture using the phenomena of Doppler shift. The movement of a hand to and from a receiver changes the relative frequency of the wireless signal observed at the receiver. The authors present signal processing techniques to detect small Doppler shifts resulting from simple hand gestures. While this is a fine-grained gesture detection system, it does not provide any information about the location of the target and these principles do not directly extend to our scenario. RF-Idraw [15] is a wireless system to detect a user writing in the air. RF-Idraw has a user wear an active RFID tag to help with tracking the movement of the finger. The technique deployed does not extend to a scenario where the target is passive. Further, the system is concerned with tracking motion of a finger and not precise location of the finger. The requirement for our considered setting is more stringent.

Finally, a recent system [16] enables a mobile device to detect keystrokes pressed on a nearby hard surface (For ex: Enabling a user a larger paper keyboard interface for a small-screen mobile device.) This paper requires the mobile device to be placed in the immediate vicinity of this paper keyboard. The system observes the acoustic channel using the microphone input on the mobile system and harnesses the multipath sound profile to detect between different keystrokes. This solution requires the reader to be placed really close to the user and requires an extensive training period for key detection and classification. In contrast, our work does not have the luxury of user cooperation to have extensive training. Furthermore, proximity is also not possible in our setup. This makes our system susceptible to more external channel variations. However, as Section 5 shows, our system can identify and filter out external variations.

Spying Keystrokes: There has been recent work focused on detecting keystrokes anonymously. In [12], the authors observe the effect of keystrokes on the electromagnetic radiation emanating from the keyboard to detect the keystroke pressed. The authors exploit the fact that keyboards are typically cheap devices with poor electromagnetic shielding. The receiver observes the electromagnetic radiation effect of the clock and data signals for different keystrokes. These signals are digital signals of the order of 10s of MHz (up to about 150 MHz after accounting for observable harmonics of the on-board clock signals). Thus, to observe these signals the receiver requires large antennas. In [12], the authors use a 1m copper wire antenna for the receiver, which can be prohibitively large to maintain discreetness as a hacker. More recently, in [1], the author employs an anonymous wireless USB device to spy on keystrokes from a wireless keyboard. However, this work relies on the information of the data-transmission protocol of the wireless keyboard and its weak security features. Our design only observes the changes in the wireless channel between the user and the hacker due to the movement of the finger because of a keystroke to detect the keystroke. Our design is agnostic to the underlying PHY and MAC protocols.

Full Duplex Systems: Thus far, we have listed the shortcomings of existing literature for the discreet hacker scenario considered in this paper. In order to be able to detect and distinguish between small changes in the phase of the wireless channel, we exploit the sensitivity of self-interference cancellation of full-duplex systems to exact phase matching. Small changes in phase of the channel causes change in the frequency at which we observe the cancellation trough. In recent times, there has been a large drive in developing full-duplex systems, [5–8]. There is a subtle difference between our cancellation and typical full-duplex self-interference cancellation.

These systems have a training phase to estimate the channel between the transmitter and the receiver. The receiver then cancels the observed signal with the internally generated copy of the received signal. In our implementation we do not have an explicit training signal for cancellation. Instead, we make use of the fact that the two receive streams receive the same signal. We model the difference in the channel between the transmitter and these two receivers to subsequently subtract the receive stream.

3. DESIGN PRINCIPLES

This section presents the core design of our system, its tuning knobs and the sensitivity of the system.

3.1 2-Receiver Setup

Fig. 2 illustrates the scenario considered in this paper. A user on a wireless device types on a keyboard interface. The wireless device has a transmitting antenna. For now, we

assume the keyboard interface is at a fixed distance and orientation with respect to the user's antenna as would be the case in laptops. The hacker places his wireless device close (< 5m) to the user to detect the keystrokes. The hacker's wireless receiver contains two antennas that are fixed in distance and orientation relative to each other. The receiver then cancels the signal received on one antenna with that received on the other antenna. This cancellation performance is sensitive to the phase mismatch between the transmitter and the two receive antennas caused by movement of the finger in the act of a key press.

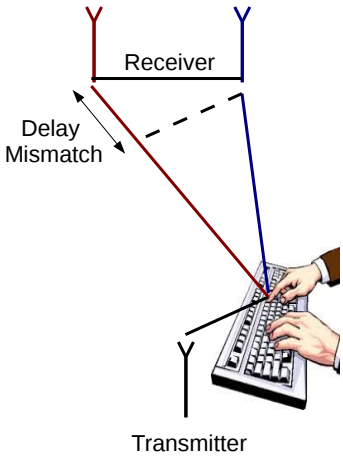


Figure 2: Delay mismatch due to keystroke δ_t.

3.2 Intuition

Initially, the receiver estimates the difference in the wireless channels between the transmitting antenna and the two receive antennas. Once the receiver matches the amplitude and phase of the channel from the transmitter to each receive antenna, it can, in theory, cancel the signals on the two receive antennas to achieve perfect cancellation across the entire transmission bandwidth, B. Let the two signals received on the two receive antennas, after this matching, be y_1 and y_2, which are functions of both frequency (in radians), ω, and time, t.

$$y_1(\omega, t) = A(\omega, t)e^{j\omega t} \tag{1a}$$
$$y_2(\omega, t) = A(\omega, t)e^{j\omega t} \tag{1b}$$

Note that if we subtract (cancel) y_1 from y_2 then the cancellation will be near-perfect and will be so across the entire bandwidth, a desired property in full duplex designs. We deviate from this goal and artificially introduce a delay of Δ_t to one of the two received signals as indicated in Eq. 2.

$$y_{2delay}(\omega, t) = A(t)e^{j\omega t - \omega\Delta_t} \tag{2}$$

Now, while the two signals are still matched in amplitude, the introduced delay causes a phase mismatch across frequencies: Each signal experiences a phase shift proportional to $\omega \times \Delta_t$ radians. Clearly, subtracting y_1 from y_2 will not yield perfect cancellation anymore (Eq. 3b). In these equations, we drop the parameters ω and t for notation simplicity.

$$y_{cancel} = y_{2delay} - y_1 \tag{3a}$$
$$= A(t)\left\{e^{j\omega t - \omega\Delta_t} - e^{j\omega t}\right\} \tag{3b}$$
$$y_{cancel} = 0 \iff \omega\Delta_t = \pm 2n\pi, n \in 0, 1, 2, \cdots \tag{3c}$$

While the cancellation of the two streams does not reduce to 0 across all frequencies, there exist specific frequencies at where this cancellation is perfect. We introduce the delay, Δ_t such that we find $\omega \in B$ where $y_{cancel} = 0$. We refer to this frequency as ω_t. This leads to observing a sharp trough at ω_t within the transmission bandwidth, B.

Now, the system is bootstrapped. As the user types different keys, however, the two received signals observe varying delays between the two receive antennas. This variation introduces significant phase changes across the bandwidth. This causes the location of the trough to change.

Assume the receiver estimates the channel between itself and the user when the user has placed his hands in a resting position on the keyboard. After introducing a delay, the receiver observes a cancellation spectrum with a trough at frequency ω_t. When the user now moves his finger to press a different key, the difference in delay between the two receive streams changes. This can be easily understood by visualizing the finger as a source of transmission. When the position of the finger changes, the delay between the finger and each receive antenna changes. This causes a change in the initial delay difference between the two receive streams from Δ_t. Let us denote this delay change due to change in finger position by δ_t. This additional delay again introduces different phase shifts at different frequencies. This causes the cancellation trough to move to a different frequency, say ω_{new}. In other words, at ω_{new}, the phase mismatch between the two streams is an integer multiple of 2π radians as indicated in Eq. 4. Fig. 3 gives a sense of how sensitive this cancellation is on the delay variations introduced by δ_t.

$$\omega_{new}\{\Delta_t + \delta_t\} = \pm 2n\pi, n \in 0, 1, 2, \cdots \tag{4}$$

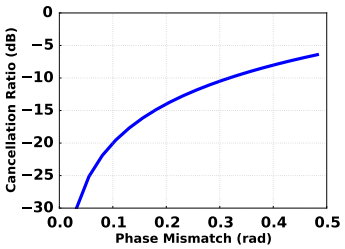


Figure 3: Cancellation Ratio vs Phase Mismatch introduced by δ_t. Cancellation Ratio is the ratio between signal power after and before cancellation: Lower the ratio, better the cancellation. A small increase in phase mismatch adversely affects cancellation quality.

Thus, by tracking ω_{new}, we can track the location of the finger over the keyboard interface. This is the core principle of our design.

3.3 System Sensitivity

In this section, we identify the variables that affect the sensitivity of our system. These variables are the tuning knobs in our design.

3.3.1 *Sensitivity to Δ_t*

As discussed in Sec. 3.2, we introduce a delay between the two receive streams to ensure we do *not* observe a flat cancellation spectrum. Once the receiver estimates the channel between the user to each of its receive antennas, we can ensure phase matches only at a particular in-band frequency by carefully selecting Δ_t. Subsequent key presses can be identified by observing the shift in frequency at which the trough is observed.

However, the selection of Δ_t serves another purpose. As indicated in Eq. 3b, the phase mismatch between the two receive streams at a particular frequency is $\omega.\Delta_t$. Thus, for a given frequency the phase mismatch increases as Δ_t increases. This makes the *width* of the observed trough narrower. Suppose the receiver observes a trough at frequency ω_t, where the phase mismatch between the two streams is 2π radians. The phase mismatch at an adjacent frequency ω is then $2\pi + |\omega - \omega_t|\Delta_t$ radians. Thus, for the same frequency offset from the frequency at which the trough is observed, ω_t, the phase mismatch increases with an increase in Δ_t. As indicated in Fig. 3 this increased phase mismatch causes a larger degradation of cancellation performance at frequencies adjacent to ω_t, causing the observed trough in the cancellation spectrum to become narrower. This is indicated in Fig. 4.

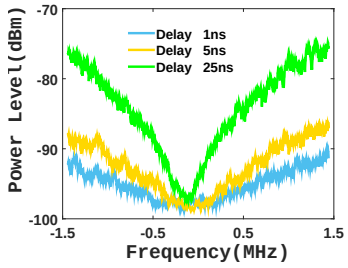


Figure 4: The width of observed trough in Cancellation Spectrum vs Δ_t. Larger Δ_t gives narrower trough.

In addition to making the trough deeper, the selection of Δ_t also affects the sensitivity of the receiver. For instance, let us assume a user presses key 4 and then, presses key 5 (on a typical keyboard interface). For this key press sequence, Fig. 4 shows the difference in the frequencies where the trough appears (marked as sensitivity in the figure) against Δ_t. It shows that reducing Δ_t increases sensitivity. This means that simply pressing adjacent keys can push the trough beyond the observable bandwidth (B). On the other hand, choosing a very large Δ_t will not noticeably move the trough. Thus, Δ_t should be chosen carefully as shown in the next section.

The following section will detail our design algorithms and present convergence times for the trough generation part of our design.

4. SYSTEM DESIGN

Thus far, we have detailed the setup of our system and the intuition behind some of the design parameters of the design. Further, we illustrated the cancellation performance's sensitivity. This section discusses the design of our system in detail.

As stated before, the first bootstrapping step is to cancel the two signals from the two receive antennas such that a trough appears within the observed bandwidth. To do this, we need to match the amplitude of the two received signals in addition to introducing a delay of Δ_t, iteratively. Fig. 5 illustrates this process.

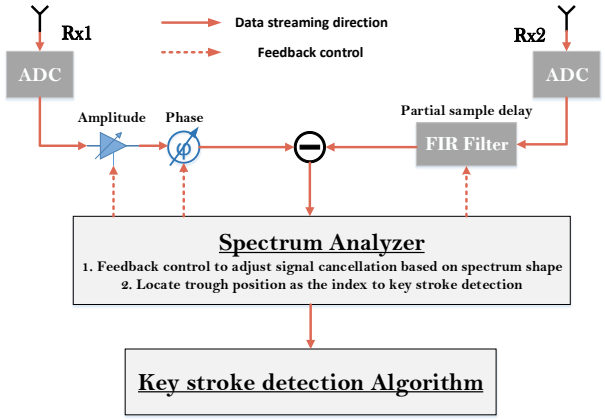


Figure 5: Block Diagram illustrating the design principles of our receiver.

4.1 Matching Amplitude

As Fig. 2 shows, we have a receiver with multiple antennas at fixed distances relative to each other. We place the receiver at a certain distance with respect to the user. The distance between the user and the receiver ($\approx 5m$) is significantly higher than the distance between the antennas ($\approx 6cms$). We initially assume that the two receive antennas receive the signal with comparable power. Thus, initially, we observe a trough in the cancellation spectrum. We are guaranteed the presence of a trough since we introduce a delay in one of the receive streams as indicated in Fig. 5. After subtracting the two receive streams, we send the cancellation signal through a spectrum analyzer block as indicated in Fig. 5. The spectrum analyzer block computes the spectrum of the cancellation signal. On the computed spectrum, we run a trough detection algorithm to accurately detect the trough frequency. Since the initial assumption of equal amplitudes on both receive streams is only an approximation, the cancellation at the detected trough is not *perfect*. In other words, the trough observed in the cancellation spectrum is not deep. Thus, in addition to a trough detection algorithm, the spectrum analyzer block also runs an amplitude matching algorithm. The amplitude matching algorithm continuously seeks to match the amplitudes of the two receive streams to result in perfect cancellation at the trough frequency. This results in a deep trough in the observed spectrum as well as increases the accuracy of the detection of the trough.

To match the amplitudes of the two streams, we define *correction ratio*. The correction ratio settles to a ratio that

Algorithm 1: Pseudo-code for Magnitude Matching Algorithm

```
   // Parameter Initialization:
1  ratio = 1
2  scale = 1.003
3  case = 1

   // Keep running Loop:
4  while true do
5  |  if pre_Ptrough > curr_Ptrough then
6  |  |  case = 1 − case
7  |  endif
8  |  switch case do
9  |  |  1: ratio = ratio × scale
10 |  |  0: ratio = ratio ÷ scale
11 |  endswitch
   |  // delay to guarantee the new ratio takes effect
12 |  delay()
13 endwhile
```

would match the amplitude of the two streams, specifically at the trough frequency. To match the amplitude of the two streams, we multiply the weaker receive stream (in terms of signal strength) with the computed *correction ratio*. The receiver subtracts one of the receive streams with the receive stream updated with the new correction ratio. The spectrum analyzer block then computes the spectrum of the updated cancellation signal. The amplitude matching algorithm observes the effect of the previous correction ratio on the cancellation spectrum.

Initially, we assume the correction ratio to be 1. The amplitude matching algorithm adjusts the correction ratio with the objective of minimizing the power at the trough frequency. The algorithm to settle on the correction ratio is given in Algorithm 1. The correction ratio is multiplied in each iteration with *scale* depending on the effect of the previous value of the correction ratio. Thus, a smaller *scale* factor makes the steady-state correction ratio more stable resulting a deeper trough. However, a smaller *scale* factor also causes the correction ratio to take longer to reach that steady-state. We evaluate this trade-off in greater detail in Sec. 7.

Convergence Time: As indicated in the algorithm listed above, we use a scale factor of 1.003 in our design. This implies that, in each iteration, we change the value of the correction ratio by 0.3% of its current value. At a first glance, this might suggest that the correction ratio might take a long time to reach steady state and the latency might be intolerable. However, in our experiments we observe the contrary. We measure the time it takes for the correction ratio to settle to a value to match the amplitude of the two receive streams. Initially, the trough detection algorithm locates the frequency at which the trough occurs. The amplitude matching algorithm then alters the correction ratio to match the amplitudes of the two receive streams. As suggested in Fig. 3 the cancellation performance is highly sensitive to phase and amplitude matching. When the amplitudes of the two streams are close to each other then a small change in the amplitude can potentially change the cancellation performance significantly. We measure the time taken by amplitude matching algorithm for different orientations of the user relative to the hacker device and under different environment conditions (with respect to objects in the environment and their corresponding multipath profiles). Fig. 6 plots the time taken for one such experiment run. In addition, it also contains the ratio of the change in amplitude of one of the streams from its initial amplitude to its matched amplitude. The time taken is quite insignificant (of the order of 10s of milliseconds) compared to the typical cadence of typing on a keyboard. Further, the amplitude matching algorithm is only instantiated periodically. Typically, once the amplitudes converge, the relative ratio of amplitudes on the two receive streams only changes in the presence of significant environment changes. We believe we can decrease the convergence time further by implementing the algorithm on the FPGA itself.

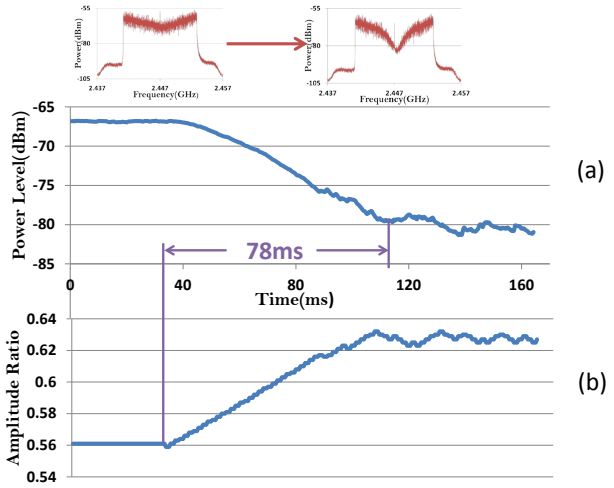


Figure 6: Amplitude Matching Algorithm: The correction ratio converges to match the amplitudes of the two RX streams in about 78ms. The resultant drop in the power at the trough frequency is also indicated. To give a qualitative feel of the effect of amplitude matching algorithm we present a snapshot of the power spectrum before and after the onset of the algorithm.

4.2 Designing Δ_t

Sec. 3.2 discussed the impact of Δ_t on the system's performance. As indicated in Fig. 5, we introduce this delay in one of the receive streams to cause a phase mismatch between the two receive streams. We add this delay mismatch between the two receive streams in the digital domain by adding it after ADC sampling. The two receive RF-chains have identical specifications. The delay of the signal from the receive antenna to the input of the ADC in each of the receive stream is deterministic and consistent. However, since each receive stream has its own independent ADC, initially we noticed a sampling offset between the two ADCs. While this sampling offset is consistent for the duration of each run, on each power cycle, this sampling offset can change to a different value. However, the NI-based platform used to implement the receiver system is equipped with control lines in the backplane shared across different FPGAs of the receiver. We use these control lines to synchronize the ADCs on each power cycle. The implementation system is explained in greater detail in Sec. 7. Synchronizing the ADCs resulted in a consistent delay between the receive antenna and the base-

band in each receive stream. This consistency enables us to add deterministic delay mismatch between the two receive streams.

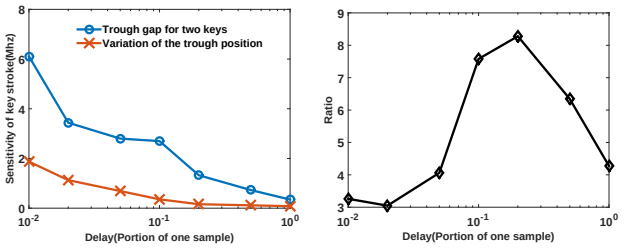
We introduce the delay mismatch between the two streams using the *partial sample delay* block as indicated in Fig. 5. We implement the partial sample delay block using an FIR filter. The FIR filter has a linear phase characteristic across its passband frequency to introduce the desired delay. The delay introduced by this filter can be configured to sub sample periods. For the desired delay, the receiver computes the FIR filter coefficients (by performing an IFFT of the delayed channel) and loads them into the filter.

One would expect the optimal delay mismatch introduced between the two streams to be dependent on the distance between the receiver and the user and the orientation of the user with respect to the receiver. As discussed in Sec. 3.2, the delay mismatch affects the depth of the observed trough in the cancellation spectrum as well as location sensitivity. Also, assume for a given keystroke, say A, the trough in the spectrum is observed at frequency ω_A. For a given orientation and distance between the user and the receiver, we would expect that for each repeated key press of A, we observe a trough at ω_A. However, due to noise in phase measurement, we observe that the trough is observed at a distribution of frequencies centered around ω_A. The variance of this distribution tends to zero as we increase the delay mismatch. We elaborate on this in Sec. 7. Considering the influence of delay mismatch on the performance of the receiver, we observe that for majority of the scenarios considered in this paper, a fixed delay mismatch provides *sufficient* performance. Thus, in our implementation, we fix the delay of this partial sample delay block and thus the delay mismatch between the two streams.

Determining Delay Mismatch:

We seek to determine the impact of delay mismatch on the performance of the system. We fix the position of the transmitter and the position of the keyboard relative to the transmitting antenna. We place the keyboard close to the transmitting antenna and at a lower plane to mimic the setting of a typical laptop. We place the hacker device at a distance of 3m from the transmitter. The two receive antennas of the hacker are spaced apart by 6cms. We fix the orientation of the hacker with respect to the transmitter and the keyboard. We have a subject repeatedly type two adjacent keys, '4' and '5' respectively. For each delay input to the partial sample delay block, we measure the sensitivity of the hacker device. We define sensitivity as the difference in the observed trough frequency, $F_4 - F_5$, where F_k is the frequency of the observed trough when key 'k' is pressed. We vary the delay in the partial sample delay from 0.01 to the time period of one sample in fractional steps (as indicated in Fig. 7(a)). For each delay setting we have the user type the keys '4' and '5' repeatedly 50 times.

For each key press, we record the location of the detected trough. We observe that the average sensitivity decreases as we increase the delay mismatch. However, while the sensitivity decreases we observe that even for a delay mismatch of one sample duration, the two keys can be reliably distinguished. The sampling rate of our system is 20 Msps. Specifically, for a delay mismatch of 0.01 × sample period, the sensitivity of adjacent keystrokes is ≈ 6 MHz. Increasing the delay mismatch to 0.1 × sample period reduces the sensitivity to ≈ 3 MHz.



(a) Sensitivity and Variance. (b) Ratio of Sensitivity to Variance.

Figure 7: Influence of delay mismatch on the sensitivity of key detection. As expected from the discussion in Sec. 3.2 the sensitivity decreases as the delay increases. However, the lack of sensitivity assists in reducing the variation of the trough location due to measurement noise. We observe introducing delay mismatch of 0.1× sample period is an acceptable value to achieve a workable trade-off in most scenarios.

In addition to computing the average sensitivity, we also measured the repeatability of the trough location for a given key stroke. For each repeated key press, we record its trough location and compute the variance in the trough location. As indicated in Fig. 7(a), we observe that for small delay offset, the variance in the trough location is ≈ 2 MHz. A small delay offset implies that the phase mismatch between the two receive streams is small. Thus a small change in the observed phase in either of the receive streams causes the frequency at which the phase matches to change. This causes high variation in the trough location for a small delay offset. Measurement noise and multipath can cause variation in the observed phase of the two streams which in turn causes variation in the observed trough location for the same key press in the case of small delay mismatch. However, as we increase the delay mismatch, we decrease the sensitivity of the hacker and the trough location for a given keystroke is more robust to measurement noise.

To quantify the trade-off between sensitivity and accuracy of a given trough location we define a ratio between sensitivity and trough accuracy. We plot this ratio for different delay mismatch inputs in Fig. 7(b). We observe that a delay mismatch of 0.1 × sampling period provides sufficient sensitivity with acceptable trough location robustness. We observe that this delay is acceptable for majority of the orientations between the hacker and the user. Thus, in our system, we preconfigure the partial sample delay block to provide 0.1×sample period delay mismatch between the receive streams.

Finally, for the selected delay mismatch, it is possible that in the cancellation spectrum, we do not observe a trough. This is because, it is possible for the frequency at which the phase matches can fall outside the transmission bandwidth. This can occur due to changes in the environment or orientation of the user with respect to the hacker. While changes in environment can push the trough outside the transmission bandwidth we are guaranteed the existence of a frequency itself at which a trough exists since we introduce the delay mismatch between the two streams. Thus, when we do not observe a trough in the cancellation spectrum, we examine

the shape of the spectrum. For example, if the trough occurs at a frequency to the *right* of the transmission bandwidth, then the observed spectrum appears *tilted down* in the direction of the trough frequency. This is illustrated in Fig. 8. We then sweep the phase of one stream between $[-\pi, \pi]$ to match its phase with the other stream at a frequency that falls within the transmission bandwidth. Similar to the amplitude matching algorithm, the spectrum analyzer block also contains a spectrum shape sensing algorithm that is responsible for observing the shape of the spectrum and updating the value in the phase block indicated in Fig. 5.

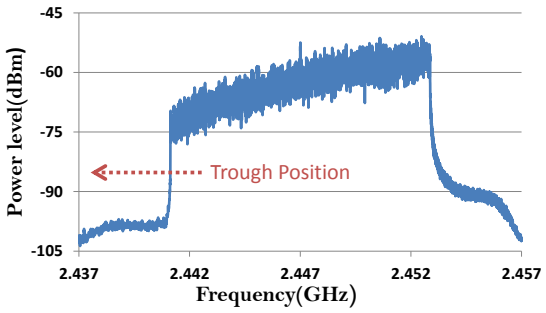


Figure 8: Spectrum shape sensing block observes the shape of the spectrum to control change in the phase block (Fig.5) to retrieve trough to an in-band frequency.

Further, this enables us to control the initial location of the trough at frequency, say $\omega_{initial}$. We re-initialize the trough location to this frequency at designed intervals. This operation only determines the location of the trough and does not vary the width or robustness of the trough location. Thus, in this operation we are not changing the delay of the signal but simply changing the phase to ensure location of the trough.

5. KEYSTROKE DETECTION

Thus far, we have discussed the design principles and decisions of all parameters until the spectrum analyzer block of the design process illustrated in Fig. 5. This section details the keystroke detection algorithm block in our system.

The design decisions discussed thus far, determine the sensitivity of the hacker system. The desired sensitivity enables the hacker to distinguish between the location of the user's finger on one key from its adjacent key. However, it does not provide any information regarding whether the key was actually pressed. Second, assume that for a given keystroke, the hacker observes a trough at frequency ω. In a relatively stable environment, the same keystroke should cause a trough in the cancellation spectrum at the same frequency as indicated in Fig. 7(a). However, when there are changes in the environment, such as a person walking in the vicinity of the user-hacker setup can cause the location of the trough in the spectrum to change. Thus, it is important for the keystroke detection algorithm to differentiate the cause of movement of the trough location in the cancellation spectrum due to an actual keystroke from that caused due to channel variations.

To overcome the above concerns, our keystroke detection algorithm tracks the change in trough frequency over time. This temporal information enables the hacker to distinguish a keystroke from other changes in the environment that affect the cancellation spectrum in a similar manner. The algorithm computes the rate of change in the trough frequency. Using an empirically selected threshold, the algorithm then classifies a given change in trough location as either a keystroke or not a keystroke. The temporal variation of the trough frequency when a keystroke is repeated is given in Fig. 9.

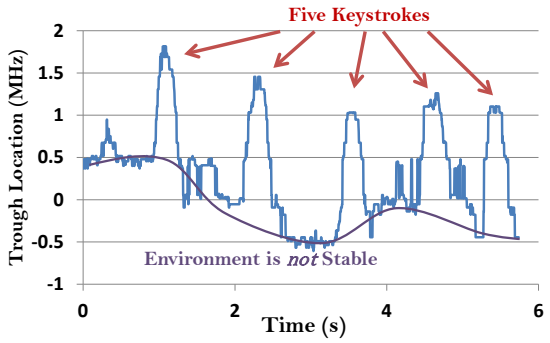


Figure 9: Tracking time variations in trough frequency due to changing environment conditions introduced by moving people near the user.

We measured this temporal information of the change in trough frequency locations by having a user repeat a keystroke repeatedly. While the user is repeatedly pressing a key, we ask another person to walk in the vicinity of the user-hacker setup. As indicated in the above figure the absolute trough location varies over time due to change in the channel caused by the movement of a person in the channel. However, the rate of change of the keystroke is significantly higher than typical changes encountered in environment. This is because a keystroke is typically a short 'impulse' like movement of the finger. It must be noted that while the trough location itself in the cancellation spectrum can change, the rate of change in the trough location for a keystroke remains reliably steady for a given orientation between the hacker and the user. We perform repeated experiments to arrive at an empirical threshold for rate of change of the trough frequency. Similar to selection of Δ_t parameter we find that a single threshold selection suffices for detecting keystrokes with high probability in most envisioned user-hacker scenarios considered in this paper.

Finally, in some cases it is possible that the change in environment is large enough to force the observed trough in the cancellation spectrum outside the transmission bandwidth. Once the trough detection algorithm in the spectrum analyzer block detects the absence of the spectrum it triggers the spectrum shape sensing block which subsequently changes the phase of one of the receive streams to force the trough back inside the transmission bandwidth. We repeat the previous experiment where the user repeatedly enters the same keystroke sequence. Unlike the previous experiment, here the user is asked to type two keys. We ask the user to introduce large variation in the channel (for example: using the free hand to mimic the action of swatting a fly). We use the keystroke detection algorithm discussed above to detect the keystroke. Fig. 10 plots the change in trough frequency location over time. The sudden variation in the absolute trough frequency is caused by a large change

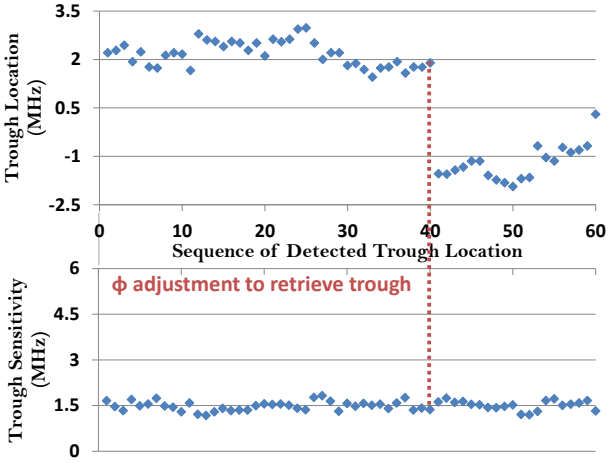


Figure 10: Tracking trough sensitivity and absolute trough frequency sequences. While large channel variations (introduced by user moving his hand) can vary absolute trough frequency, trough sensitivity remains the same for a given receiver location, orientation and given set of user keystrokes.

in the environment which causes the hacker to adjust the phase to reinforce the trough frequency to within the transmission bandwidth. However, as indicated in the figure, this operation does not change the observed sensitivity. The key takeaway from this experiment is the following: Even in the case of large channel variations, the sensitivity of the hacker for a given pair of keystrokes remains reliably stable as long as the orientation of the hacker relative to the user remains the same. We exploit this observation to increase the confidence of relative keystroke detection given knowledge of an initial reference keystroke. This is explained in greater detail in Sec. 7.

6. IMPLEMENTATION

We implement the hacker's receiver system on the NI-based SDR platform. We implement an initial receive system with two antennas. We later extend this design to five receive antennas for preliminary results. In this section we present the implementation details of a two-antenna receiver to detect keystrokes.

We equip our SDR platform with two antennas and two independent receive RF chains. Each RF chain uses the NI-5791 radio front-end and a Virtex-5 based FPGAs for baseband processing. We operate the receivers in the 2.4 GHz ISM band. The ADC of each receive chain samples at 130 MSps. The ADC resolution is 14 bits. We sample data into the FIFO itself at 20MSps. We establish a DMA (Direct Memory Access) channel between the two receive FPGAs to transfer samples from the RX2's ADC (indicated in Fig. 5), to RX1. In the FPGA of the second receive chain we cancel the two received streams of samples. We implement the partial sample delay block indicated in Fig. 5 and the amplitude correction block also on the same FPGAs. The subsequent cancellation samples are processed into a FIFO that are transferred to a central controller. The central controller is implemented on the RTOS based PXIe-8133 to enable deterministic processing and data transporting delays between the central controller and the FIFOs in the FPGA of the receive chains. The central controller is responsible for synchronizing the independent RF oscillators as well as the sampling clocks of the ADCs of each receive chain. In addition, the central controller reads the cancellation samples from the second FPGA. We implement the Spectrum Analyzer block along with the trough detection, spectrum shape sensing and amplitude matching algorithms on the central controller. The computed correction ratio is continuously fed back to the second FPGA using the control lines between the central controller and the FPGAs. The keystroke detection algorithm is implemented on the central controller as well.

The central controller synchronizes the baseband and RF clocks of the two receive RF chains. The SDR platform provides a backplane of 8 wires to interface each FPGA to the central controller. These wires are used for both the purpose of data transfer as well as for providing control signals. We share the internal reference clock of the SDR platform on one of the lines in the backplane. Each receive chain synchronize their RF and baseband clocks to this reference. We also implement timing synchronization between the two FPGAs to render consistency to the sample offset between the two ADCs.

Finally, we implement the transmitter using the same hardware configuration i.e. Virtex-5 based FPGA for baseband processing and an NI-5791 front-end. We operate the transmitter in the 2.4 GHz ISM band with a transmission bandwidth of 20 MHz. The receiver hacker is agnostic to the underlying PHY design of the transmitter. None the less, we implement an OFDM based PHY to mimic WiFi transmissions.

7. EVALUATION

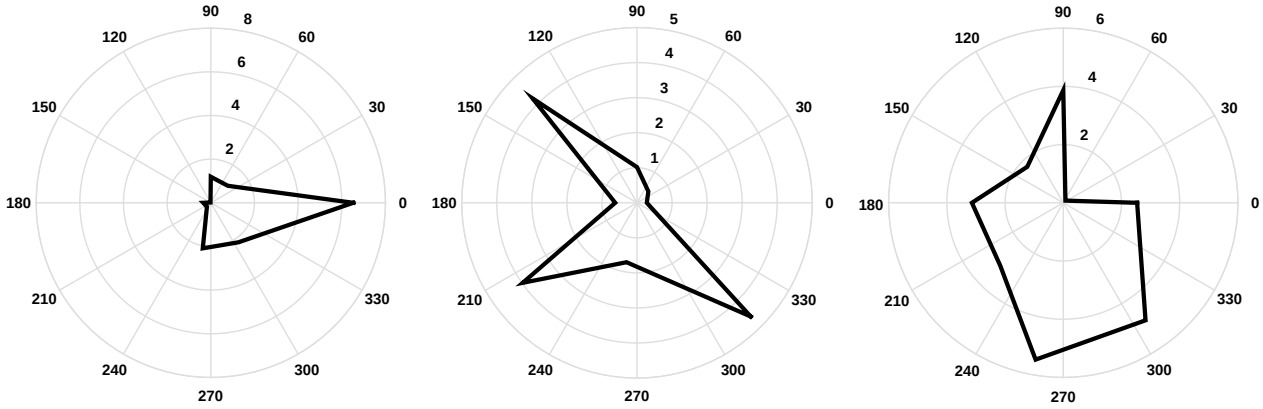
This section presents a thorough evaluation of our receiver system. In the first half of this section, we evaluate the various design decisions that affect the performance of the system. In the second half, we present a holistic evaluation of the system performance.

7.1 Microbenchmark Evaluation

In this section, we evaluate the design decisions of all parameters that affect key differentiation. Specifically, the design of algorithms to detect specific keystrokes is discussed in the next subsection. Here we evaluate the design of only those parameters that influence the ability to observe distinct signals for different closely spaced keys.

7.1.1 Orientation Effect

We set out to measure the sensitivity of the hacker for different positions of the hacker relative to the user, in a given environment. For this exercise, we equip our hacker with two antennas. We place the hacker 3m from the user and move the device to different positions along a circle centered around the user. We sweep the circle in eight equi-angular steps. For each position of the hacker, we have the user type adjacent keys '4' and '5'. We measure the difference in the respective observed frequency troughs at the hacker. We plot the measured sensitivity for each position of the hacker in Fig. 11(a). As indicated in the figure, while in the initial position, we observed a sensitivity of ≈ 6 MHz, as we moved around the user we noticed that the sensitivity significantly depreciates for the same keystrokes. Specifically, when the hacker is placed diametrically opposite to the initial position



(a) Moves aroud the user.

(b) Rotates across different orientations at a given location.

(c) Moves Around User For Different Orientations.

Figure 11: Effect of Orientation on Receiver Sensitivity: The receiver is moved around the user as described. For each receiver location the sensitivity for given keystrokes is measured. Figs. 11(a),11(b),11(c) plot the sensitivity at the receiver for different locations and orientations relative to the user. The units of sensitivity is MHz.

(at an angle of 180 degrees), we observe that the hacker can barely distinguish between the two key strokes and that the sensitivity is orders of magnitude below 1 MHz.

To investigate the significant sensitivity degradation, we placed the hacker device at the 180 degree position again. This time, we rotated the hacker device at this position through 2π radians, again in eight equi-angular steps. This changes the orientation of the hacker's antennas relative to the user. For each position we again have the user repeat the same keystrokes, namely '4' and '5'. We measure the sensitivity at each of these orientations and plot the sensitivity in Fig. 11(b). The significance of the orientation between the hacker and the user is evident. While at our initial orientation the sensitivity is very poor when we rotate the hacker by 45 degrees in the clock wise direction, we notice a sensitivity of ≈ 4 MHz. Further, we discover two more orientations at which we measure decent sensitivity.

Finally, we repeat the experiment we initially set out to do i.e. measure the hacker's sensitivity for different positions around the user. This time, however, we increase the distance between the hacker and user to 4m. We again move the hacker around the user in a circle. At each position we rotate the hacker across four orientations. We plot the maximum sensitivity observed at each position in Fig. 11(c). We note that at except one position, we observe appreciable sensitivity of the order of ≈ 3 MHz for at least one of the four orientations around the user. The increase in distance between the user and the hacker does *not* significantly degrade the sensitivity.

The main takeaway from the above set of experiments is that the orientation between the hacker and user is more critical to the hacker's performance than the absolute distance between them. The orientation is a function of the placement of the hacker's antennas and the location of successive key presses. For a given pair of receive antennas at the hacker, the orientation of the user's keystrokes relative to the hacker can change depending on the keystrokes pressed by the user. For instance, if the orientation between the hacker and the user is consistent when the user presses key 'A' and then key 'S', then the hacker's orientation rotates by ≈ 90 degrees when the user presses key 'A' and then key 'Z'. This can cause the hacker to have different sensitivity performance between differentiating key 'A' from key 'S' and key 'A' from key 'Z'. To make our hacker less sensitive to the orientation of the hacker relative to the user, we can increase the number of antennas at the hacker. We study the increase in reliability of key detection in the presence of additional antennas later in this section.

7.1.2 *Multiple Antennas*

The previous experiments established the sensitivity of the hacker system's performance to the orientation between the hacker and the user. To make the hacker robust to different orientations, we require multiple antennas at the hacker. As described in the previous section, for a given location of the hacker, the orientation of the hacker relative to the user can change depending on the keystrokes pressed by the user. Thus, for a given pair of antennas at the hacker, while they might show high sensitivity for a pair of keys pressed by the user, the same pair can demonstrate poor sensitivity for another set of keys at a different orientation.

In a hacker with just two antennas, different keystrokes are distinguished by the difference in sensitivity. For example, say the hacker detects a trough at frequency F_A when key A is pressed and at frequency F_S when key S is pressed. To make the example extreme, assume the hacker detects a trough at frequency F_A even when key Z is pressed. Now assume the hacker is aware that the current key press is A. If the hacker observes the trough at the next keystroke detection at frequency F_S then the hacker can detect the keystroke as S. With reference to A, the keystroke S can then be mapped to the number $F_S - F_A$. However, if at the next keystroke detection the trough is located again at frequency F_A then the hacker cannot decide between keys A and Z since relative to F_A both these keys are mapped to 0.

Now assume we add a third antenna to this hacker device. The third antenna is placed such that the three antennas are not co-linear. The design process of the hacker system with

three antennas is simply an extension of the design process illustrated in Fig. 5 for a two antenna hacker system. We simply add an additional DMA channel between the third FPGA and the first FPGA to transfer the samples from the ADC of the third antenna to the first FPGA. This stream of samples is subjected to the same delay and correction ratio as the samples from the second antenna. Since the orientation of the first and third antenna pair relative to the user is different from that of the first and second antenna pair, the sensitivity observed on this pair of antennas is different for the same set of keystrokes. Say, for this pair of antennas, the relative mapping of key S relative to key A is $F'_S - F'_A$ and that of key Z relative to key A is $F'_Z - F'_A$. Combining the mapping from both antenna pairs the cumulative mapping of key S relative to key A is an array $[F_S - F_A, F'_S - F'_A]$ and key Z relative to key A is $[0, F'_Z - F'_A]$. Intuitively, one would expect that the probability that relative to a given keystroke the inability of distinguishing different keystrokes significantly reduces as the number of antennas increase.

Extending the discussion to a hacker with N antennas, each keystroke can be mapped relative to a given keystroke with an array of N-1 elements $[F^1_{Diff}, F^2_{Diff}, \cdots, F^{N-1}_{Diff}]$ where F^k_{Diff} is the sensitivity of that keystroke relative to the given keystroke observed on the pair of antennas $1, K+1$. Finally, following the observations in Fig. 10, we must note that given the trough frequency array of a particular keystroke, the mapping of other keystrokes relative to this keystroke remains constant under all channel conditions. However, the absolute trough frequency of the reference keystroke itself may change over time.

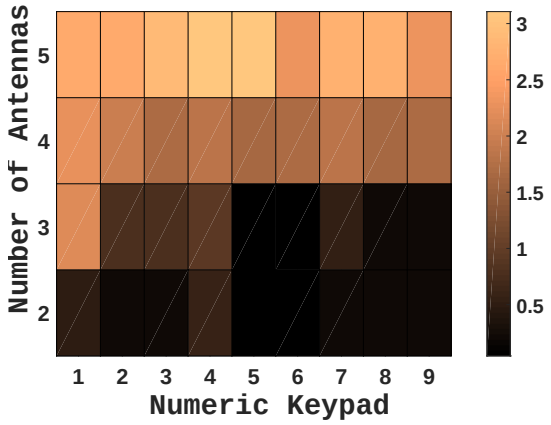


Figure 12: Minimum Sensitivity (MHz) of numeric keypad relative to each key on the numeric keypad as a function of the number of antennas.

We seek to evaluate the ability of the hacker to distinguish between different keystrokes relative to a given keystroke as a function of the number of antennas on the hacker. For this experiment we limit the key presses to the number pad on the right side of a typical keyboard. We have a user type an initial number (between 1-9). Using that key as a reference we ask the user to type the remaining number keys between 1-9 in a random order alternating each number key press with a key press of the selected initial reference number key. We ask the user to repeat with this process by selecting a different initial reference key. We repeat the experiment with five users for a fixed location and orientation of the hacker relative to the user device. We compute the sensitivity between the selected reference key and each number key. We compute the minimum sensitivity for a given reference key and record it. For a generic N antenna hacker device, we define the sensitivity between a given keystroke and the reference keystroke by the Euclidean distance between the mapping of the given keystroke and that of the reference keystroke. By the definition of our mapping, the reference keystroke always maps to an array with $N-1$ zeros. The sensitivity then is simply $\sqrt{F_1^2 + F_2^2 + \cdots F_{N-1}^2}$ where F_i is the measured sensitivity between the antenna pair $1, i+1$.

For each user, we repeat the experiment by equipping the hacker with the effect of 2,3,4 and 5 antennas respectively. For each setup of the hacker we place the antennas at a distance of approximately 6 cm from each other in a compact configuration. The results are indicated in Fig. 12. We observe that once the receiver is equipped with 4 or more antennas, then the minimum sensitivity is at least 2 MHz to all the number pad keys independent of the initial reference key selection. This affirms our intuition that additional antennas makes the hacker robust to different orientations. Further, increasing the number of antennas also increases the sensitivity of the hacker thus increasing its ability to distinguish between closely spaced keys. As stated before, the sensitivity between a pair of keys is independent of channel variations for given location and orientation of the hacker relative to the user.

7.2 System Evaluation

This section evaluates the performance of the hacker system. Following the observations from the multi-antenna experiment recorded in Fig. 12, we imlpement our hacker system with five antennas. We fix the location of the user and the hacker. For all experiments in this section, we fix the location of the hacker at a distance of 5m from the user. All experiments are performed inside our lab furnished with metal furniture and other strong sources of multipath. We repeat all the experiments in this section with five users. Each user types one key at a time using a single finger at slower than average typing speeds.

Robustness to Channel Variation: In the first experiment, we seek to evaluate the key detection capability of our hacker system under varying channel conditions. In this experiment, we have the user repeatedly press the number pad key 5. While the user is typing, we introduce variation in the channel by having people walk in the lab in the vicinity of the user-hacker system. We must note that we have the people move outside the line-of-sight of the user-hacker channel. However, it still effects the multipath profile of the channel. The user himself is in a *rest* position while typing. Here, by rest position we mean that while the user can move intermittently (say his hand or head), the user is not constantly in motion. Initially, we train the hacker with the sensitivity of the number pad keys from 1-9 relative to 5 by having the user type keys sequentially from 1 to 9. Further, we train the receiver with an initial absolute mapping of key 5. In other words, the hacker has an initial reference of trough locations observed on each pair of antennas when it is aware that the key being pressed is 5. Let us denote this initial absolute mapping of key 5 with V_5. Since we have five antennas, each mapping of the number pad keys relative to key 5 is an array of length 4. Let us denote the mapping of number pad key K relative to 5 with $V_{K,5}$. We have the

user repeat this sequence twice to train the receiver.

Because of the change in the channel between the user and the hacker, the mapping of the key recorded by the hacker can change each time. This can potentially cause the repeated key press 5 sometimes be interpreted as key press of an adjacent key. Further, since key 5 is in the middle of the number pad, the probability of detecting a 5 with one of its neighbors is highest among the keys in the number pad. Let us assume the current mapping recorded by the hacker is V'. Let V' be different from V_5. We measure the Euclidian distance between V' and V_5. For majority of the key presses, we notice that this distance is close to 0 and significantly lower than the sensitivity between key 5 and any other key on the number pad. This results in the hacker decoding that the received mapping is for key 5. Once the hacker detects the key press as key 5, it attributes the difference in the observed mapping from the reference mapping to variation in the channel. The hacker subsequently updates its reference mapping after accounting for channel variation. With each user we repeat the experiment for fifty runs and observe that we detect the key press as 5 correctly on 90% of the runs. In the few instances where the channel change is large, key press 5 can be misinterpreted as one of the adjacent keys. This result is illustrated in Fig. 13.

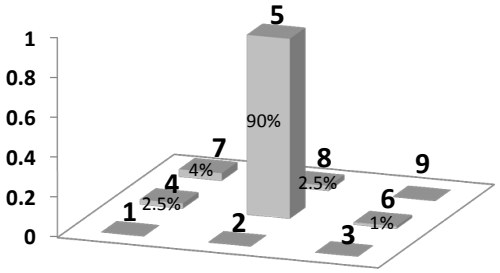


Figure 13: Detecting repeated keystroke '5' in a changing channel.

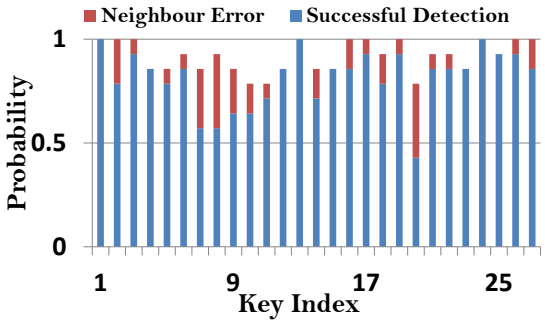


Figure 14: Keystroke detection accuracy given full training.

Complete Training: Next, we evaluate the effectiveness of the hacker to detect different keystrokes spread across the keyboard. We perform this experiment under the same conditions as before. Like before, for the partial and complete training experiments, we have people move in the lab during the experiment. Specifically, on a typical QWERTY keyboard, the hacker intends to detect all the letter keys as well as the ',' key adjacent to key 'M'. First, to evaluate the feasibility of a wireless keystroke detection technique, we assume the hacker device is trained for each of the 27 keys. We have the user type each of these keys *once* to train the hacker. For each of these key presses the hacker constructs an array to map the observed trough locations to the key press. Once trained, the user types from these keys randomly one key at a time. Each user types 270 keys such that he types each key ten times. However, the keys are typed in a random order. The hacker estimates the keystroke by mapping the readings to the key for which the Euclidian distance between the observed trough locations and the reference readings of the key is minimum. Similar to the previous experiment, in the presence of a perfectly static channel, one would expect the Euclidian distance between the estimated key and the reference key to be 0. We attribute change in the Euclidian distance to noise and changes in the environment. Once we assign the key, we update the reference readings of all the keys such that the Euclidian distance between the observed reading and the new reference of the assigned key is zero. The hacker's performance in estimating each of the 27 keys given full, short training is given in Fig. 14. The hacker has an average detection accuracy of 81.48%. In most instances where the key was misinterpreted, it was estimated to be one of its horizontally adjacent keys. The hacker estimates either the key or its neighbor with an accuracy of 91.8%.

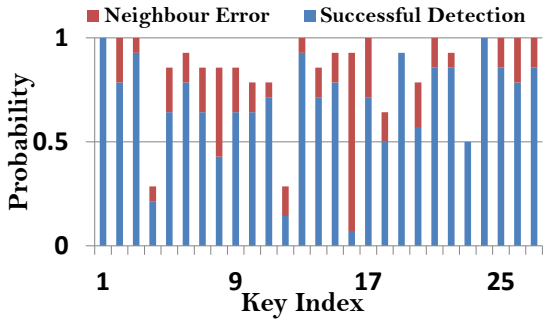


Figure 15: Keystroke detection accuracy given partial training data. 13 equally spaced keys are input once to provide initial training data.

Partial Training: Finally, we evaluate the hacker's keystroke detection ability in the event of limited training inforamtion. In this experiment, the user types 13 of the 27 keys (leaving every alternate key) *once* to provide initial training information to the hacker. Then, the user proceeds to type from the 27 keys in a random order. Like the previous experiment, each user types 270 keys in total. The hacker estimates a reference for those keystrokes that do not have a direct training input by extrapolation across neighboring keys with training input. Fig. 15 indicates the key detection accuracy in this scenario. The average detection accuracy is $\approx 70\%$ and the detection accuracy of either the intended keystroke or its neighbor is 85%. Although majority of the keystrokes are detected with reasonable accuracy, some of the keystrokes are almost always misinterpreted due to a noisy training input either at that key or its neighbor. While the detection accuracy at first glance might seem a little underwhelming, the word detection accuracy can potentially be significantly higher with simply an initial short training keystroke input of ≈ 10 characters.

8. DISCUSSION AND LIMITATIONS

In this section we discuss the limitations of our implementation.

1. **Typing at faster speeds:** In our experiments, the users type at a slow and lower than average speed as indicated in Fig. 9. In our setup, we have the algorithms for keystroke detection and locating the trough in the cancellation spectrum on the central controller. The samples from different ADCs are transferred across FPGAs and the subtracted samples at one of these FPGAs are then forwarded to the central controller. The central controller then computes the power spectrum on these samples and performs subsequent keystroke detection algorithms. In our implementation, the central controller loads samples over multiple symbols for each power spectrum computation. The speed is limited by our implementation time which includes the time to move the samples in addition to the processing delay of our power spectrum computation and keystroke detection algorithms on the central controller. This limits our implementation from detecting typing at faster speeds reliably. The implementation speed of our algorithm can be significantly improved by moving its execution to the FPGA. This should enable the receiver to detect faster keystrokes.

2. **Packetized WiFi Transmissions:** Fundamentally, our technique requires the user's device to transmit a wireless signal in order to facilitate keystroke identification. Thus, if the user is not transmitting a wireless signal, the receiver cannot decode the keystrokes. The experiments in Sec. 7 were carried out using *continuous* transmissions from a SDR transmitter. However, WiFi transmissions are packetized and not continuous. Our trough identification and location algorithm works on the cancellation spectrum. As the transmission becomes increasingly discontinuous and packetized the observed cancellation spectrum fluctuation increases. We observe negligible spectrum fluctuations for UDP transmissions with an inter packet latency of up to 10ms. As the packets become less continuous and the spectrum fluctuation increases, our implementation will not detect the keystrokes reliably due to its implementation speed for reasons discussed above. However, if the packet transmission is extremely scarce, for example at rates of 1 packet/second, then our receiver will be fundamentally not be able to detect the user keystrokes.

3. **Typing with two hands:** In our experiments, the users type one key at a time with the index finger with just one hand. We have this restriction to ensure consistency in the way a user types a certain key over both the training phase and the random key input phase. As observed in our experiments, this setting ensures that as long as the user in a rest position, the recording of each keystroke is repeatable even in environments where there is movement around the user. However, with longer training sequences and/or more sophisticated self-learning classifiers and algorithms, we believe there is no fundamental limit in preventing the system from decoding users that type with both hands.

4. **User Movement:** In the experiments in Sec. 7, each user maintains a rest position while typing. As illustrated in Fig. 10, when the user introduces movement (say a movement of the hand or the head), the sensitivity of the receiver is retained as long as the position of the user device and the hacker remains the same. For this reason the training information recorded in the experiments is the sensitivity between pairs of antennas for each keystroke. Thus, while user movements can change the absolute trough locations, the keystroke is still reliably identified by our algorithm since the *relative* trough location does not change.

5. **Change in keyboard orientation:** Currently, our system does not support a scenario wherein the user changes the position of the keyboard after the training phase. Changing the position causes the observed recordings to change from the associated trained readings. In our implementation, this will cause the observed recording to be wrongly interpreted as a different key. However, using simple algorithms to examine a series of keystrokes (which can be interpreted as letters forming a word), we can detect persistent unusual patterns and discard the available training information. This will prevent the receiver from misinterpreting the detected keystrokes. With more sophisticated self-learning algorithms, we can potentially refine the training information with the observed recordings of known keys. (For example: keys such as the space bar and the vowels are used more frequently while typing and the frequency of similar observations can be used to determine the key and update its training information.)

6. **Impact of User Antenna Placement:** If the user's antenna was directional such that *none* of the radiation power is directed towards the keyboard and the user's hands then our receiver can fundamentally not detect the keystrokes. Thus, we require some wireless energy transmitted by the user to the receiver to be reflected off the user's keyboard/fingers and hand. We carry out our experiments using a single omni-directional antenna to transmit the continuous wireless signals for the user. We placed the antenna above the user keyboard at an angle typical of a laptop (Laptops typically have the Wi-Fi antennas on the top edge of the screen). For different positions of the antenna relative to the keyboard (within the constraint of typical laptop settings) we did not observe a dependency between the antenna placement and the keystroke detection of our receiver.

7. **Multi-antenna transmitter:** Our experiments used a single antenna transmitter. As 802.11 evolves, portable devices will have more antennas. If multiple antennas transmit the same symbol (for diversity gain) then our existing receiver system should still be able to detect the keystrokes by treating the multiple antennas as an effective single antenna. However, if the antennas transmit independent data streams, then our current receiver system design as is will not support keystroke detection. We can deploy receive beamforming techniques to isolate the individual streams to perform keystroke detection but this will sacrifice the PHY-agnostic characteristic of our current design.

9. CONCLUSION

In this paper, our goal was to demonstrate keystroke identification using wireless transmissions. We believe that this

exposes a serious security threat even for current commodity wireless users. As described before, we envision such an attack to happen in public venues where the attacker can be far enough to look unsuspicious and close enough (within 5m) to decipher key strokes. Our system can be scaled in many ways to improve the accuracy of key detection and improve distance range. We highlight these possible improvements in this section.

Spectra other than 2.4G: Wi-Fi also operates in 5.8GHz ISM band. Higher center frequency will allow more antennas to be easily mounted on current nodes. This will improve our accuracy. In fact, Wi-Fi can operate in wider bandwidths than 20MHz; 40MHz, 80MHz and even more. As Section 3 showed, wider bandwidth will allow more control over sensitivity. Thus, higher precision in keystroke detection will be possible.

Keystroke identification ⇒ hacking: This paper demonstrated the feasibility of identifying keystrokes and also identifying which key was pressed. Clearly, the next step is to use this attack to hack username/password, credit card details and so on. There exist immense machine learning algorithms that can assist with this step. Such techniques can also deal with intermittent keystroke data from a given user, similar to auto-correct on smart phones. However, autofill options can make it more difficult for such techniques to carryout a fully successful hack. We leave this as future work.

Moving forward, we should rethink ways to protect against such security and privacy issues in the future. One possible research is to look for new user interfaces that allow users to input data into the system reliably. Clearly, biometric scans are not prone to our attack, while traditional username/password-type inputs are.

Acknowledgments

We would like to thank our shepherd, Shyamnath Gollakota, and the anonymous reviewers for their comments and suggestions to improve our paper.

10. REFERENCES

[1] `http://samy.pl/keysweeper/`.

[2] Adib, F., Kabelac, Z., Katabi, D., and Miller, R. C. 3d tracking via body radio reflections. In *Usenix NSDI* (2013), vol. 14.

[3] Adib, F., and Katabi, D. See through walls with wifi! In *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM* (2013), ACM, pp. 75–86.

[4] Bahl, P., and Padmanabhan, V. N. Radar: An in-building rf-based user location and tracking system. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE* (2000), vol. 2, Ieee, pp. 775–784.

[5] Bharadia, D., and Katti, S. Full duplex mimo radios. *Self 1*, A2 (2014), A3.

[6] Bharadia, D., McMilin, E., and Katti, S. Full duplex radios. In *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM* (2013), ACM, pp. 375–386.

[7] Choi, J. I., Jain, M., Srinivasan, K., Levis, P., and Katti, S. Achieving single channel, full duplex wireless communication. In *Proceedings of the sixteenth annual international conference on Mobile computing and networking* (2010), ACM, pp. 1–12.

[8] Jain, M., Choi, J. I., Kim, T., Bharadia, D., Seth, S., Srinivasan, K., Levis, P., Katti, S., and Sinha, P. Practical, real-time, full duplex wireless. In *Proceedings of the 17th annual international conference on Mobile computing and networking* (2011), ACM, pp. 301–312.

[9] Joshi, K. R., Hong, S. S., and Katti, S. Pinpoint: Localizing interfering radios. In *NSDI* (2013), pp. 241–253.

[10] Kellogg, B., Talla, V., and Gollakota, S. Bringing gesture recognition to all devices. In *Usenix NSDI* (2014), vol. 14.

[11] Pu, Q., Gupta, S., Gollakota, S., and Patel, S. Whole-home gesture recognition using wireless signals. In *Proceedings of the 19th annual international conference on Mobile computing & networking* (2013), ACM, pp. 27–38.

[12] Vuagnoux, M., and Pasini, S. Compromising electromagnetic emanations of wired and wireless keyboards. In *USENIX security symposium* (2009), pp. 1–16.

[13] Wang, G., Zou, Y., Zhou, Z., Wu, K., and Ni, L. M. We can hear you with wi-fi! In *Proceedings of the 20th annual international conference on Mobile computing and networking* (2014), ACM, pp. 593–604.

[14] Wang, J., and Katabi, D. Dude, where's my card?: Rfid positioning that works with multipath and non-line of sight. In *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM* (2013), ACM, pp. 51–62.

[15] Wang, J., Vasisht, D., and Katabi, D. Rf-idraw: virtual touch screen in the air using rf signals. In *Proceedings of the 2014 ACM conference on SIGCOMM* (2014), ACM, pp. 235–246.

[16] Wang, J., Zhao, K., Zhang, X., and Peng, C. Ubiquitous keyboard for small mobile devices: harnessing multipath fading for fine-grained keystroke localization. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services* (2014), ACM, pp. 14–27.

[17] Wang, Y., Liu, J., Chen, Y., Gruteser, M., Yang, J., and Liu, H. E-eyes: device-free location-oriented activity identification using fine-grained wifi signatures. In *Proceedings of the 20th annual international conference on Mobile computing and networking* (2014), ACM, pp. 617–628.

[18] Xiong, J., and Jamieson, K. Arraytrack: A fine-grained indoor location system. In *NSDI* (2013), pp. 71–84.

[19] Xiong, J., Jamieson, K., and Sundaresan, K. Synchronicity: Pushing the envelope of fine-grained localization with distributed mimo. In *Proceedings of the 1st ACM Workshop on Hot Topics in Wireless* (2014), HotWireless '14.