Keystroke Recognition Using WiFi Signals

Kamran Ali[†] Alex X. Liu[†‡] Wei Wang[‡] Muhammad Shahzad[†]
[†]Dept. of Computer Science and Engineering, Michigan State University, USA
[‡]State Key Laboratory for Novel Software Technology, Nanjing University, China
[†]{alikamr3,alexliu,shahzadm}@cse.msu.edu, [‡]ww@nju.edu.cn

ABSTRACT

Keystroke privacy is critical for ensuring the security of computer systems and the privacy of human users as what being typed could be passwords or privacy sensitive information. In this paper, we show for the first time that WiFi signals can also be exploited to recognize keystrokes. The intuition is that while typing a certain key, the hands and fingers of a user move in a unique formation and direction and thus generate a unique pattern in the time-series of Channel State Information (CSI) values, which we call CSI-waveform for that key. In this paper, we propose a WiFi signal based keystroke recognition system called WiKey. WiKey consists of two Commercial Off-The-Shelf (COTS) WiFi devices, a sender (such as a router) and a receiver (such as a laptop). The sender continuously emits signals and the receiver continuously receives signals. When a human subject types on a keyboard, WiKey recognizes the typed keys based on how the CSI values at the WiFi signal receiver end. We implemented the WiKey system using a TP-Link TL-WR1043ND WiFi router and a Lenovo X200 laptop. WiKey achieves more than 97.5% detection rate for detecting the keystroke and 96.4% recognition accuracy for classifying single keys. In real-world experiments, WiKey can recognize keystrokes in a continuously typed sentence with an accuracy of 93.5%.

Categories and Subject Descriptors

C.2.1 [**Network Architecture**]: Wireless Communications; D.4.6 [**Security and Protectione**]: Keystroke recovery

Keywords

Gesture recognition; Wireless security; Keystroke recovery; Channel State Information; COTS WiFi devices

1. INTRODUCTION

Keystroke privacy is critical for ensuring the security of computer systems and the privacy of human users as what being types could be passwords or privacy sensitive information. The research community has studied various ways to recognize keystrokes, which can be classified into three categories: acoustic emission based approaches, electromagnetic emission based approaches, and vision based approaches. Acoustic emission based approaches recognize keystrokes based on either the observation that different keys in a keyboard produce different typing sounds [**?**, **?**] or the observation that the acoustic emanations from different keys arrive at different surrounding smartphones at different time as the keys are located at different places in a keyboard [**?**]. Electromagnetic emission based approaches recognize keystrokes based on the observation that the electromagnetic emanations from the electrical circuit underneath different keys in a keyboard are different [**?**]. Vision based approaches recognizes keystrokes using vision technologies [**?**].

In this paper, we show for the first time that WiFi signals can also be exploited to recognize keystrokes. WiFi signals are pervasive in our daily life at home, offices, and even shopping centers. The key intuition is that while typing a certain key, the hands and fingers of a user move in a unique formation and direction and thus generate a unique pattern in the time-series of Channel State Information (CSI) values, which we call *CSI-waveform*, for that key. The keystrokes of each key introduce relative unique multi-path distortions in WiFi signals and this uniqueness can be exploited to recognize keystrokes. Due to the high data rates supported by modern WiFi devices, WiFi cards provide enough CSI values within the duration of a keystroke to construct a high resolution CSI-waveform for each keystroke.

We propose a WiFi signal based keystroke recognition system called WiKey. WiKey consists of two Commercial Off-The-Shelf (COTS) WiFi devices, a sender (such as a router) and a receiver (such as a laptop), as shown in Figure **??**. The sender continuously emits signals and the receiver continuously receives signals. When a human subject types in a keyboard, on the WiFi signal receiver end, WiKey recognizes the typed keys based on how the CSI value changes. CSI values quantify the aggregate effect of wireless phenomena such as fading, multi-paths, and Doppler shift on the wireless signals in a given environment. When the environment changes, such as a key is being pressed, the impact of these wireless phenomena on the wireless signals change, resulting in unique changes in the CSI values.

There are three key technical challenges. The first technical challenge is to segment the CSI time series to identify the start time and end time of each keystroke. We studied the characteristics of typical CSI-waveforms of different keystrokes and observed that the waveforms of different keys show a similar rising and falling trends in the changing

MobiCom'15, September 7–11, 2015, Paris, France.

DOI: http://dx.doi.org/10.1145/2789168.2790109.

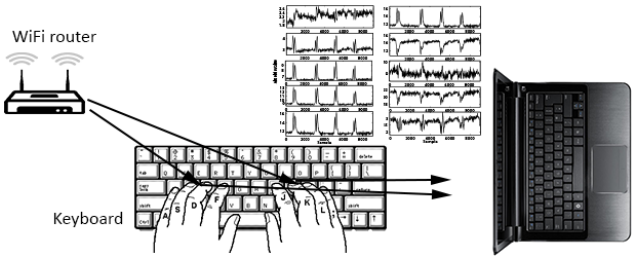


Figure 1: WiKey System

rate of CSI values. Based on this observation, we design a keystroke extraction algorithm that utilizes CSI streams of all transmit-receive antenna (TX-RX) pair pairs to determine the approximate start and end points of individual keystrokes in a given CSI-waveform by continuously matching the trends in CSI time series with the experimentally observed trends using a sliding window approach.

The second technical challenge is to extract distinguishing features for generating classification models for each of the 37 keys (10 digits, 26 alphabets and 1 space-bar). As the keys on a keyboard are closely placed, conventional features such as maximum peak power, mean amplitude, root mean square deviation of signal amplitude, second/third central moment, rate of change, signal energy or entropy, and number of zero crossings cannot be used because the values of these features for adjacent keys are almost identical. To address this challenge, we use the CSI-waveform shapes of each key from each TX-RX antenna pair as features. As the waveforms for each key contain a large number of samples, we apply the Discrete Wavelet Transform (DWT) technique on these waveforms to reduce the number of samples while keeping the shape preserving time and frequency domain information intact. We use the waveforms resulting from the DWT of individual keystrokes as their shape features.

The third technical challenge is to compare shape features of any two keystrokes. The midpoints of extracted CSI-wavforms of different keystrokes rarely align with each other because the start and end points determined by extraction algorithm are never exact. Moreover, the lengths of different keystroke waveforms also differ because the duration of pressing any key is often different. Consequently, the midpoints and lengths of shape features do not match either. Another issue is that the shape of different keystroke waveforms of the same key are often distorted versions of each other because of slightly different formation and direction of motion of hands and fingers while pressing that key. Thus, two shape features cannot be compared using standard measures like correlation coefficient or Euclidean distance. To address this challenge, we use the Dynamic Time Warping (DTW) technique to quantify the distance between the two shape features. DTW can find the minimum distance alignment between two waveforms of different lengths.

The key novelty of this paper is on proposing the first WiFi signal based keystroke recognition approach. Some recent work uses CSI values to recognize various *macro* aspects of human movements such as falling down [**?**], household activities [**?**], detection of human presence [**?**], and estimating the number of people in a crowd [**?**]. These schemes extract coarse grained information from the CSI values to recognize the macro-movements such as falling down or recognizing fullbody/limb gestures. They cannot be directly adapted to recognize keystrokes because such coarse grained information does not capture the minor variations in the CSI values caused by human *micro*-movements such as those of hands and fingers while typing. Some recent work, namely WiHear, uses CSI values to extract the micro-movements of mouth to recognize 9 syllables in the spoken words [**?**]. However, WiHear uses special hardware including directional antennas and stepper motors to direct WiFi beams towards speaker's mouth and extract the micro-movements. We implemented the WiKey system using COTS devices, i.e. a TP-Link TL-WR1043ND WiFi router and a Lenovo X200 laptop with Intel 5300 WiFi NIC. In the evaluation process, we build a keystroke database of 10 human subjects with IRB approval. WiKey achieves more than 97.5% detection rate for detecting the keystroke and 96.4% recognition accuracy for classifying single keys. In real-world experiments, WiKey can recognize keystrokes in a continuously typed sentence with an accuracy of 93.5%.

In this paper, we have shown that fine grained activity recognition is possible by using COTS WiFi devices. Thus, the techniques proposed in this paper can be used for several HCI applications. Examples include zoom-in, zoom-out, scrolling, sliding, and rotating gestures for operating personal computers, gesture recognition for gaming consoles, in-home gesture recognition for operating various household devices, and applications such as writing and drawing in the air. Other than being a potential attack, our WiKey technology can be potentially used to build virtual keyboards where human users type on a printed keyboard.

2. RELATED WORK

2.1 Device Free Activity Recognition

Device-free activity recognition solutions use the variations in wireless channel to recognize human activities in a given environment. Existing solutions can be grouped into three categories: (1) Received Signal Strength (RSS) based, (2) CSI based, and (3) Software Defined Radio (SDR) based.

RSS Based: Sigg *et al.* proposed activity recognition schemes that utilize RSS values of WiFi signals to recognize four activities including crawling, lying down, standing up, and walking [**?**, **?**]. They achieved activity recognition rates of over 80% for these four activities. To obtain the RSS values from WiFi signals, they used USRPs, which are specialized hardware devices compared to the COTS WiFi devices that we used in our work. While RSS values can be used for recognizing macro-movements, they are not suitable to recognize the micro-movements such as those of fingers and hands in keyboard typing because RSS values only provide coarse-grained information about the channel variations and do not contain fine-grained information about small scale fading and multi-path effects caused by these micro-movements.

CSI Based: CSI values obtained from COTS WiFI network interface cards (NICs) (such as Intel 5300 and Atheros 9390) have been recently proposed for activity recognition [**?**, **?**, **?**, **?**, **?**, **?**] and localization [**?**, **?**, **?**]. Han *et al.* proposed WiFall that detects fall of a human subject in an indoor environment using CSI values [**?**]. Zhou *et al.* proposed a passive human detection scheme which exploits multi-path variations for detecting human presence in an indoor environment using CSI values [**?**]. Zou *et al.* proposed Electronic Frog Eye that counts the number of people in a crowd using

CSI values by treating the people reflecting the WiFi signals as "virtual antennas" [?]. Wang *et al.* proposed E-eyes that exploits CSI values for recognizing household activities such as washing dishes and taking a shower [?]. Nandakumar *et al.* leverage the CSI and RSS information from off-the-shelf WiFi devices to classify four arm gestures - push, pull, lever, and punch [?]. The fundamental difference between these schemes and our scheme is that these schemes extract coarse grained features from the CSI values provided by the COTS WiFi NIC to perform these tasks while our proposed scheme refines these CSI to capture fine grained variations in the wireless channel for recognizing keystrokes. Wang *et al.* propose WiHear that uses CSI values recognizes the shape of mouth while speaking to detect whether a person is uttering one of a set of nine predefined nine syllables [?]. While WiHear can capture the micro-movements of lips, it uses special purpose directional antennas with stepper motors for directing the antenna beams towards a person's mouth to obtain a clean signal for recognizing mouth movements. In contrast, our proposed scheme does not use any special purpose equipment and recognizes the micro-movements of fingers and hands using COTS WiFi NIC.

SDR Based: Researchers have proposed schemes that utilize SRDs and special purpose hardware to transmit and receive custom modulated signals for activity recognition [?, ?, ?, ?]. Pu *et al.* proposed WiSee that uses a special purpose receiver design on USRPs to extract small Doppler shifts from OFDM WiFi transmissions to recognize human gestures [?]. Kellogg *et al.* proposed to use a special purpose analog envelop detector circuit for recognizing gestures within a distance of up to 2.5 feet using backscatter signals from RFID or TV transmissions [?] . Lyonnet *et al.* use micro Doppler signatures to classify gaits of human subjects into multiple categories using specialized Doppler radars [?]. Adib *et al.* proposed WiTrack that uses a specially designed frequency modulated carrier wave radio frontend to track human movements behind a wall [?]. Recently, Chen *et al.* proposed an SDR based custom receiver design which can be used to track keystrokes using wireless signals [?]. In contrast to all these schemes, our scheme does not use any specialized hardware or SDRs rather utilizes COTS WiFi NICs to recognize keystrokes.

2.2 Keystrokes Recognition

To the best of our knowledge, there is no prior work on recognizing keystrokes by leveraging variations in wireless signals using commodity WiFi devices. Other than the SDRs based keystroke tracking approach proposed in [?] which uses wireless signals to track keystrokes, researchers have proposed several keystrokes recognition schemes that are based on other sensing modalities such as acoustics [?,?,?,?], electromagnetic emissions [?], and video cameras [?]. Next, we give a brief overview of the other existing schemes that utilize these sensing modalities to recognize keystrokes.

Acoustics Based: Asonov *et al.* proposed a scheme to recognize keystrokes by leveraging the observation that different keys of a given keyboard produce slightly different sounds during regular typing [?]. They used back-propagation neural network for keystroke recognition and fast fourier transform (FFT) of the time window of every keystroke peak as features for training the classifiers. Zhuang *et al.* proposed another scheme that recognizes keystrokes based on the sounds generated during key presses [?]. They used cepstrum features [?] instead of FFT as keystroke features and used unsupervised learning with language model correction on the collected features before using them for supervised training and recognition of different keystrokes. Zhu *et al.* proposed a context-free geometry-based approach for recognizing keystrokes that leverage the acoustic emanations from keystrokes to first calculate the time difference of keystroke arrival and then estimate the physical locations of the keystrokes to identify which keys are pressed [?].

Electromagnetic Emissions Based Vuagnoux *et al.* used a USRP to capture the electromagnetic emanations while pressing the keys [?]. These electromagnetic emanations originated from the electrical circuit underneath each key in conventional keyboards. The authors proposed to capture the entire raw electromagnetic spectrum and process it to recognize the keystrokes. Unfortunately, this scheme is highly susceptible to background electromagnetic noise that exists in almost all environments these days such as due to microwave ovens, refrigerators, and televisions.

Video Camera Based Balzarotti *et al.* proposed ClearShot that processes the video of a person typing to reconstruct the sentences (s)he types [?]. The authors propose to use context and language sensitive analysis for reconstructing the sentences.

3. CHANNEL STATE INFORMATION

Modern WiFi devices that support IEEE 802.11n/ac standard typically consist of multiple transmit and multiple receive antennas and thus support MIMO. Each MIMO channel between each transmit-receive (TX-RX) antenna pair of a transmitter and receiver comprises of multiple subcarriers. These WiFi devices continuously monitor the state of the wireless channel to effectively perform transmit power allocations and rate adaptations for each individual MIMO stream such that the available capacity of the wireless channel is maximally utilized [?]. These devices quantify the state of the channel in terms of CSI values. The CSI values essentially characterize the *Channel Frequency Response* (CFR) for each subcarrier between each transmit-receive (TX-RX) antenna pair. As the received signal is the resultant of constructive and destructive interference of several multipath signals scattered from the walls and surrounding objects, the disturbances caused by movement of hands and fingers while typing on a keyboard near the WiFi receiver not only lead to changes in previously existing multipaths but also to the creation of new multipaths. These changes are captured in the CSI values for all subcarriers between every TX-RX antenna pair and can then be used to recognize keystrokes.

Let M_T denote the number of transmit antennas, M_R denote the number of receive antennas and S_c denote the number of OFDM sub-carriers. Let $\mathbf{X}_i$ and $\mathbf{Y}_i$ represent the M_T dimensional transmitted signal vector and M_R dimensional received signal vector, respectively, for subcarrier i and let $\mathcal{N}_i$ represent an M_R dimensional noise vector. An $M_R \times M_T$ MIMO system at any time instant can be represented by the following equation.

$$\mathbf{Y}_i = \mathsf{H}_i \mathbf{X}_i + \mathcal{N}_i \quad i \in [1, S_c] \tag{1}$$

In the equation above, the $M_R \times M_T$ dimensional channel matrix H_i represents the Channel State Information (CSI) for the sub-carrier i. Any two communicating WiFi devices estimate this channel matrix H_i for every subcarrier by regularly transmitting a known preamble of OFDM symbols

between each other. For each Tx-Rx antenna pair, the driver of our Intel 5300 WiFi NIC reports CSI values for $S_c = 30$ OFDM subcarriers of the 20 MHz WiFi Channel [**?**]. This leads to 30 matrices with dimensions $M_R \times M_T$ per CSI sample.

4. NOISE REMOVAL

The CSI values provided by commodity WiFi NICs are inherently noisy because of the frequent changes in internal CSI reference levels, transmit power levels, and transmission rates. To use CSI values for recognizing keystrokes, such noise must first be removed from the CSI time series. For this, WiKey first passes the CSI time series from a low-pass filter to remove high frequency noises. Unfortunately, a simple low pass filter does not denoise the CSI values very efficiently. Although strict low-pass filtering can remove noise further, it causes loss of useful information from the signal as well. To extract useful signal from the noisy CSI time series, WiKey leverages our observation that the variations in the CSI time series of all subcarriers due to the movements of hands and fingers are correlated. Therefore, it applies Principal Component Analysis (PCA) on the filtered subcarriers to extract the signals that only contain variations caused by movements of hands. Next, we first describe the process of applying the low-pass filter on the CSI time series and then explain how WiKey extracts hand and finger movement signal using our PCA based approach.

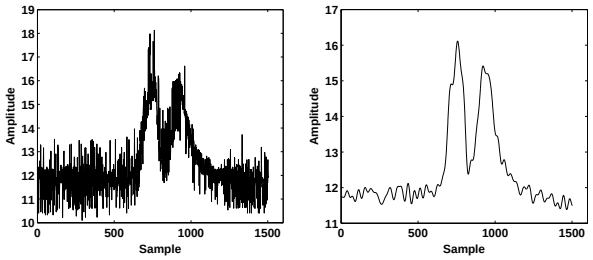
4.1 Low Pass Filtering

The frequency of variations caused due to the movements of hands and fingers lie at the low end of the spectrum while the frequency of the noise lies at the high end of the spectrum. To remove noise in such a situation, *Butterworth* low-pass filter is a natural choice which does not significantly distort the phase information in the signal and has a maximally flat amplitude response in the passband and thus does not distort the hand and finger movement signal much. WiKey applies the Butterworth filter on the CSI time series of all subcarriers in each TX-RX antenna pair so that every stream experiences similar effects of phase distortion and group delay introduced by the filter. Although this process helps in removing some high frequency noise, the noise is not completely eliminated because Butterworth filter has slightly slow fall off gain in the stopband.

We observed experimentally that the frequencies of the variations in CSI time series due to hand and finger movements while typing approximately lie anywhere between 3Hz to 80 Hz. As we sample CSI values at a rate of $F_s = 2500$ samples/s, we set the cut-off frequency ω_c of the Butterworth filter at $\omega_c = \frac{2\pi * f}{F_s} = \frac{2\pi * 80}{2500} \approx 0.2$ rad/s. Figure **??** shows the amplitudes of the unfiltered CSI waveform of a keystroke and Figure **??** shows the resultant from the Butterworth filter. We observe that Butterworth filter successfully removes most of the bursty noises from the CSI waveforms.

4.2 PCA Based Filtering

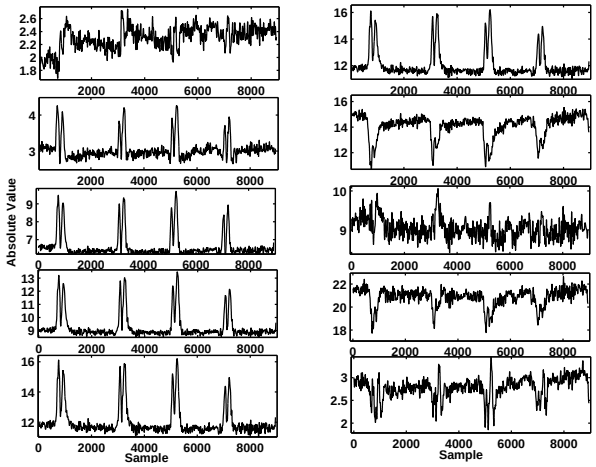
We observed experimentally that the movements of hands and fingers results in correlated changes in the CSI time series for each subcarrier in every transmit-receive antenna pair. Figure **??** plots the amplitudes of CSI time series of 10 different subcarriers for one transmit-receive antenna pair while a user was repeatedly pressing a key. We observe from this figure that all subcarriers show correlated variations in



(a) Original time series (b) Filtered time series

Figure 2: Original and filtered CSI time series

their time series when the user presses the keys. The subcarriers that are closely spaced in frequency show identical variations whereas the subcarriers that farther away in frequency show non-identical changes. Despite non-identical changes, a strong correlation still exists even across the subcarriers that are far apart in frequency. WiKey leverages this correlation and calculates the principal components from all CSI time series. It then chooses those principal components that represent the most common variations among all CSI time series.



(a) # 1,2,3,4,5 (b) # 5,10,15,20,25

Figure 3: Correlated variations in subcarriers

There are two main advantages of using PCA. First, PCA reduces the dimensionality of the CSI information obtained from the 30 subcarriers in each TX-RX stream, which is useful because using information from all subcarriers for keystroke extraction and recognition significantly increases the computational complexity of the scheme. Consequently, PCA automatically enables WiKey to obtain the signals that are representative of hand and finger movements, without having to devise new techniques and define new parameters for selecting appropriate subcarriers for further processing. Second, PCA helps in removing noise from the signals by taking advantage of correlated varations in CSI time series of different subcarriers. It removes the uncorrelated noisy components, which can not be removed through traditional low pass filtering. This PCA based noise reduction is one of the major reasons behind high keystroke extraction and recognition accuracies of our scheme.

5. KEYSTROKE EXTRACTION

WiKey segments the CSI time series to extract the CSI waveforms for individual keystrokes. For this, WiKey operates on the CSI time series resulting from the butterworth filtering. Let $\mathsf{H}_{t,r}(i)$ be an $S_c \times 1$ dimensional vector containing the CSI values of the S_c subcarriers between an arbitrary TX-RX antenna pair $t-r$ for the i^{th} CSI sample. Let $\mathbf{H}_{t,r}$ be an $N \times S_c$ dimensional matrix containing the CSI values of the S_c subcarriers between an arbitrary TX-RX antenna pair $t-r$ for N consecutive CSI samples. This matrix is given by the following equation.

$$\mathbf{H}_{t,r} = [\mathsf{H}_{t,r}(1)|\mathsf{H}_{t,r}(2)|\mathsf{H}_{t,r}(3)|...|\mathsf{H}_{t,r}(N)]^T \quad (2)$$

The columns of the matrix $\mathbf{H}_{t,r}$ represent the CSI time series for each OFDM subcarrier. To detect the starting and ending points of any arbitrary key, WiKey first normalizes the $\mathbf{H}_{t,r}$ matrix such that every CSI stream has zero mean and unit variance. We denote the normalized version of $\mathbf{H}_{t,r}$ by $\mathbf{Z}_{t,r}$. WiKey then performs the PCA based dimensionality reduction and denoising (as described in Section **??**) on $\mathbf{Z}_{t,r}$ and the resultant waveforms are further processed to detect the starting and ending points of the keystrokes from this particular TX-RX antenna pair. WiKey repeats this process on the CSI time series for all antenna pairs and obtains values for starting and ending points for keys based on the CSI time series from each antenna pair one by one. Finally, WiKey combines the starting and ending points obtained from all TX-RX antenna pairs to calculate a robust estimate of starting and ending points of the time windows containing those keystrokes. Next we explain these steps in more detail.

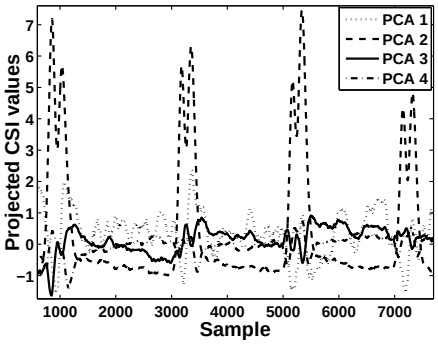
5.1 PCA on Normalized Stream

Let $\Phi_Z^{\{1:p\}}$ be an $S_c \times p$ dimensional matrix that contains the top p principal components obtained from PCA on $\mathbf{Z}_{t,r}$. We remove the first component from those top p principal components based on our observation that the first component captures majority of the noise, while subsequent components contain information about movements of hands and fingers while typing. This happens because PCA ranks principal components in descending order of their variance, due to which the noisy components with higher variance gets ranked among top principal components. Due to correlated nature of variations in multiple CSI time series, the removal of this PCA component does not lead to any significant information loss as remaining PCA components still contain enough information required for successfully detecting starting and ending points of the keystrokes.

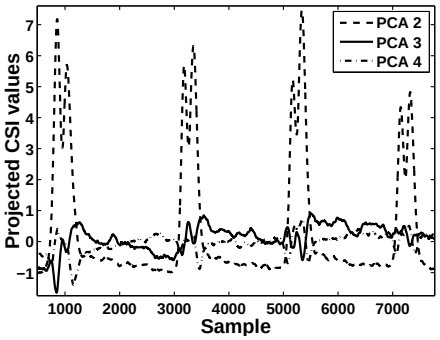
If we exclude the first component, the projection of the CSI stream $\mathbf{Z}_{t,r}$ of t-r transmit-receive antenna pair onto the remaining principal components $\Phi_Z^{\{2:p\}}$ can then be written as:

$$\mathbf{Z}_{t,r}^{\{2:p\}} = \mathbf{Z}_{t,r} \times \Phi_Z^{\{2:p\}} \quad (3)$$

where $\mathbf{Z}_{t,r}^{\{2:p\}}$ is an $N \times (p-1)$ dimensional matrix containing the projected CSI streams in its columns. We choose the $p = 4$ in our implementation based on our observation that only top 4 principal components contained most significant variations in CSI values caused by different keystrokes. Figure **??** shows the result of projecting normalized CSI time series $\mathbf{Z}_{t,r}$ onto its top 4 principal components. We observe from Figure **??** that by removing the first principle component, we essentially remove the most noisy projection among the all 4 projections of $\mathbf{Z}_{t,r}$.



(a) Top 4 projections



(b) Projections 2, 3 & 4

Figure 4: PCA of Z-normalized CSI stream $\mathbf{Z}_{t,r}$

5.2 Keystroke Detection

Although existing DFAR schemes propose techniques to automatically detect the start and end of activities, they can not be directly adapted for use in detecting the start and end of keystrokes. Existing schemes use simple threshold based algorithms for detecting the start and end of activities. While, threshold based schemes work well for macro-movements, they are not well suited for micro-movements such as those of hands and fingers while typing, where we need to precisely segment time series of keystrokes that are closely spaced in time. Unlike general purpose threshold based algorithms, we propose a keystroke detection algorithm that provides better detection accuracy, since it is strictly based on the experimentally observed shapes of different keystroke waveforms. The intuition behind our algorithm is that the CSI time series of every keystroke shows a typical increasing and decreasing trend in rates of change in CSI time series, similar to the one shown in Figure **??**. To detect such increase and decrease in rates of change in CSI time series, our algorithm uses a moving window approach to detect the increasing and decreasing trends in rates of change in all $p-1$ time series for each transmit-receive antenna pair *i.e.*, on each column of $\mathbf{Z}_{t,r}^{2:p}$. Our algorithm detects the starting and ending points of keystrokes in following six steps.

First, the algorithm calculates the mean absolute deviation (MAD) for each of the $p-1$ time series for each window of size W at j-th iteration. This is done primarily to detect the extent of variations in the values of a given time series. The main reason behind choosing MAD instead of variance is that in calculating, the deviations from the mean are squared which gives more weight to extreme values. In cases where a time series contains outliers, this results in undue weight given to those outlying values and that sig-

nificantly corrupts the measure of deviation. The MAD is calculated using following equation.

$$\triangle m_j[k] = \frac{\sum_{i=j}^{j+W} |\mathbf{Z}_{t,r}^{\{k\}}(i) - \overline{\mathbf{Z}}_{t,r}^{\{k\}}(j : j+W)|}{W} \quad (4)$$

where $\overline{\mathbf{Z}}_{t,r}^{\{k\}}(j : j+W)$ represents the vector of means of the kth projected CSI stream in j-th window. It calculates the value of $\triangle m_j$ for each sample point j and for the principle components $2 \leq k \leq p$.

Second, the algorithm adds the mean absolute deviations in each waveform to calculate a combined measure $\triangle M_j$ of MAD in all $p-1$ waveforms, which is calculated in the following equation.

$$\triangle M_j = \sum_{k=2}^{p} \triangle m_j[k] \quad (5)$$

Third, the algorithm compares $\triangle M_j$ to a heuristically set threshold $Thresh$. Let $\delta_j = \triangle M_j - Thresh$, then $\delta_j > 0$ shows that the current window j contains significant variations in CSI amplitudes.

Fourth, the algorithm compares δ_j to its value in last window δ_{j-1} to detect increasing or decreasing trend in detected variations. When $\delta_j - \delta_{j-1} > 0$, there is an increasing trend in the rate of change in combined MAD ($\triangle M_j$) of CSI time series and vice versa. These increasing and decreasing trends are captured in variables i_u and d_u, respectively. The algorithm increments the value of i_u by 1 whenever $\delta_j - \delta_{j-1} > 0$ and d_u by 1 whenever $\delta_j - \delta_{j-1} < 0$. Let σ represent *forgetting factor*, which is used to "forget" the variations caused by noise to avoid false positives. To forget such variations, the algorithm decrements both i_u and d_u by 1 if $\triangle M_j < Thresh$ for a duration of σW.

Fifth, as soon as the values of i_u and d_u exceed empirically determined thresholds I_u and D_u, respectively, the algorithm detects the start of the keystroke. As soon as the algorithm detects a keystroke, it estimates the starting point s_m and ending point e_m of the keystroke waveforms using following equations.

$$s_m = j - \beta W - B_{left} \quad (6)$$

$$e_m = j - \beta W + \mathfrak{t}_{avg} + B_{right} \quad (7)$$

where $\mathfrak{t}_{avg}$ is the average number of data points spanned by waveforms of different keystrokes, β is the *span factor* which determines the estimated starting point of the keystroke and B_{left} and B_{right} are *guard* intervals on both sides of the estimated keystroke interval. The guard intervals ensure that the detected keystroke waveforms are complete.

Last, our algorithm calculates the sum of powers in all waveforms lying within those starting and ending points and then compares this combined power with a *sum power threshold* (P_{avg}) to confirm the presence of a complete keystroke within that interval. This ensures that the training models are built using only those waveforms which contain complete shapes of the keystrokes. Once keystroke detection is confirmed, the algorithm finally returns the starting point (s_m) of the detected keystroke and jumps $\triangle \mathfrak{t}_{avg}$ data points ahead of s_m to look for next keystroke, where $\triangle \mathfrak{t}_{avg}$ is the average number of data points between arrival of two consecutive keystrokes. From the CSI data set we collected from our volunteers, we observed that on average the waveforms of a keystroke spanned $\mathfrak{t}_{avg} \approx 650$ data points and average number of data points between arrival of two consecutive keystrokes was $\triangle \mathfrak{t}_{avg} \approx 1250$ data points at the CSI sampling rate of $F_s = 2500$ samples/s. We empirically determined appropriate values for the remaining constants including W, D_u, I_u, σ, β, B_{left}, B_{right}, $Thresh$ and P_{avg}.

5.3 Combining Results from Antenna Pairs

As mentioned earlier, we obtain the starting points of keystrokes independently from each TX-RX antenna pair. Let $\mathbf{S}_{t,r}$ represent the set containing the starting points of all keystrokes obtained from the keystroke detection algorithm applied on the antenna pair $t-r$. First, we obtain the set $\mathbf{S}_{t,r}$ for each $t-r$ pair. Second, we take the average of all the starting points that are within $\triangle \mathfrak{t}_{avg}$ of each other in all sets $S_{t,r}$ to obtain a robust estimate of starting points of keystrokes. Third, based on experimentally measured average span $\mathfrak{t}_{avg}$ of different keystrokes, we calculate the ending points of all keystrokes by simply adding $\mathfrak{t}_{avg}$ to the corresponding starting point.

5.4 Extracting Keystroke Waveforms

Once the algorithm calculates the set of starting and corresponding ending points for keystrokes, we use those points to extract the waveforms from CSI matrix $\mathbf{H}_{t,r}$. Let $\mathbf{K}_{m,t,r}$ represent the CSI waveform of m^{th} keystroke extracted from the antenna pair t-r. Let $\overline{s}_m$ represent the average of the starting points for the m^{th} keystroke from all antenna pairs. We can express $\mathbf{K}_{m,t,r}$ in terms of $\mathbf{H}_{t,r}$ follows.

$$\mathbf{K}_{m,t,r} = \mathbf{H}_{t,r}(\overline{s}_m : \overline{s}_m + \mathfrak{t}_{avg}) \quad (8)$$

After extracting the CSI waveforms $\mathbf{K}_{m,t,r}$ from all subcarriers of the t-r antenna pair, we apply PCA on those CSI waveforms to remove the noisy components and obtain the components that represent the variations caused by movements of hands and fingers.

Unlike principle components derived from normalized streams, it is difficult to decide which PCA component represents noise and should be removed from the top p principal components for the case of $\mathbf{K}_{m,t,r}$. The difficulty arises because $\mathbf{K}_{m,t,r}$ contains the set of waveforms for a specific keystroke instead of the whole CSI stream, due to which the variance of noisy component often becomes small. We observe that the noisy PCA component keeps changing positions between 1^{st} and 2^{nd} place among the sorted PCA components for different extracted keystroke waveforms. In order to get rid of this problem, we first project $\mathbf{K}_{m,t,r}$ onto all top q principal components. Let $\Phi_K^{\{1:q\}}$ be an $S_c \times q$ dimensional matrix that represent the top q principal components in $\mathbf{K}_{m,t,r}$ obtained after applying PCA and $\mathbf{K}_{m,t,r}^{\{1:q\}}$ be an $L \times q$ dimensional matrix containing the projected CSI streams in its columns, where L is the length of segmented keystroke waveform. Thus, $\mathbf{K}_{m,t,r}^{\{1:q\}}$ is given by the following equation.

$$\mathbf{K}_{m,t,r}^{\{1:q\}} = \mathbf{K}_{m,t,r} \times \Phi_K^{\{1:q\}} \quad (9)$$

In our implementation, we choose $q = 4$. This choice is again based on the observation that the top 4 principal components contain enough information about keystrokes required to achieve high accuracy during classification.

To detect which waveform in $\mathbf{K}_{m,t,r}^{\{1:q\}}$ represents the noisy projection, we chose the top 2 projected waveforms and divide each of them into R bins and calculate the variances

in those bins. We then compare the variances calculated for different bins of one waveform with the corresponding bins of the other waveform. The waveform that has larger number of higher variance bins is considered to be the noisy projection, which we remove from $\mathbf{K}_{m,t,r}^{\{1:q\}}$ to finally get $q-1$ waveforms. Here we leverage the fact that although overall variance of a noisy projection may be smaller than the variance of other waveforms, but if the waveform is divided into appropriate number of smaller bins then the number of bins in which the variance of the noisy projection is higher than the corresponding bins of other waveforms is always larger. This is because the impact of noise is more dominant in smaller time windows compared to larger time windows. We used R=10 in our implementation of WiKey.

PCA can lead to different ordering of principal components in waveforms of different keystrokes of the same key, because the ordering of waveforms done by PCA is based solely on the value of their variance, which can change even if a key is pressed in a slightly different way. This is problematic because to recognize the keystrokes, we need to compare the projections of an unseen key with the corresponding projections of the keys in the training data. In order to minimize the possibility of reordering, we order the projected keystroke waveforms in descending order of their peak to peak values before using the waveforms for feature extraction and classifier training.

6. FEATURE EXTRACTION

To differentiate between keystrokes, we need to extract features that can uniquely represent those keystrokes. As different keys on a keyboard are closely placed, standard features such as maximum peak power, mean amplitude, root mean square deviation of signal amplitude, second/third central moments, rate of change, signal energy or entropy, and number of zero crossings cannot be used because adjacent keys give almost the same values for these features. Tables **??** and **??** show means and variances of some of these features calculated for 2^{nd} waveform in the extracted CSI-waveforms for keystrokes of alphabetic keys pressed by a users. It can be observed that the values of these features for different keys (for example 'c' and 'd') come out to be very similar. Looking at the means calculated for features like energy and number of zero crossings in Table **??**, it seems that they have different values for different keys. But as we observe from Table **??**, the variance of those features is high. Due to the reasons above, it becomes infeasible to use these features for keystroke classification. Frequency analysis is also not feasible because the frequency components in keystrokes of many different keys are similar. Another reason behind inapplicability of frequency domain analysis is that they lead to complete loss of time domain information.

From our data set, we have observed that although the frequency components in most keys are similar, they occur at different time instants for different keys. Therefore, we use shapes of the extracted keystroke waveforms as their features because the shapes retain both time and frequency domain information of the waveforms and are thus more suited for use in classification. We observed experimentally that the shapes of different keystroke waveforms were quite different from each other, as shown by Figure **??** and **??**.

Directly using the extracted keystroke waveforms as keystroke features leads to high computational costs in the classification process because waveforms contain hundreds of data points per keystroke. Therefore, we apply Discrete Wavelet Transform (DWT) to compress the extracted keystroke waveforms while preserving most of the time and frequency domain information.

The DWT of a discrete signal $y[n]$ can be written in terms of wavelet basis functions as:

$$y[n] = \frac{1}{\sqrt{L}} \sum_{k} \lambda(j_0, k)\varphi_{j_0,k}(n) + \frac{1}{\sqrt{L}} \sum_{j=j_0}^{\infty} \sum_{k} \gamma(j,k)\psi_{j,k}(n)$$

where L represents the length of signal $y[n]$. The functions $\varphi_{j,k}(n)$ are called *scaling functions*, where as the corresponding coefficients $\lambda(j,k)$ are known as *scaling* or *approximation coefficients*. Similarly, the functions $\psi_{j,k}(n)$ are known as *wavelet functions* and the corresponding coefficients $\gamma(j,k)$ are known as *wavelet* or *detail coefficients*. To calculate approximation and detail coefficients, the scaling and wavelet functions are chosen such that they are orthonormal to each other. Thus, the following condition holds.

$$\langle \psi_{j,k}(n), \varphi_{j_0,m}(n) \rangle = \delta_{j,j_0}\delta_{k,m}$$

Based on the condition above, the approximation and detail coefficients calculated for j-th scale can then be written as:

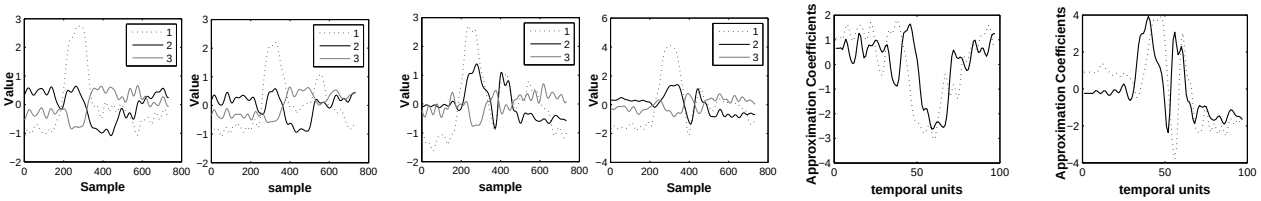
$$\lambda(j,k) = \langle y[n], \varphi_{j+1,k}(n) \rangle = \frac{1}{\sqrt{L}} \sum_{n} y[n]\varphi_{j+1,k}(n)$$

$$\gamma(j,k) = \langle y[n], \psi_{j+1,k}(n) \rangle = \frac{1}{\sqrt{L}} \sum_{n} y[n]\psi_{j+1,k}(n)$$

To achieve desired compression using DWT, we need to select appropriate wavelet and scaling filters. We tested the accuracy of our classifier using two different wavelet filters: Daubechies and Symlets. We choose Daubechies D4 (four coefficients per filter) wavelet and scaling filters because the models trained with the DWT features extracted using these filters achieved higher classification accuracy. For each keystroke, we perform DWT 3 times on each one of its $(q-1) = 3$ waveforms, which is achieved by applying DWT on the approximation coefficients obtained from the previous steps. We choose to apply DWT 3 times because this preserves enough details of those waveforms required for successful classification while achieving maximum compression. WiKey uses only the approximation coefficients as keystroke features and discards the detail coefficients because approximation coefficients alone result in good classification accuracy. Therefore, we have $3 \times M_T \times M_R$ keystroke shapes for every keystroke, *i.e.* the approximation coefficients of all 3 waveforms extracted from the CSI time series in each TX-RX antenna pair, where M_T is the number of transmitting antennas and M_R is the number of receiving antennas. Figures **??** through **??** show feature extraction procedure performed on the 2^{nd} keystroke waveforms for keys 'i' and 'o', extracted from TX-1, RX-1 antenna pair.

7. CLASSIFICATION

After obtaining DWT based shape features of keystrokes, WiKey builds training models for classification using them. As WiKey needs to compare shape features of different keystrokes, we need a comparison metric that provides an effective measure of similarity between shape features of two keystrokes. WiKey uses a well-known method called *dynamic time warping* (DTW) that calculates the distance between waveforms by performing optimal alignment between them.



(a) Keystroke waveforms for key **i** (b) Keystroke waveforms for key **o** (c) DWT features of **i** from 2^{nd} waveforms (d) DWT features of **o** from 2^{nd} waveforms

Figure 5: Feature extraction from 2^{nd} keystroke waveforms extracted from TX-1, RX-1 for I and O

Table 1: Average values of features extracted from keystrokes of keys a-z collected from user 10

Features	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Mean amplitude	-0	-0.04	0.0124	-0.03	0.045	-0.043	-0.076	-0.06	0.014	-0.03	0.03	-0.01	-0	0.032	0.02	0.03	-0.012	0.008	0.054	7E-04	-0.013	-0.02	-0	-0.1	-0.02	0.06
Second central moment	0.08	0.133	0.0801	0.083	0.156	0.1818	0.6523	0.263	0.12	0.231	0.33	0.11	0.1	0.108	0.09	0.19	0.1022	0.051	0.245	0.192	0.062	0.12	0.097	0.26	0.09	0.21
Third central moment	0.02	-0.03	0.0036	-0.01	0.029	-0.06	-0.919	-0.05	-0.01	-0.1	0.05	0.02	-0	0.01	0.01	0.04	-0.006	0.003	0.098	-0.101	-0.01	0.029	0.023	-0	-0.02	0.04
RMS deviation	0.27	0.359	0.2782	0.285	0.385	0.4244	0.7899	0.506	0.332	0.472	0.57	0.32	0.3	0.323	0.29	0.43	0.3137	0.222	0.472	0.434	0.242	0.335	0.306	0.5	0.3	0.45
Energy	71.5	116.6	69.788	73.34	137.5	159.43	570.8	232.1	104.8	201.4	288	95.2	83.7	94.98	75.6	167	88.928	44.22	215.5	167.1	54.56	104.4	84.48	227	81.5	182
Entropy	9.76	9.762	9.7616	9.762	9.762	9.7616	9.7616	9.762	9.762	9.762	9.76	9.76	9.76	9.762	9.76	9.76	9.7616	9.762	9.762	9.762	9.762	9.762	9.762	9.76	9.76	9.76
Zero Crossings	11.8	6.913	12.363	6.225	6.4	4.375	4.075	3.4	12.08	9.088	6.05	13.7	10	9.063	13.8	12.9	11.85	15.41	6.35	12.85	16.75	11.88	14.3	6.48	10.1	7.55

Using DTW distance as the comparison metric between keystroke shape features, WiKey trains an ensemble of *k-nearest neighbour* (kNN) classifiers using those features from all TX-RX antenna pairs. WiKey obtains decisions from each classifier in the ensemble and uses majority voting to obtain final result. Next, we first explain how we apply DTW on the keystroke shape features and then explain how we train the ensemble of classifiers.

7.1 Dynamic Time Warping

DTW is a dynamic programming based solution for obtaining minimum distance alignment between any two waveforms. DTW can handle waveforms of different lengths and allows a non-linear mapping of one waveform to another by minimizing the distance between the two. In contrast to Euclidean distance, DTW gives us intuitive distance between two waveforms by determining minimum distance warping path between them even if they are distorted or shifted versions of each other. *DTW distance* is the Euclidean distance of the optimal warping path between two waveforms calculated under boundary conditions and local path constraints [**?**]. In our experiments, DTW distance proves to be very effective metric for comparing two shape features of different keystrokes. WiKey uses the open source implementation of DTW in the Machine Learning Toolbox (MLT) by Jang [**?**]. WiKey uses local path constraints of 27, 45, and 63 degrees while determining minimum cost warping path between two waveforms. For the features extracted for keys 'i' and 'o' shown in figures **??** and **??**, the DTW distance among features of key 'i' was 18.79 and the DTW distance among features of key 'o' was 19.44. However, the average DTW distance between features of these keys was 44.2.

7.2 Classifier Training

To maximize the advantage of having multiple shape features per keystroke obtained from multiple transmit-receive antenna pairs, we build separate classifiers for each of those shape features. We build an ensemble of $3 \times M_T \times M_R$ classifiers using kNN classification scheme. WiKey requires the user to provide training data for the keystrokes to be recognized and each classifier is trained using the corresponding features extracted from CSI time series from all TX-RX antenna pairs. To classify a detected keystroke, WiKey feeds the shape features of that keystroke to their corresponding kNN classifiers and obtains a decision from each classifier in the ensemble. Each kNN classifier searches for the majority class label among k nearest neighbors of the corresponding shape feature using DTW distance metric. WiKey calculates the final result through majority voting on the decisions of all kNN classifiers in the ensemble.

8. IMPLEMENTATION & EVALUATION

8.1 Hardware Setup

We implemented our scheme using off-the-shelf hardware devices. Specifically, we use a Lenovo X200 laptop with Intel Link 5300 WiFi NIC as the receiver that connects to the three antennas of the X200 laptop. The X200 laptop has 2.26GHz Intel Core 2 Duo processor with 4GB of RAM and Ubuntu 14.04 as its operating system. We used TP-Link TL-WR1043ND WiFi router as transmitter operating in 802.11n AP mode at 2.4GHz. We collect the CSI values from the Intel 5300 NIC using a modified driver developed by Halperin *et al.* [**?**]. The transmitter has 2 antennas and the receiver has 3 antennas, *i.e.*, $M_T = 2$ and $M_R = 3$. This gives $3 \times 2 \times 3 = 18$ classification models for each key in our evaluations.

We place the X200 laptop at a distance of 30 cm from the keyboard such that the back side of its screen faces the keyboard on which the users type and its screen is within the line-of-sight (LOS) of the WiFi router it is connected to. The distance of WiFi router from the target keyboard is 4 meters. The CSI values are measured on ICMP *ping* packets sent from the WiFi router, *i.e.*, the TP-Link TL-WR1043ND, to the laptop at high data rate of about 2500 packets/s. Setting a higher *ping* frequency leads to higher sampling rate of CSI, which ensures that the time resolution of the CSI values is high enough for capturing maximum details of different type of keystrokes.

8.2 Data Collection

To evaluate the accuracy of WiKey, we collected training and testing dataset from 10 users. These 10 users were general university students who volunteered for the experiments and only 2 out of them had some know how of wire-

Table 2: Variance of different features extracted from keystrokes of keys a-z collected from user 10

Features	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Mean amplitude	0.00029	4E-04	0.0003	1E-04	4E-04	0.0002	0.0003	8E-04	5E-04	5E-04	5E-04	2E-04	5E-04	3E-04	3E-04	5E-04	0.0003	0.00018	6E-04	4E-04	3E-04	1E-04	3E-04	4E-04	6E-04	4E-04
Second central moment	0.00513	0.003	0.0011	0.001	0.007	0.0028	0.1008	0.012	0.005	0.009	0.016	0.006	0.002	0.003	0.002	0.007	0.0017	0.00041	0.03	0.003	0.001	0.006	0.002	0.007	0.003	0.005
Third central moment	0.00155	9E-04	0.0001	2E-04	0.002	0.0033	0.7021	0.002	0.001	0.007	0.009	0.017	5E-04	3E-04	5E-04	0.003	0.0003	7.70E-05	0.024	0.003	1E-04	0.015	9E-04	0.001	6E-04	0.003
RMS deviation	0.0108	0.006	0.0031	0.004	0.011	0.0038	0.0348	0.011	0.011	0.01	0.012	0.009	0.006	0.005	0.004	0.008	0.0042	0.00196	0.026	0.004	0.004	0.008	0.004	0.007	0.007	0.007
Energy	3874.59	2283	816.91	912.4	5204	2160	76863	9315	3925	6846	12094	4883	1679	2153	1150	5048	1296.2	308.95	23201	2181	886.9	4714	1403	5166	2100	4147
Entropy	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Zero Crossings	26.3859	12.36	33.196	12.94	9.433	6.2627	3.9943	3.585	44.91	13.14	12.58	31.14	28.51	15.25	29.24	24.09	21.673	17.3847	12.21	17.7	36.85	21.63	27.71	13.67	31.49	6.529

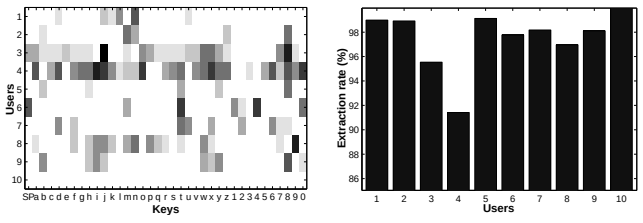
less communication. Users 1–9 first provided 30 samples for each of the 37 keys (26 alphabets, 10 digits and 1 space bar) by pressing that key multiple times. After this, these users typed the sentence S_1 = "the quick brown fox jumped over the lazy dog" two times, without spaces.

To evaluate how the number of training samples impact the accuracy, we collected 80 samples for each of the 37 keys from User 10. Afterwards, this user typed each of the following sentences 5 times, without spaces: S_1 ="the quick brown fox jumps over the lazy dog", S_2 = "nobody knew why the candles blew out", S_3 = "the autumn leaves look like golden snow", S_4 = "nothing is as profound as the imagination" and S_5 = "my small pet mouse escaped from his cage". We asked users to type naturally with multiple fingers but only press one key at a time while keeping the average keystroke inter-arrival time at 1 second. After recording the CSI time series for each of the above experiments, we first applied our keystroke extraction algorithm on those recorded CSI time series to extract the CSI waveforms for individual keys and then extracted the DWT based shape features from each of the extracted keystroke waveforms.

8.3 Keystroke Extraction Accuracy

We evaluate the accuracy of our keystroke extraction algorithm in terms of the detection ratio, which is defined as the total number of correctly detected keystrokes in a CSI time series divided by the total number of actual keystrokes.

The detection ratio of our proposed algorithm is more than 97.5%. Figure **??** shows the color map showing the percentage of the missed keystrokes of all 37 keys for all 10 users.



(a) Colormap for missed keys (b) Keystroke extraction rates per user averaged over all keys

Figure 6: Keystroke extraction results

The darker areas represent higher rate of missed keystrokes. We can observe from this figure that the number of missed keystrokes vary for different individuals depending upon their typing behaviors. For example we observed that the keystrokes of user 4 were missed in higher percentage with average detection ratio of 91.8% whereas the keystrokes of user 10 were not missed at all with average detection ratio of 100% calculated over all 37 keys. The lower extraction accuracy for user 4 shows that more keystrokes were missed, which is due to the significant difference in his typing behavior compared to other users. The accuracy of our scheme for such a user can be increased significantly by tuning the parameters of our algorithm for the given user. We also observe from this figure that the keystrokes that are missed are usually those for which fingers move very little when typing. For example, in pressing keys 'a', 'd', 'f', 'i', 'j' and 'x' the hands and fingers move very little, and thus the variations in the CSI values sometimes go undetected. Figure **??** shows the keystroke extraction rate for each user averaged over all 37 keys. The experimental results show that our keystroke extraction algorithm is robust because it consistently achieves high performance over different users without requiring any user specific tuning of system parameters.

8.4 Classification Accuracy

We evaluate the classification accuracy of WiKey through two sets of experiments. In the first set of experiments, we build classifiers for each of the 10 users using 30 samples and measure the 10-fold cross validation accuracy of those classifiers. In the second set of experiments, we build classifier for user 10 while increasing the number of samples from 30 to 80 in order to observe the impact of increase in the number of training samples on the classification accuracy. Cross validation automatically picks a part of data for training and remaining for testing and does not use any data in testing that was used in training. Recall that the WiKey uses kNN classifiers for recognizing keys. In all of following experiments, we set $k = 15$.

8.4.1 Accuracy with 30 Samples per Key

We evaluate the classification accuracy of WiKey in terms of average accuracy per key and average accuracy on all keys of any given user. We also present confusion matrices resulting from our experiments. A confusion matrix tells us which key was recognized by WiKey as which key with what percentage. We calculate the average accuracy per key by taking the average of confusion matrices obtained from all users and average accuracy on all keys of any given user by averaging the accuracy on all keys within the confusion matrix of that user. For each user, we trained each classifier using features from 30 samples of each key. We conducted our experiments on all 37 keys as well as on only 26 alphabet keys and performed 10-fold cross validation to obtain the confusion matrices.

WiKey achieves an overall keystroke recognition accuracy of 82.87% in case of 37 keys and 83.46% in case of 26 alphabetic keys when averaged over all keys and users. Figure **??** shows the recognition accuracy for each key across all users for the 26 alphabetic keys. Similarly, Figure **??** shows the recognition accuracy for each key across all users for all 37 keys. Figure **??** shows the average recognition accuracy achieved by each user for both 26 keys and 37 keys. We observe that the recognition accuracy for 26 alphabetic keys is on average greater than the recognition accuracy for the all

37 keys. This is because the keystroke waveforms of the digit keys (0-9) often show similarity with keystroke waveforms of alphabet keys in the keyboard row staring with QWE, which leads to slightly greater number of misclassifications.

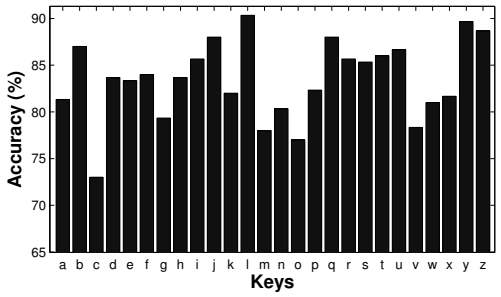


Figure 7: Mean accuracy for keys A-Z (Users 1-10)

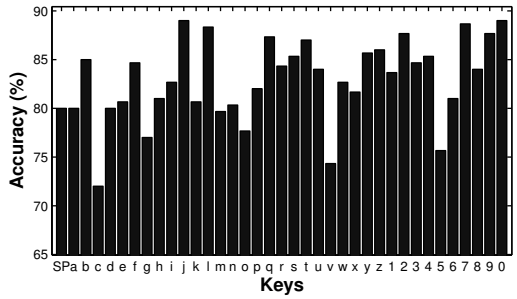


Figure 8: Mean accuracy for all 37 keys (Users 1-10)

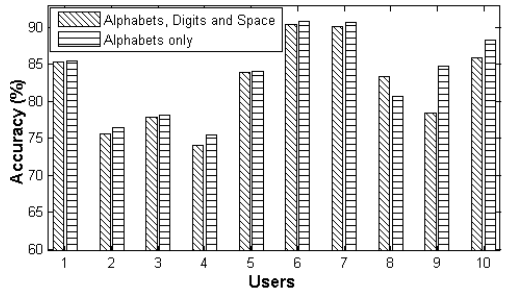


Figure 9: Per user average classifier accuracies

8.4.2 *Accuracy vs. the Size for Training Set*

To determine the impact of the number for training samples on the accuracy of WiKey, we again perform two sets of experiments: one for 26 alphabetic keys and other for all 37 keys.

The accuracy of WiKey increases when the number of training samples per key are increased from 30 to 80. Figure **??** shows the results from 10-fold cross validation for the 26 alphabetic keys when 80 training samples are used per key. We observe from this figure that the recognition accuracy increased from 88.3% (as seen in Figure **??**) to 96.4% when the number of training samples are increased from 30 to 80. Figure **??** shows the results from 10-fold cross validation for all 37 keys when 80 training samples are used per key. We again observe that the recognition accuracy increased from 85.95% (as seen in Figure **??**) to 89.7% when the number of training samples are increased from 30 to 80. The gray-scale maps of the confusion matrix obtained after 10-fold cross-validation on 80 training samples of User 10 is shown in Figure **??**.

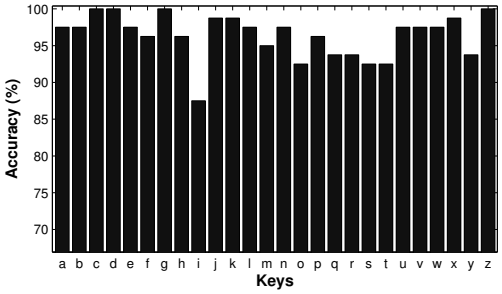


Figure 10: Accuracy for keys A-Z from user 10

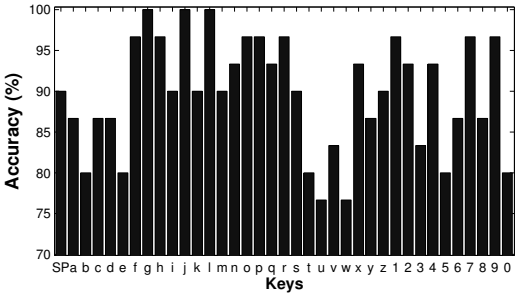


Figure 11: Accuracy for all 37 keys from user 10

8.4.3 *Effects of CSI Sampling Rate and Number of Training Samples*

In previous experiments, we used high CSI sampling rate of 2500 samples/s. Furthermore, the 10-fold cross validation automatically chose 10% of the data for testing and remaining 90% for training. Next, we evaluate the effect of changing the CSI sampling rate and the percentage of data used for training on accuracy. To extract keystrokes, we halved the values used for W, D_u, I_u, B_{left}, and B_{right}. We performed $X-$fold cross validation ($2 \leq X \leq 10$) on the data obtained for alphabetic keys from user 10. Figure **??** plots the accuracies for number of folds varying from 2 to 10, where each plotted value if the average over all alphabet keys. We observe from Figure **??** that the accuracies dropped compared to previously achieved accuracy because of the drop in resolution of keystroke shapes due to reduced sampling rate. We also observe that recognition accuracies of the keys for which hands and fingers move little were affected the most. When 50% of data was used for training, *i.e.*, for 2-fold cross validation, the accuracies for keys 'j','x','v' and 'p' dropped below 60%. However, the average accuracy remained approximately 80% for all folds.

8.5 Real-world Evaluation on Sentences

To evaluate WiKey in real world scenarios, we collected CSI data for different sentences typed by users 1-10 as mentioned earlier in Section **??**. For recognition of keystrokes in sentences, we performed training using the dataset of individual keystrokes and the keystrokes extracted from datasets obtained from typing the sentences were used as test data.

WiKey achieves an average keystroke recognition accuracy of 77.43% for typed sentences when 30 training samples per key were used. For each user, we trained classifiers using 30 samples for each of the 26 alphabetic keys. We then applied our keystroke extraction algorithm to first extract waveforms of individual keys, applied PCA on them to denoise the waveforms and then extracted the shape features for each extracted key and feeded them to the classifiers to

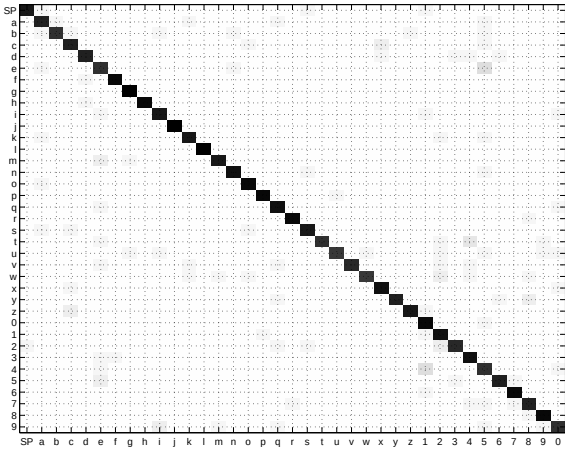


Figure 12: Color map of user 10's confusion matrix

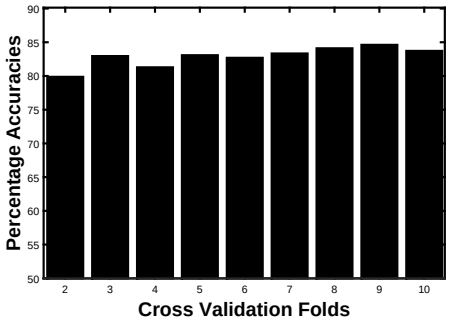


Figure 13: Multifold cross-validated average accuracies for user 10's lower resolution keystrokes

recognize the keystrokes in the sentence. Figure **??** shows the keystroke recognition accuracy for the sentences typed by each user.

WiKey achieves an average keystroke recognition accuracy is 93.47% in continuously typed sentences with 80 training samples per key. We first trained classifiers using 80 samples for each of the 26 alphabetic keys and then fed them with keystrokes from typed sentences. Figure **??** shows the keystroke recognition accuracy for all the sentences (S_1 to S_5) typed by user 10. The average keystroke recognition accuracy rate for user 10 in previous experiment, which used 30 samples for training classifiers was just 80%. Thus, we can conclude that increasing the number of training samples increases the accuracy of WiKey.

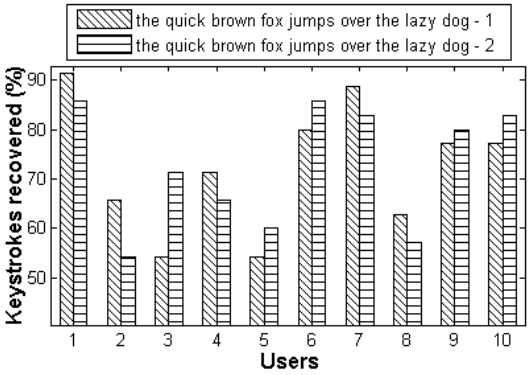


Figure 14: Keystroke recognition for sentences collected from all users using 30 samples per key

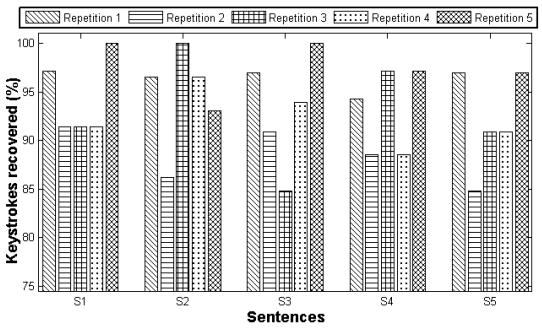


Figure 15: Keystroke recognition for sentences collected from user 10 using 80 samples per key

9. LIMITATIONS

In this section, we discuss the key limitations of the current implementation of WiKey.

Currently WiKey works well only under relatively stable and controlled environments. The accuracy of our current scheme is affected by variations in the environment such as human motion in surrounding areas, changes in orientation and distance of transceivers, typing speeds, and keyboard layout and size. Next, we elaborate some limitations of our current implementation of WiKey.

Interference Free Surroundings. During our experiments, we assumed that the major motion is due to keystrokes of the target user only and no other major motion such as walking occurs in the room where CSI data is collected. WiKey is currently designed for and tested with only two persons in a room i.e. the person guiding the user during data collection and the user himself. WiKey may be extended to allow small movement in the environment e.g. having multiple persons walking in a library, however this would require training WiKey with the profiles of those activities and adding the capability to subtract the waveforms of those activities to extract the waveforms of keystrokes. Furthermore, most of the parameters used in our keystroke extraction algorithm are scenario dependent and need to be changed if CSI sampling rates or physical environment changes such as change in distance or orientation of transceivers. In future, we plan to develop schemes for automatic tuning of system parameters.

Devices Positioning. We tested the accuracy of WiKey using the same keyboard for all users while keeping the transceivers at the same distance and in same direction with respect to the keyboard. Furthermore, we took all samples from any given user on the same day because it took around 2 hours to get complete data from each user. It was difficult for the users to spare time to provide data for different orientations and distances and on multiple days. Therefore, we put the laptop at a fixed distance and direction from the user and used maximum sampling rate to get maximum information from CSI time series. Figures **??** and **??** show the shapes of the waveforms for key '5' when distance and orientation of receiver is changed with respect to the target keyboard, while WiFi AP is at 2 meters distance from the keyboard. We increased the distance linearly, while remaining in LOS of the transmitter. We observe from these figures that not only the shapes of the waveforms change but the measured variations (e.g. peak to peak values) due to keystrokes also attenuate. Similarly, when changing ori-

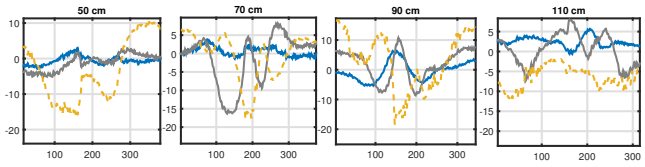


Figure 16: Change in shape with RX distance

entation, we observed that as we orient away from LOS of transmitter, the variations attenuate progressively. Similar trends were observed when AP was placed at different distances from keyboard, while keeping the receiver 30cm from keyboard, as shown in figure **??**. These observations are consistent with the results of previous studies [**?**] [**?**] that also show that CSI values tend to be different at different locations in an area. Figure **??** shows how CSI waveforms of the key look on different days. The dissimilarities in CSI waveforms can be attributed to inconsistencies in user's typing behavior on different days. As the shapes of waveforms tend to differ on different days or when distance and orientation of transceivers change, in its current form, WiKey needs to be trained in each given scenario.

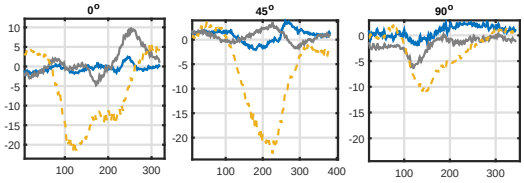


Figure 17: Change in shape of with angle

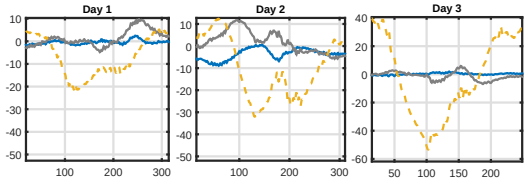


Figure 18: Change in shape with days

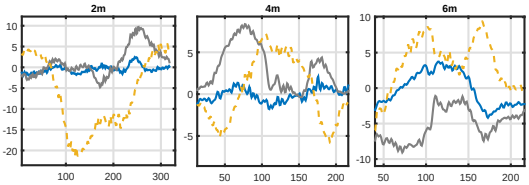


Figure 19: Change in shape with AP distance

Controlled Typing. During data collection, we instructed the users not to move their heads or other body parts significantly while typing. However, we allowed natural motions which occur commonly when a person types, such as eye winking and movements in the arm, shoulder and fingers on the side of the hand being used for typing. We also instructed the users to type one key at a time while keeping the inter arrival time of keystrokes between 0.5 to 1 second to facilitate correct identification of start and end times of keystrokes. However, we did allow users to use multiple fingers for typing so that they use whichever finger they naturally use to press any given key.

CSI Sampling Rate. The sampling rate of CSI is directly related to allowable inter-arrival times of keystrokes *i.e.*, how fast a user can be allowed to type. Both keystroke extraction and keystroke recognition accuracies depend on the time resolution of CSI values. The time resolution is high, there are more samples between two keystrokes, which increases the keystroke extraction accuracy. Similarly, when the time resolution is high, there are more samples for each keystroke; consequently, there is more information in the CSI waveform, which increases the keystroke recognition accuracy. We used highest possible sampling rate of 2500 samples/second, which translates to 2500 packets/second. Although real-world WiFi APs may not transmit at such high rates, an attacker can use his own AP to transmit ping packets at such a high rate to launch the attack.

Training Requirements. When collecting data from users, we did not bind the users to be consistent in their typing behavior. Consequently, waveforms for different keystrokes for a specific key were sometimes inconsistent even though we collected data from the users in the same sitting. To get high recognition accuracies with such a data, WiKey requires many samples per key from each user which may be difficult to obtain in real life attack scenarios. Still, there exist ways through which an attacker can obtain the training data. For example, an attacker can start an online chat session with a person sitting near him and record CSI values while chatting with him.

User Specific Training. In our current implementation of WiKey, we train the classifiers using one user and test the classifier using the test samples from the same user. However, we hypothesize that if we train our classifier using a large number of users, the trained classifier will be able to capture commonalities between users and will then be able to recognize the keystrokes of any unknown user. At the same time, we also acknowledge that it is extremely challenging to build such a universal classifier that works for almost every user because WiFi signals are susceptible to various factors such as finger length/width, typing styles, and environmental noise.

10. CONCLUSION

In this paper, we make the following key contributions. First, we propose the first WiFi based keystroke recognition approach, which exploits the variations in CSI values caused by the micro-movements of hands and fingers in typing. The key intuition is that while typing a certain key, the hands and fingers of a user move in a unique formation and direction and thus generate a unique pattern in the time-series of CSI values for that key. Second, we propose a keystroke extraction algorithm that automatically detects and segments the recorded CSI time series to extract the waveforms for individual keystrokes. Third, we implemented and evaluated the WiKey system using a TP-Link TL-WR1043ND WiFi router and a Lenovo X200 laptop. Our experimental results show that WiKey achieves more than 97.5% detection rate for detecting the keystroke and 96.4% recognition accuracy for classifying single keys. In real-world experiments, WiKey can recognize keystrokes in a continuously typed sentence with an accuracy of 93.5%. The key scientific value of this work is in demonstrating the possibility of recognizing micro-gestures such as keystrokes using commodity WiFi devices. We have shown that our technique works in controlled environments, and in future we plan to address the problem of mitigating the effects of more harsh wireless environments by building on our micro-gesture extraction and recognition techniques proposed in this paper.