SIA: Smartwatch-Enabled Inference Attacks on Physical Keyboards Using Acoustic Signals

Ülkü Meteriz-Yıldıran
University of Central Florida
Orlando, USA
meteriz@knights.ucf.edu

Necip Fazıl Yıldıran
University of Central Florida
Orlando, USA
yildiran@knights.ucf.edu

David Mohaisen
University of Central Florida
Orlando, USA
mohaisen@ucf.edu

ABSTRACT

The convergence of various technologies, such as smartwatches, smartphones, etc. has proven to be beneficial, although poses various security and privacy risks. In this paper, we explore one such risk where a smartwatch can be exploited to infer what a user is typing on a physical keyboard while wearing the smartwatch. We exploited the acoustic emanations of the keyboard as recorded by the smartwatch to perform the proposed attack—SIA. To address various environment-related challenges, SIA employs four stages: *Noise Cancelling*, *Keystroke Detection*, *Key Identification* and *Word Correction*, where several digital signal processing, machine learning, and natural language processing techniques are utilized to produce the final inference. Our results show that an acoustic emanation of a physical keyboard captured by a smartwatch recovers up to 98% of the typed text. We also showed that utilizing the noise cancellation, SIA is robust to the changes in the attack environment, which further boosts the practicality of the attack. The findings are alarming and call for further investigation on methods to cope with inference attacks due to the convergence of those technologies.

CCS CONCEPTS

• **Security and privacy**; • **Human-centered computing** → **Ubiquitous and mobile devices**; • **Computing methodologies** → *Natural language processing*; *Machine learning*;

KEYWORDS

Smartwatch; Wearable Privacy; Keylogging; Signal Processing; Noise Reduction; Machine Learning; Natural Language Processing.

ACM Reference Format:
Ülkü Meteriz-Yıldıran, Necip Fazıl Yıldıran, and David Mohaisen. 2021. SIA: Smartwatch-Enabled Inference Attacks on Physical Keyboards Using Acoustic Signals. In *Proceedings of the 20th Workshop on Privacy in the Electronic Society (WPES '21), November 15, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3463676.3485607

1 INTRODUCTION

Smartwatches have skyrocketed in popularity for their mobility features, which brings a lot of convenience, allowing improving users' quality of life with the numerous features they support; e.g., health and fitness monitoring, phone calls, voice commands, activity recognition [19, 34], gesture recognition [8, 14, 36, 40], and authentication [20, 22, 24, 40]. According to a recent survey by Rock Health [12], the adoption of wearable devices increased from 24% in 2018 to 33% in 2019. Per another report [1], the year-over-year growth of the smartwatches market is expected to be 14.5% for 2020—2025. Moreover, the same report suggests that IoT-driven smartwatches that are capable of operating standalone as well as interacting with other devices are a key trend. Another study [35] that quantifies smartwatches usages suggests that smartwatches are nowadays used more frequently than smartphones.

To keep up with the ever-growing user expectations, mass-market smartwatches are equipped with different I/O mechanisms, e.g., motion sensors, touch screen, heart rate sensor, thermometer, microphone, speaker, etc. Such a range of input mechanisms brings about promising as well as controversial aspects of smartwatches.

Despite the rise of smartwatches, personal computers (PC), including laptops, are still the most essential electronic devices for many users. Per a survey by Statista (2017), 88% of respondents (U.S.) stated they used a PC/laptop, either professionally or personally. The COVID-19 outbreak pushed teleworking (i.e., "work from home") and further boosted personal computers usage.

Both PCs and smartwatches are used for collecting and storing sensitive data of users [2, 5, 27, 30], and encryption is a commonly used for protecting such data on those devices. However, I/O peripherals, such as keyboards, touch screens, and printers, which are used for the input and output of unencrypted data, are a constant target of attacks. For example, various recent studies have shown the privacy risk introduced by keyboard acoustic emanations and their use for inferring sensitive data. While those attacks are alarming, a number of them require a malicious microphone to be planted by the adversary near the victim's keyboard, limiting the practicality of those attacks. Although using the victim smartphone's microphones is ideal for capturing the acoustic emanations and facilitating such attacks, the assumption is quite strong and often unrealistic, entailing that the phone should be in the same exact position at all times or that the adversary has to be able to *position* or *control* the victim's smartphone.

Motivated by those intricacies, this paper explores the attack surface due to a victim's smartwatch, which addresses those shortcomings. Our contribution is SIA, an attack that is facilitated by keyboard acoustic emanations captured by a smartwatch microphone. The attack goal is to recover characters typed by a victim using the keyboard acoustic emanations captured by the user's smartwatch. Obtaining the physical signals from the smartwatch microphones alleviates (or even eliminates) the positioning problem

WPES '21, November 15, 2021, Virtual Event, Republic of Korea

ACM ISBN 978-1-4503-8527-5/21/11…$15.00
https://doi.org/10.1145/3463676.3485607

introduced in the previous studies since the smartwatch is typically near the physical keyboard; e.g., the user wears it while typing.

Contributions. In this paper, we make the following unique contributions. (1) We propose a side-channel attack, SIA, which is facilitated by the smartwatches of a victim. We use SIA to uncover keystrokes using the physical signals of the acoustic emanations. In designing SIA, We demonstrated a threat model that features two plausible attack scenarios. (2) In realizing SIA, we building a four-stage pipeline: *Noise Cancelling*, *Keystroke Detection*, *Key Identification* and *Word Correction*. The noise canceling stage allows us to neutralize the environmental changes in the captured data, while the audio features representative, using multiple customized learning techniques, allows us to identify the correct keys. On top of the baseline performance, which is already promising, we employ state-of-the-art natural language processing techniques to improve SIA's accuracy by word correction. (3) We present an extensive set of experiments through which we show that an adversary can recover up to 98% of the typed text by a victim. (4) We evaluate various keyboards with different recording devices and use configurations, and demonstrate the effect of the recording device mobility as well as keyboards physical properties on the performance of the attack. (5) We collected three types of test datasets to examine how our attack performs under different types of use patterns. All datasets will be made public upon the publication of this work.

Organization. The organization of the rest of this paper is as follows. In section 2 we review the related work. In section 3 we outline the system and threat models. In section 4 we review our attack methodology with the associated technical details. In section 5, we present our evaluation, including a small user study and a practical attack. In section 6 we enlist the limitations of our work and the future directions, followed by our concluding remarks in section 7

2 RELATED WORK

There has been a significant number of studies on side channels and their utilization for physical keyboards inference, which we review. We start by a brief background, followed by various techniques falling under a broad set of mediums utilized for their operation.

Earlier Work and Theme. Several studies are reported in the literature on the topic of keyboard inference using side-channel information. However, those studies differ from our work in various aspects, including their system settings and threat models, which bring about additional challenges that we address in this work.

One of the earliest works to document keylogging side-channel was introduced in the early 1970s [13] by researchers from Bell Laboratory who observed that an unattached oscilloscope showed "interesting" measurements whenever a key is pressed on a teletype terminal. Since then, keylogging side-channel attacks have become a topic of significant interest with many studies [28]. A central theme in those studies has been around two aspects: the discovery of mediums through which side-channel information is collected and the development of techniques and using them for recovering information from those signals to launch the inference attacks.

Exploited Medium for Attacks. Over the past few years, researchers addressed the first aspect by identifying and examining various mediums for side-channel information collection, including *acoustic emanations* [4, 9, 11, 16, 41], *electromagnetic emanations* [37], *vibrations* [7, 26], *motion* [23, 25, 38, 39], *visually observable clues* [6, 32], and *WiFi signal distortions* [3, 10], etc.

Acoustic Emanation for Inference. The first keylogging attack through acoustic emanation was presented by Asonov and Agrawal [4], where they developed and utilized a supervised learning algorithm in which the fast Fourier transform (FFT) coefficients of the audio signal are used as features. With their learning model, they recognized and classified the keystrokes of a particular user with an accuracy of 79%, which demonstrated the risks caused by acoustic emanations. Another study by Zhuang et al. [41] utilized unsupervised learning by using the cepstrum features of keystroke moments in the audio signal. The cepstrum is the result of the inverse Fourier transform (IFT) of the logarithm of the estimated signal spectrum. They show various promising results, uncovering 90% of 5-character random passwords using only letters in fewer than 20 attempts by the adversary. As an upgrade to the text inferences, a dictionary-based attack is proposed by Berger et al. [9], where the similarity of keystrokes in a word and the acoustic patterns learned from dictionaries are exploited to infer the typed text. They achieved 73% of accuracy from the top-50 word predictions.

Halevi and Saxena [16] performed another keylogging attack with a new similarity measure, the time-frequency classification, which considers time- and frequency-domain characteristics, and achieved 64% key detection accuracy. Compagno et al. [11] studied the risks due to the acoustic signals transmitted through IP telephony; the used the audio signals emitted by a keyboard at the victim's end to guess random keystrokes with 83% accuracy.

Motion-based Techniques. Another type of side-channel information is the readings of motion sensors, i.e., accelerometer and gyroscope. Liu et al. [23] exploited the signals retrieved from the accelerometer of a smartwatch to track the hand movements of a user and infer keyboard inputs with 55% top-5 word accuracy. Maiti et al. [25] used accelerometer and gyroscope readings to detect wrist movements and inferred the keystrokes with 51% top-10 word accuracy via physical position detection.

Besides the hand dislocation information retrieved from the motion sensors, Wang et al. [39] utilized a language model to further increase the leakage and achieved 30% top-5 word accuracy. Wang et al. [38] also performed training- and context-free attacks for key-based security systems, e.g., ATM and electric lock doors, and predicted what is entered by observing the motion trajectories of the smartwatch. The attacks recovered 80% of the PINs.

Vibration-based Techniques. The motions around a physical keyboard induced by typing actions creates a vibration on the underlying surface which propagates over short distances. Keylogging side-channel attacks exploited the vibration propagation by capturing it using a laser microphone [7] or a hijacked smartphone with capable sensors within proximity [26].

Visual Cues. Visual cues are also considered as side-channel information in the literature. Sabra et al. [32] proposed an attack framework that only uses the video feed in video call software, such as Skype or Zoom, to perform a keylogging side-channel attack. In this attack framework, they performed a dictionary-based

Table 1: A summary of the related work. The detection is measured by TPR (True Positive Rate). UP in training stands for User Profiling, and HM in the exploited medium stands for Hand Movement. Blank proximity and detection indicate that those values are not provided in the corresponding study.

Reference	Year	Exploited Medium	Proximity	Training?	Performance	
					Detection	Identification
Asonov et al. [4]	2004	Keyboard acoustics	1m	Yes (UP)	-	79% (top-1 key)
Berger et al. [9]	2006	Differences of keyboard acoustics	-	No	-	73% (top-50 word)
Balzarotti et al. [6]	2008	Video of the typing hands	<1m	No	-	46% (top-1 word)
Zhuang et al. [41]	2009	Bootstrapped keyboard acoustics	-	No	-	90% (top-1 word)
Marquardt e al. [26]	2011	Vibrations sensed via smartphone	50mm	Yes (UP)	-	43% (top-10 word)
Halevi et al. [16]	2014	Keyboard acoustics	-	Yes	-	64% (top-1 key)
Ali et al. [3]	2015	WiFi CSI distortion	4m	Yes (UP)	98%	96% (top-1 key)
Chen et al. [10]	2015	WiFi Multipath localization	5m	Yes (UP)	-	92% (top-5 key)
Wang et al. [39]	2015	Dislocation of hand	Smartwatch	Yes	57%	30% (top-5 word)
Liu et al. [23]	2015	HM+Acoustic emanations	Smartwatch	Yes	-	55% (top-5 word)
Maiti et al. [25]	2016	HM+Acoustic emanations	Smartwatch	Yes	-	51% (top-10 word)
Wang et al. [38]	2016	Hand movement	Smartwatch	No	-	80% (top-1 PIN)
Compagno et al. [11]	2016	Acoustics via VoIP	Remote	Yes (UP)	-	83% (top-1 key)
Sabra et al. [32]	2020	Video feed	Remote	No	92%	35% (top-50 word)
SIA	2021	Acoustics via Smartwatch	Remote	Yes (UP)	99%	98% (top-1 key)
SIA	2021	Acoustics via Smartwatch	Remote	Yes	99%	85% (top-1 key)

attack considering the displacements in between video frames and achieved 35% top-50 word accuracy with their practical settings.

WiFi Signals. Any motion in an environment with wireless signals distorts the signals, which is then leveraged as side-channel information for keylogging attacks. Ali et al. [3] exploited patterns in the time series of the Channel State Information (CSI) values between sender and receiver WiFi devices for inference. Their method achieved 96% top-1 key accuracy. Chen et al. [10] used five antennas to localize the hand movements using the WiFi signals to trace keystrokes, which resulted in 92% top-5 key accuracy.

A summary of the related work and a comparison with SIA, across various aspects, is shown in Table 1.

3 SYSTEM AND THREAT MODEL

In the following we review the details of the system model in which our attack is launched, and the threat model, which characterizes the capabilities of the adversary under which the attack is viable.

System Model. In this paper, we assume a system that consists of a user typing on a keyboard to input various texts, e.g., email addresses, passwords, etc., to a computer terminal. We also assume that the user is equipped with a smartwatch that features a microphone. We note that the overwhelming majority of smartwatches on the market today are equipped with microphones (e.g., Apple Watch, Fossil Gen 5 Carlyle, TechWatch Pro, Samsung Galaxy Watch, Huawei Watch 2, etc.). We also assume that the user is equipped with a smartphone (although this assumption is only necessary for rationalizing and demonstrating multiple attack avenues, it is not necessary for the attack in the abstract).

Threat Model. In this paper, we assume an adversary that is consistent with assumptions made in the literature concerning adversaries' objectives and capabilities. Namely, the objective of the adversary in our threat model is to infer what the targeted user in our system model is typing on the keyboard by utilizing the acoustic signals associated with the keystrokes.

In our threat model setting, we assume that the targeted user types some passage, or username password tuples *while wearing the smartwatch*. We consider two plausible scenarios for our threat model, as shown in Figure 1: (i) the targeted user types a passage while wearing a smartwatch that is *infected by a malicious application with access to the microphone* (Scenario 1), or (ii) the targeted user is engaging with the adversary in a phone call while typing on the keyboard (Scenario 2). In both cases, we assume that the adversary and the targeted users are not in the same physical space, which further emphasizes the power and versatility of the remote adversarial setting. In Scenario 1, the smartwatch records the surrounding acoustic signals while the targeted user is typing on the keyboard and uploads the recordings to a server maintained by the adversary. In Scenario 2, the adversary and the targeted user are in a phone call and the adversary records the call.

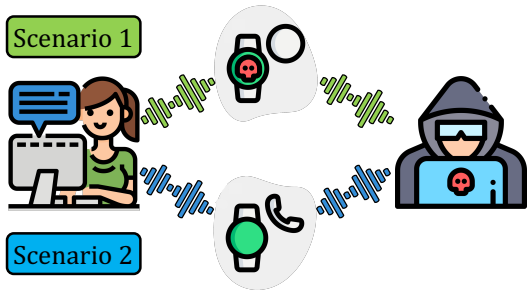


Figure 1: System and Threat Model Scenarios. Scenario 1 assumes an infected smartwatch for data transmission, while scenario 2 exploits the acoustic emanations in a phone call.

Challenges. While our system and threat models are to a great extent consistent with the literature, the fact that we use a physical keyboard and a smartwatch as the source and the recording devices, respectively, bring about four challenges (**C-1**–**C-4**), as follows:

C-1. The smartwatch mobility adds another layer of challenges, impacting the observed signal quality and consistency.

C-2. The difference in background noises creates an unknown factor that vastly affects the prediction performance.

C-3. Similar to **C-2**, the changes in the environmental setting affects the acoustics scale, causing identification inaccuracies.

C-4. The lower quality of the smartwatch microphones due to the space constraints enforced by the wearability influences the signal quality, which implicitly affects the predictions.

In the Methodology section (§ 4), we address those challenges.

4 SIA EXPLAINED

In this section, we introduce the methodology of SIA, followed for implementing the attack objectives in our threat model. Figure 2 demonstrates SIA's pipeline. The attack takes a recording denoted as raw signal as input, which is readily available to the attacker due to the above threat scenarios and outputs a prediction of what is typed in the input recording by analyzing the signal. The attack pipeline consists of four main stages: *Noise Cancelling*, *Keystroke Detection*, *Key Identification*, and *Word Correction*. We elaborate on each of those stages in the following subsections.

4.1 Noise Cancelling

Since the recording devices record every sound that exists in the environment, they record the background noises, which affect the quality of the recordings. The type and structure of the noise depend on various external and internal factors, such as the ambient noises (external) and the quality of the recording equipment (internal). Especially the inferiority of a microphone embedded in the smartwatch has the biggest influence on the background noises.

During our preliminary experiments, we observed that the alternations of the background noise significantly affect the performance of the identification. Therefore, as a first step, we prepare the data by cleaning any background noise. Two types of background noise exist, which we address in the following: (i) the white noise (hiss) and (ii) other ambient noise in the environment, e.g., street noises, computer fan noise, etc.

4.1.1 White Noise Canceling. Our white noise canceling algorithm utilizes the Fourier analysis where the fingerprint of the static background noise is structured using the spectrum of the pure tones in the quiet parts of the recording.

First, the noise in the whole recording is filtered using the fingerprint of the background noise. This fingerprinting is a crucial step, since cleaning the frequency bands of the white noise directly from the recording also cleans the actual keystroke acoustics.

Figure 3 shows the spectrum of the white noise and the acoustics of a keystroke event. The significant overlap in the frequency bands restrains us to directly crop the frequency components structuring the white noise. Second, to facilitate processing, the recording is divided into segments and the frequency spectrum of each segment is calculated. Third, the spectrum of each segment is analyzed such that the volume of any pure tones that are no louder than the average levels obtained in the fingerprint is reduced. This procedure is typically named as *spectral noise gating*.

More technically, in the first step of *Noise Cancelling*, the Fast Fourier Transform (FFT) using a Hann window is calculated for each windowed segment of the recording. FFT requires data with a certain length (2048). Therefore, the signal is divided into equal-length segments, which may cause discontinuity around the edges. The Hann windowing corrects such discontinuities at the edges of the segments before they are forwarded to the FFT. After having the spectrum (FFT of a time-domain signal), statistics, including the mean power, are tabulated for each frequency band. Those statistics and the *sensitivity* parameter determine a threshold for each frequency band. Gain control for each frequency band is set such that if the sound exceeds the threshold, the gain is set to 0dB, otherwise, the gain is set to the *Noise Reduction* parameter (e.g., -12dB), to suppress the noise. Next, time smoothing is applied to obtain a smooth transaction over frequency bands. Then, frequency smoothing is applied to avoid suppressing or boosting a single frequency in isolation. Finally, the gain controls are applied to the FFT of the signal and the inverse FFT is applied, followed by another Hann window. The output signal of each segment is combined to structure the whole recording of which the noise is reduced.

4.1.2 Ambient Noise Canceling. Due to the unpredictable nature of ambient noise, i.e., what quantity of noise (what) and wherein the time domain it is injected (when), canceling it is more challenging than canceling the white noise. In this part, we first manually locate the signal pieces where an ambient noise profile is intertwined with keystrokes to answer the "when" question. For each piece, the noise profile is structured as done with the white noise, which answers the "what" question. The noise profile is then removed only from the corresponding signal piece. Hereafter, we refer to the output of this stage by the "clean signal".

4.2 Keystroke Detection

For *Keystroke Detection*, we locate the keystroke events in a clean signal, a task that is possible using two observations: (i) a keystroke event yields two peaks, a hit peak and a release peak, in the acoustic signal which lasts typically for 200 ms total together with the small inactive portions at the start and the end [4, 11, 26, 28, 41], (ii) the acoustic signal emanated when a key is pressed is more powerful than the one when the key is released.

Based on these observations, we simplify the keystroke detection problem to finding the strong peak observed in a keystroke event. Locating the strong peak suffices to locate a keystroke event by encapsulating the hit peak with a 200 ms window. For the simplified version of the detection problem, the solution is based on another observation: a typical hit peak lasts for only 10 ms. Therefore, we slide a 10 ms window, w, over the clean signal and calculate the spectrum energy E along the way, yielding a set of window-energy (w_i, E_i) tuples (Part b in Figure 4). Then, the maximum number of keystrokes, n, is calculated considering the typical duration of a keystroke (200 ms), the typical pause duration (500 ms) between keystrokes, and the length of the recording. Since the solution builds on the idea that the energy of a hit peak is greater than the other parts, it is reasonable to assume that the windows with high energy are likely to contain the hit peak.

We capitalize on this observation by sorting the (w_i, E_i) tuples in terms of the energy component E_i (Part c in Figure 4). From this sorted list, `energy_list`, we start to form the actual 200 ms windows, W_i, encapsulating a whole keystroke event. However, we observe that the `energy_list` may contain 10 ms windows that are encapsulated in a single 200 ms window. To avoid creating multiple keystroke windows for a single keystroke, we use a suppression technique: whenever a 200 ms window, W_i, is created, the suppression technique discards the 10 ms windows that overlap with W_i

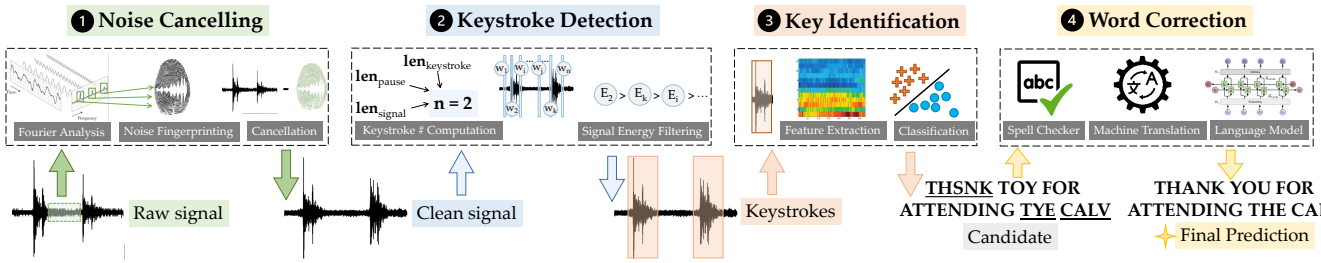


Figure 2: SIA **consists of four main stages:** *Noise Cancelling*, *Keystroke Detection*, *Key Identification*, **and** *Word Correction*. *Noise Cancelling* **takes the raw signal, cancels any ambient background noise or white noise (hiss) from the signal, and returns the clean signal.** *Keystroke Detection* **takes the clean signal and returns 200 ms windows encapsulating the keystroke events.** *Key Identification* **takes the windows and predicts the associated characters.** *Word Correction* **takes the predictions and corrects the misspelled words.**

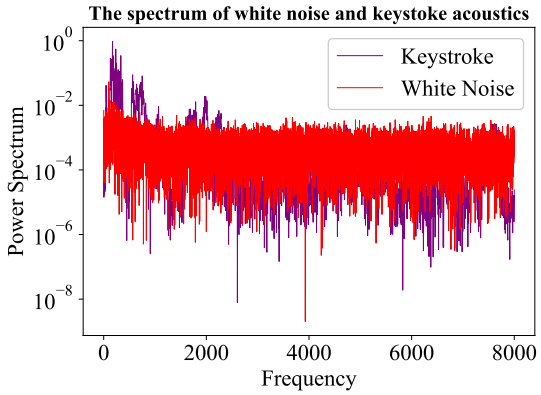


Figure 3: The significant frequency band overlap between the white noise and the acoustics of a keystroke event.

from the `energy_list` (Part d in Figure 4). Once the number of 200 ms windows, i.e., keystroke events, reaches n, the events are sorted in terms of the timestamps, and the detection process is completed.

4.3 Key Identification

After locating the keystroke events, the next step is to find which signal belongs to which key. For that, we extract various discriminative features from the time-domain signal and use a multi-class classifier to determine the key that is pressed as our outcome.

4.3.1 Feature Extraction. The most common feature candidates for audio are evaluated before deciding which one is utilized. In the following, we consider the Fast Fourier Transform (FFT) coefficients, Cepstrum coefficients, Mel-Frequency Cepstral Coefficients (MFCCs), and Chroma features.

FFT Coefficients. FFT is an optimized algorithm that computes the Discrete Fourier Transform (DFT) of a signal. DFT, implicitly an FFT, converts the input signals from their original domain (typically time or space) to the frequency domain, where the FFT coefficients are the coefficients of this frequency-domain representation. FFT coefficients signify the frequency components, which is useful for associating similar signal structures.

Cepstrum. Cepstrum converts a signal in the time domain to a signal in the quefrency domain. The quefrency domain is intuitively

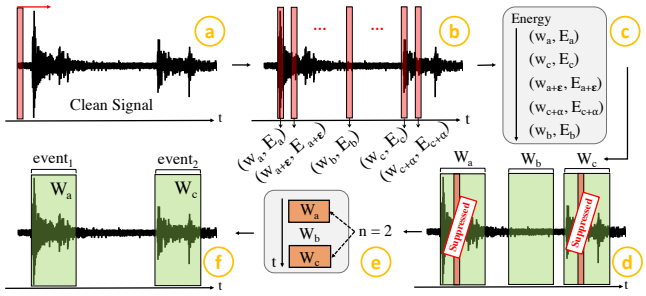


Figure 4: The maximum number of keystrokes, n, is computed considering the length of the clean signal coming from *Noise Cancelling* **stage. (a) A 10 ms window is slid over the clean signal. (b) The spectrum energy, E_a, of each 10 ms windows w_a is calculated. (c) The window-energy tuples, (w_a, E_a), are sorted in terms of the energy. (d) Starting from the most powerful 10 ms window, 200 ms windows, W_a, encapsulating keystroke events are created. Whenever a new window is created, the subsequent 10 ms windows $(w_{a+\epsilon})$, which are less powerful and overlaps with it, are suppressed. (e) The n-most powerful 200 ms windows are fetched and sorted in time. (f)** *Keystroke Detection* **returns a set of windows encapsulating the keystroke events.**

defined as the rate of change in the different spectrum bands, and its formulation, for a signal $f(x)$, is given as follows:

$$C_p = |\mathrm{FFT}(\log(|\mathrm{FFT}(f(x))|^2))|^2. \quad (1)$$

The inner FFT function in Equation 1 converts the signal to the frequency domain and the outer FFT converts the frequency domain to the quefrency domain. The Cepstrum method is effective for pitch detection in human speech. Since a keystroke event consists of peaks, Cepstrum is a preferred candidate feature for our task.

MFCC. MFCCs are derived from the cepstral representation of a sound signal, i.e., Cepstrum. MFCC differs from Cepstrum by the spacing of the frequency bands. In MFCC, the frequency bands are spaced on the mel scale which approximates the human's auditory perception. In Cepstrum, the frequency bands are linearly-spaced. In other words, with the mel scale, the human perceptible portions of the audio are boosted and are made more distinguishable.

Table 2: A comparison of feature options for *Key Identification* in terms of True Positive Rate (TPR). Mel-Frequency Cepstral Coefficients (MFCC) performed best for the task.

Feature	FFT	Cepstrum	MFCC	Chroma
TPR	0.07	0.70	0.85	0.37

MFCC is commonly used in speech recognition, music information retrieval, and audio similarity measurements; its performance is proved to be successful on these tasks. Since our classification is based on the similarity of the sounds, MFCC is an appropriate feature candidate for our purpose.

Chroma. Chrome features, also known as the pitch class profiles, are most capable of providing high-quality representation when pitches exist in the audio. Chrome features are particularly powerful for music audio where the spectrum is projected onto 12 bins, i.e., 12 chromas. Chroma features are designed based on the observation that notes one octave apart are perceived as similar. Therefore, knowing about Chroma even without the frequency information can give us insights about the similarity. Although it is handy for music audio, it would still give us some sense of similarity; thus it is worth exploring as a feature for our purpose.

We conducted experiments on a small data sample and found out that MFCC stands out among the other approaches discussed above (the results are shown in Table 2).

4.3.2 Multi-Class Classification. Once the features of the signal are obtained, the detection of the keys associated with the signal is done by employing a machine learning algorithm in the multi-class classification task. For our multi-class classifier, we considered multiple models, which ranged from simple to more complex: logistic regression, support vector machine, multi-layer perceptron, and convolutional neural network, which we review in the following.

Logistic Regression (LR). LR explains the relationship between one dependent binary variable and one or more independent variables. Although LR is a probabilistic model typically used when the target (dependent variable) is binary, the idea behind it can be extended to the multi-class classification using a one-vs-rest scheme, in which a classifier per class is trained to return positive if the sample belongs to the class, and negative otherwise. LR assumes that data from the different classes have no high correlation. In some cases where our data show similar patterns, especially when the keys are in close proximity, LR is not an ideal choice and comes second per our empirical results (Table 3).

Support Vector Machine (SVM). SVM aims to find a set of hyperplanes that best separate classes in the feature space by maximizing the margin from the hyperplane to the data points. SVM is ideal for binary classification. Similar to LR, the one-vs-rest scheme is used to support multi-class classification. We used the Radial Basis Function (RBF) kernel, and we set the regularization parameter, which dictates the degree of importance given to misclassifications, to 1.0. Since SVM outperformed the other classification methods, we employ SVM as the multi-class classifier in our study.

Multi-Layer Perceptron (MLP). MLP is the stepping stone to the deep learning area and is a feedforward fully-connected neural network capable of solving complex problems. Using non-linear activation functions, MLP can capture complex relations/patterns

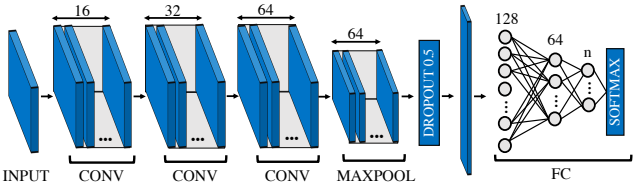


Figure 5: The Convolutional Neural Network (CNN) architecture employed during the classifier selection. Three subsequent convolutional layers (*CONV*) with the depth of 16, 32, 64 are utilized. Then, a *MAXPOOL* layer selects the maximum weights in the convolutions. Before the fully-connected layer (*FC*), a dropout regularization is performed to avoid overfitting. The softmax at the end of *FC* returns the class probabilities for the given input.

in data, which is helpful considering our feature space. Although we used a quite deep architecture (100 hidden layers), MLP cannot provide an optimal recall for the task.

Convolutional Neural Network (CNN). CNN is one of the most advanced deep learning methods in the literature. With the help of convolutional layers, CNN can capture not only the complex relations but also the temporal patterns in a given sample.

The CNN architecture used in this evaluation is demonstrated in Figure 5. First, the input (MFCC features) is reshaped into a two-dimensional (2D) array. Three consecutive convolutional layers (`CONV`) are then applied to the input. Due to the enlarging depth in the convolutions, the architecture becomes more capable of capturing complex patterns in the input. The kernel size of each convolutional filter is determined as (3×3) and the stride is determined as (1×1). The following max pooling layer (`MAXPOOL`) then fetches the most important portions of the features by selecting the maximum weights in (2×2) kernel with (1×1) stride. Such a deep architecture has lots of weights/parameters which typically lead to overfitting. To avoid overfitting, a dropout regularization [33] with 0.5 probability is applied to the output of `MAXPOOL`. Next, the output of the dropout layer is flattened and forwarded to the fully-connected layer (`FC`). Softmax at the end of `FC` computes the class probabilities, and the most likely class is returned as the prediction. For all the applicable layers, i.e., `CONV` and `FC`, the ReLU activation function is used. Adam [17] optimization is utilized for training and the architecture is compiled with the Categorical Cross-Entropy (CCE) loss function. CCE is the combination of a softmax ($f(\hat{y})$) and cross-entropy loss:

$$f(\hat{y})_i = \frac{e^{\hat{y}_i}}{\sum_j^C e^{\hat{y}_j}} \qquad CCE = -\sum_i^C y_i \log(f(\hat{y})_i), \tag{2}$$

where y is the target vector, $\hat{y}$ is the output of the model, C is the number of classes. CCE loss is beneficial for multi-class classifications where the last layer is a softmax, and the target vector can be represented as a one-hot vector.

Preliminary Results. We use those techniques for comparison, and our empirical results with MFCC features (Table 3) showed that SVM performed better than any other algorithm for the given classification task. Since MFCC features are frequency-domain features, the capability of capturing temporal patterns could not give much

Table 3: Comparison of different multi-class classifiers with MFCC features in terms of TPR. Support Vector Machine (SVM) is the best classifier alternative.

Classifier	LR	SVM	MLP	CNN
TPR	0.83	0.85	0.73	0.73

of an advantage for our task. Therefore, CNN could not perform better when compared to others.

4.4 Word Correction

We observed that *Key Identification* stage may produce misspelled word predictions due to the signal similarities, especially between the keys are in close proximity. To further improve the prediction accuracy, we added a *Word Correction* stage to the attack pipeline. We evaluated three methods for *Word Correction*: Simple Spell Checker, Machine Translation, and Next Word Prediction. Each method is explained in the following.

4.4.1 Simple Spell Checker. In the simple spell checker (SSC) method, an algorithm based on the Levenshtein distance is used to find permutations within an edit distance of 2 from the original word [29]. Levenshtein distance (LD) [21] is a string metric that measures the difference between two sequences. It counts the minimum number of single-character edits, such as insertion, deletion, or substitution required to make two strings identical. After finding all candidates within LD of 2, all permutations are compared to known words in a word frequency list. Those words that are found more often in the frequency list are more likely the correct results.

4.4.2 Machine Translation. In the machine translation (MT) method, we utilized one of the most advanced approaches in Natural Language Processing (NLP): Transformers. With the help of transformers, the spell correction task is converted into a machine translation task, an NLP task where two languages are automatically translated. For this, we used the `xfspell` [15] tool. `xfspell` views the misspelled words and their correct versions as languages, and trains the network accordingly. For training, the tool developer mined data from GitHub. The commits correcting 'typos' are fetched and the so-called language datasets are built. However, such training data yielded highly technical spell corrections. To avoid this bias, the developer used the machine translation technique itself to enhance the datasets. By reversing the machine translation model, he provided correctly spelled English words and got the misspelled version from the other end, and retrained the model.

Although this is in many ways a creative method for addressing this task, the misspelling errors we observed in the key identification predictions are unusual. For instance, the sample 'CONCERNS' can be predicted as 'DLMCERNX' where the mispredicted characters are *one-hop neighbors* of the actual character. Such spelling errors may remain uncovered with such a method. However, the idea is worth further exploration. To this end, we can slightly modify the `xfspell` tool by providing different *languages*. The misspelled dataset can be synthetically created by modifying the correct words. Then, the model can be retrained with the correct and misspelled correspondences. We left this improvement as future work.

4.4.3 Next Word Prediction with Language Model. The idea of using a next word (NW) prediction as a word correction mechanism is based on the following observation: some word predictions are not misspelled, but simply do not align with the context of the sentence. For example, the predicted sentence "Thank toy for attending the call." has no misspelled words. However, if the context is considered, the word that does not align with the context can be predicted using the next word prediction methods to convert the original sentence into "Thank you for attending the call." In this method, we utilized a state-of-the-art GPT-2 [31] language model as a next word predictor. GPT-2 uses a unidirectional transformer model that is pretrained using language modeling on a very large corpus of 40 GB of text data. We feed the predicted words, $word_i$ for all i, to the model sequentially and get the most likely next word $word_i^n$ from the model. If the likely next word produces a better score than the actual prediction, we replace $word_i$ with $word_i^n$. Otherwise, we keep $word_i$ as it is. Although this method corrects the out-of-context words, in some cases it can still decrease the overall accuracy of the prediction. For example, for the predicted sentence "Please let me know if you have any **zlmcervx**.", this model outputs "Please let me know if you have any **questions**.". For this particular case, even though the new word completes the sentence by following the context, it changes the word from "concerns" to "questions" which reduces the prediction accuracy.

5 EVALUATION

For our experiments, we collected three types of test data for two keyboards from two users, and evaluated SIA pipeline, including *Keystroke Detection*, *Key Identification*, and *Word Correction* stages. In the following, we elaborate on the data collection process and the experiments of the pipeline stages.

5.1 Data Collection

We collected two sets of data: training data and test data, using various devices, which we choose for their popularity. For both the training and test data collection, Samsung Galaxy Watch Active 2 and Samsung s10e Smartphone are used as recording devices; a MacBook Air with Magic Keyboard (MBA) and a Bluetooth Magic Keyboard (BMK) are used as the target keyboards. All used devices were among the best selling in 2020.

For data collection, we implemented a keylogger interface that logs the timestamp and key information when a key is pressed. Using the timestamps, each keystroke event is encapsulated in a 200 ms window and the window is labeled with the key.

Two users are used to perform data collection, USER A and USER B, and they placed the keyboard in a convenient position. They wore the smartwatch on their left wrists and put the smartphone to the left side of the keyboards. There are no enforced constraints in terms of which side they should wear/put the smartwatch/smartphone. The dataset and the keylogger can be found in SIA Git Repository.

5.1.1 Training Data. In training data collection, for each keyboard-recording device combination, 45 pangrams are typed by a single user (USER A), acting as an adversary, and the acoustic emanations are recorded. A pangram is a string that includes all alphanumeric characters (36 characters) in the English language. The order of the characters is randomized in the pangrams. Therefore, the transition movements from one key to another, which may affect the acoustic emanations, are randomized implicitly. The training data is recorded

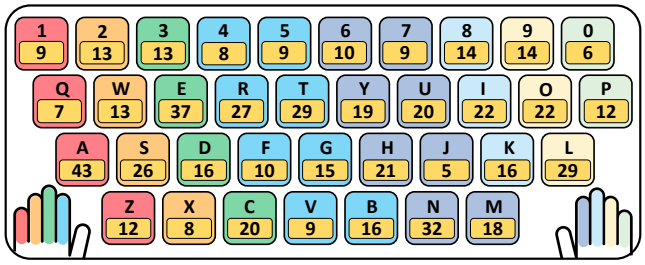


Figure 6: The cumulative character-wise distribution of the test datasets. The colors on the keys encodes the associated hand and finger.

in a quiet room, without any ambient noise. The white noise is canceled using the *Noise Cancelling* stage of the pipeline.

Scaling is an important step for optimizing the learning process. The scale and distribution of the data may be different for each sample. Differences in the scales across samples may complicate the problem being modeled. For instance, large input values can result in large weights in the model. Large weights make a model unstable - the model with large weights performs poorly during learning and may have high sensitivity to the input values resulting in generalization errors. Thus, before training, the training data is scaled using a `MinMaxScaler` and the scaler parameters are saved for scaling the test data.

5.1.2 Test Data. Three types of test data for each target keyboard and each recording device are collected from two users, USER A and USER B. (1) **E-mail:** A randomly selected e-mail sample from Enron mail dataset [18]. The e-mail consists of 213 letters and 42 words. (2) **Random Password:** 32 randomly generated passwords of length 8. Hereafter, this data is referred to as "Random". (3) **Selected Password:** 20 randomly selected passwords from the RockYou password dataset. Hereafter, this type of data is referred to as "Selected". Figure 6 shows the character-wise distribution of the characters in the test dataset.

The first user (USER A) participating in the test data collection is the same person collecting the training data. Since the model in *Key Identification* is trained with the data collected from the same user USER A, the experiments on the USER A's test data are considered as *user profiling*(§ 5.2). The experiments on the USER B's test data are considered as a *practical attack*(§ 5.3).

Unlike in the prior studies [11] where the evaluations are done with the k-fold cross-validation techniques, we recorded our test data in a different room which demonstrates the practicality of the attack. For example, Compagno et al. [11] collected a single dataset and directly evaluated the performance using k-fold cross-validation without noise canceling. For comparison, we implemented their method and tested our data on it. Table 4 shows the empirical result of this comparison.

We observed that Compagno et al.'s method (Noisy Data + LR) does not work as well when the test data is recorded in a different room (we label this recording as "Practical" in Table 4). Using the k-fold cross-validation method is *implicitly* biased for such evaluation. Because the test split contains very similar samples to the training split when the environmental acoustics are stable, which is a very idealized environment setting/assumption and yields impractical experiments. Table 4 also demonstrates that SIA outperforms Compagno et al. [11] (0.85 for SIA vs 0.53 for their model). We emphasize that our better results in comparison with [11] are obtained in a more practical system setting: Compagno et al. [11] has an advantage (strong assumption; weakness) in the threat model, where they assume the recording device to be static and fixed in location (laptop microphone). SIA, using the smartwatch, on the other hand, uses a moving recording device with the user's wrist, which affect the quality of the observed raw signals.

Table 4: A comparison between different inference techniques (Clean Data+SVM / Noisy Data+LR) and evaluation methods (5-fold / Practical). In the practical evaluation, the test data recorded in a different environment is used.

Method	5-fold	Practical
Clean Data + SVM	0.99	0.85
Noisy Data + LR	0.94	0.53

5.2 User Profiling

In this set of experiments, we trained the model in *Key Identification* with the data recorded by USER A and used the USER A's test data, i.e., USER A is the targeted user. Such evaluation setting assumes an adversarial strength: the adversary has some knowledge about the acoustic emanations of the targeted user's unique typing style. In the following sections, we elaborate on the evaluation of each stage of the pipeline.

5.2.1 Keystroke Detection. *Keystroke Detection* returns a set of windows, $W_{\text{keystrokes}} = \{W_i\}_{i=1}^{n}$, and those windows encapsulate the keystroke events. Given each ground truth keystroke event gt_j in $GT = \{gt_j\}_{j=1}^{M}$, and upon prediction, we associate each key tap segment W_i in $W_{\text{keystrokes}}$ with the ground truth event gt_j which is closest to W_i in time.

For *Keystroke Detection*, as an evaluation metric, we used the intersection-over-union (*IoU*), which is a common evaluation metric for temporal localization problems. IoU is a measure of the overlap ratio between the detected and the ground truth keystroke events in time. For two keystroke events, $E_1(t_1, t_2)$ and $E_2(t_3, t_4)$, where t_1 and t_3 are the start, t_2 and t_4 are the finish timestamps, and $t_3 > t_1$, *IoU* between E_1 and E_2 is:

$$IoU(E_1, E_2) = \frac{\min(0, t_2 - t_3)}{(t_4 - t_1)} \in [0, 1]. \tag{3}$$

When the events fully overlap, the *IoU* value is 1. This process produces a set of associations (A) of keystroke events together with their *IoU* values. We then eliminate the associations with $IoU < 0.75$ from A, where 0.75 is the *IoU* threshold that we set. (Typical *IoU* threshold is set as 0.5 in temporal localization problem—higher value means more overlap.) Following the common evaluation methodology in temporal localization problems, we interpret the events in A as true positives, $W_{\text{keystrokes}} \setminus A$ as false positives, and $GT \setminus A$ as false negatives. Table 5 shows the TPR, False Negative Rate (FNR), and Precision (P) of the *Keystroke Detection* on each dataset-keyboard-recorder combinations.

Table 5: Keystroke detection results. The IoU threshold is 0.75. True Positive Rate (TPR), False Negative Rate (FNR), and Precision (P) values for each test data (E-mail/Random/Selected), recording device (Phone/Watch), and target keyboard (MBA/BMK).

Device	MBA (Watch)			BMK (Watch)			MBA (Phone)			BMK (Phone)		
Metrics	TPR	FNR	P	TPR	FNR	P	TPR	FNR	P	TPR	FNR	P
E-mail	0.994	0.006	0.900	0.965	0.035	0.754	0.994	0.006	0.855	0.994	0.006	0.855
Random	0.996	0.004	0.988	0.984	0.016	0.933	0.988	0.012	0.941	0.996	0.004	0.910
Selected	0.994	0.006	0.900	0.983	0.017	0.818	0.988	0.012	0.895	0.967	0.033	0.931

Observations on Random Text. Users typically type relatively slower when typing random text, since they have to keep track of the next character by looking at the display when knowledge of the text is of very limited value. Therefore, the margin between the keystroke events is wider. When two keystroke events are wide apart, the distinct gap in between two keystroke events facilitates the keystroke detection. Therefore, the TPR in keystroke detection for the random dataset is generally higher than the others. Overall, the low FNR, high precision, and TPR demonstrate the robustness of the keystroke detection method.

5.2.2 Key Identification. In this section, we discuss our results for the *Key Identification* stage for all devices and datasets. The *Key Identification* stage takes keystroke event windows W from the *Keystroke Detection* stage and returns a string corresponding to the given keystroke events. Each keystroke event window W is forwarded to the trained model and the most likely class is assigned to the window. For the *Key Identification* evaluations, we performed three sets of experiments: (i) character-wise TPR for each keyboard and recording device, (ii) cross-entropy of the probability distribution estimated by the model, and (iii) string-wise evaluations, where we calculated the TPR and the Normalized Levenshtein Distance (NLD) for each recording device, keyboard, and test dataset. Before discussing the results, we present the evaluation metrics.

Cross-entropy. The cross-entropy is a measure of the difference between two probability distributions for a given random variable. To evaluate the performance of our classifier, we used the cross-entropy of the predicted probability distribution, f, relative to the actual distribution, p. The cross-entropy formula is given as:

$$H(p, f) = -\sum_{i} p(x_i) \log(f(x_i) + \epsilon) \tag{4}$$

The ϵ value in (4) is used to avoid the undefined values yielded by log(0). For our evaluations, we calculated (i) the cross-entropy between the probability distribution that our model estimates and that of the actual distribution, $H(p, f_{\text{SVM}})$, (ii) the cross-entropy between the uniform distribution and the actual distribution, $H(p, f_u)$, and (iii) the cross-entropy between the actual distribution and itself representing the ideal cross-entropy, $H(p, p)$. $H(p, p)$ is calculated as a reference point for all other cross-entropy calculations.

Normalized Levenshtein Distance (NLD). As briefly mentioned above, LD is a string metric that measures the difference between two strings. It measures the minimum number of edits (insertion, deletion, or substitution) to make the strings identical. For our experiments, we used its normalized variant as a string comparison metric. NLD is calculated by dividing the LD between two strings (predicted and target) by the length of the target string.

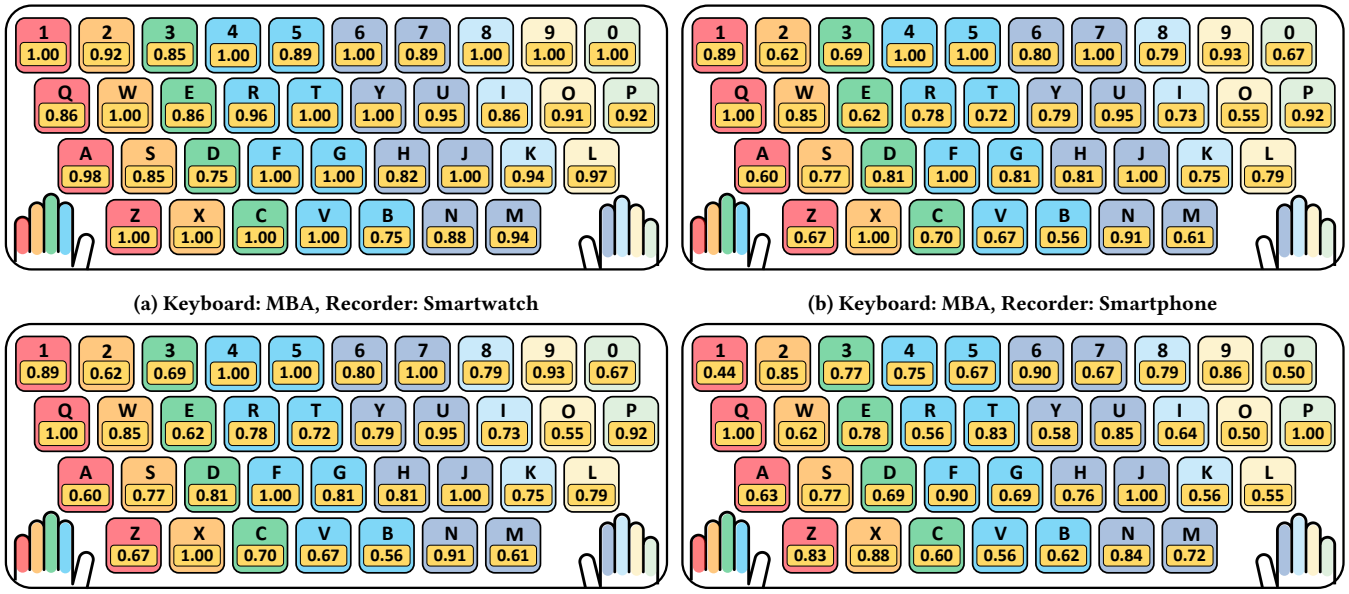
Table 6: Similarity measures for the keyboards. The average cross correlation of the key tuples that are "most confused" and "never confused" by the keyboard models. BMK emanates more similar acoustics than MBA.

Setting	MBA	BMK
Most Confused	67895	104834
Never Confused	54219	80835

Character-wise TPR. Figure 7 shows the character-wise TPR for each recording device and keyboard. While computing the TPR for each character, the whole test data is concatenated into one big test dataset and forwarded to the model for prediction. Then, a 36×36 confusion matrix is generated. From the confusion matrix, the TPR for each class, i.e., character, is calculated. Figure 7a shows the character-wise TPR for the MBA keyboard and smartwatch recording device. The TPR values range from 0.75 to as high as 1.00, which demonstrates the success of the attack on the MBA keyboard. Figure 7a also shows the location of each key and the colors emphasize the finger that presses that key. The most successful predictions, with an average of 0.96, are done on the keys pressed with the left little finger, left index finger, right ring finger, and right little finger. The least successful predictions are recorded with the left middle finger with an average of 0.86.

Figure 7c shows the character-wise TPR for the BMK keyboard recorded by the smartwatch. The results range from 0.55 to 1.00. For the BMK keyboard, the highest TPR, an average of 0.85, was observed with the left index finger. The smallest TPR is observed with the right middle finger, with an average of 0.70. When the used hands are considered, TPR for right hand (MBA $\rightarrow$ 0.94, BMK $\rightarrow$ 0.81) is slightly higher than that of left hand (MBA $\rightarrow$ 0.93, BMK $\rightarrow$ 0.78). This is reasonable, because the smartwatch is worn to the left wrist, and the location of the microphone does not change when the right hand is in use.

MBA vs. BMK. When the character-wise TPR results are considered as a whole, we observed that the attack performed better with the MBA keyboard than with the BMK keyboard. We claim that such an outcome is expected when a keyboard emanates "similar" acoustic signal from the different keys. To support this claim, we calculated the similarity between the keys using cross-correlation. To be able to compare the similarity, we calculated the similarity of (i) the key tuples that are most confused by the models, and (ii) the average similarity between the key tuples that are never confused by the model (Table 6). The similarity measurements on *scaled* data show that the BMK keyboard (80835) emanates more similar acoustics when compared to the MBA keyboard (54219). Our claim is further supported by the cross-correlation of the most confused keys. The average similarity between the most confused keys is greater than that of the never confused keys.



(a) Keyboard: MBA, Recorder: Smartwatch

(b) Keyboard: MBA, Recorder: Smartphone

(c) Keyboard: BMK, Recorder: Smartwatch

(d) Keyboard: BMK, Recorder: Smartphone

Figure 7: Character-wise TPR obtained on the test data after *Key Identification* stage for every keyboard model and recording device combinations. The colors encode the hand and finger used for each key.

Smartwatch vs. Smartphone. To compare the effect of the mobility introduced with the smartwatches, we performed the same evaluations with the data collected from the smartphone. In principle, we notice that when the overall TPR averages are considered, the models performed slightly better with the smartphone recordings than with the smartwatch. We note, however, that two key reasons contributing to this performance improvement: (i) the smartphone is static in position compared to the mobile smartwatch, (ii) the smartphone has a better recording quality, as a result of a higher quality microphone compared to that of the smartwatch. In particular, the smartphone is equipped with higher-grade microphones with a higher sampling rate (44.1 kHz) than smartwatches (16 kHz). For a comparable setting and to mitigate the gap in the microphone quality, we down-sampled the recordings from the smartphone and used them as our data source. Moreover, we observed some differences in the background white noises between the recordings coming from different devices. However, the *Noise Cancelling* stage of the pipeline is shown to be useful in normalizing the signals by addressing the gap and reducing the differences. Since we evaluated after eliminating the quality gap between recording using those steps, the increase in TPR shows that the mobility of the recording device has a slight performance effect.

Cross-entropy. To observe how much the attack improves the entropy, we computed the cross-entropy between the actual probability distribution, which is a one-hot vector (1 on the correct class), and the probability distribution returned by the classifier (Table 7). The perfect match with the actual distribution ("Ideal" column in Table 7) is a very small number ϵ. The **Frequency** column in Table 7 shows different cross-entropy values for the different datasets considering the character frequency for the corresponding domain. For the E-mail dataset, **Uniform** shows the cross-entropy with the character probability distribution of the English language. For the "Selected" dataset, it shows the cross-entropy with the character probability distribution against the RockYou leaked password dataset. For the "Random" dataset, it shows the cross-entropy against the uniform distribution. The entropy loss introduced with our method shows the severity of the attack.

String-wise Evaluations. We also evaluated the predictions' string-wise TPR against the ground truth. Table 8 shows the TPR, TPR with one-hop, NLD, and NLD with one-hop for each dataset, device, and keyboard model. The choice of one-hop is not arbitrary. During our experiments, we observed that some of the mispredicted letters are within the one-hop distance with the actual key (on the keyboard layout). This is reasonable since the acoustic emanations coming from keys within close proximity are similar to one another. To understand to what extend the mispredictions caused by the proximity affect the performance, we also computed the one-hop variations of the string measures. We found that the average improvement in the TPR of MBA is 4.5%, where that of the BMK is 14.3%. We can interpret these values as a measure of the confusion among the keys nearby. The high TPR improvement for the BMK further supports our claim that *BMK emanates more similar acoustics.*

Since the success of the keystroke detection propagates to the subsequent stages and keystroke detection performs better on Random, the TPR of Random is generally higher than the others.

5.2.3 Word Correction. We used the three methods outlined in § 4.4 for word correction. *Word Correction* stage is only applicable for the E-mail dataset since other datasets do not include actual English words. Table 9 shows the performance improvements of the word correction methods. SSC and MT improved the prediction's TPR in all cases. However, NW is shown to decrease the TPR in some

Table 7: Cross entropy shows the difference between two probability distribution. Ideal row is the cross entropy of the actual distribution (one-hot vector) with itself ($\epsilon = -1 \times 10^{-9}$). Frequency raw is the cross entropy between the actual distribution and the frequency distribution associated with each dataset (E-mail dataset (E) → English letter frequency, Random dataset (R) → Uniform distribution, Selected dataset (S) → Character distribution of RockYou leaked passwords) Ours row shows the cross entropy between the actual distribution and the distribution estimated by our method.

Device	MBA (Watch)			BMK (Watch)			MBA (Phone)			BMK (Phone)		
Dataset	E	R	S	E	R	S	E	R	S	E	R	S
Ideal	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ
Frequency	14.02	3.58	18.98	14.02	3.58	18.98	14.02	3.58	18.98	14.02	3.58	18.98
Ours	0.66	0.40	0.86	0.65	0.33	0.80	0.88	1.20	0.96	0.41	1.79	1.10

Table 8: String-wise evaluation results of *Key Identification* for all device-keyboard-dataset combinations. The predictions for the Key Identification stage are compared with the ground truth strings. E: E-mail, R: Random, and S: Selected datasets.

Device	MBA (Watch)			BMK (Watch)			MBA (Phone)			BMK (Phone)		
Dataset	E	R	S	E	R	S	E	R	S	E	R	S
TPR	0.918	0.972	0.878	0.796	0.812	0.679	0.901	0.980	0.933	0.901	0.777	0.442
TPR (1-hop)	0.988	0.992	0.939	0.918	0.921	0.834	0.971	1.000	0.966	0.976	0.929	0.712
NLD	0.065	0.025	0.110	0.164	0.177	0.290	0.079	0.018	0.060	0.798	0.210	0.505
NLD (1-hop)	0.009	0.007	0.055	0.065	0.073	0.150	0.023	0.000	0.030	0.018	0.066	0.260

Table 9: String-wise evaluation results of Word Correction on E-mail dataset for all device-keyboard-method combinations. The output of Word Correction stage is compared with the ground truth strings.

Device	MBA (Watch)				BMK (Watch)				MBA (Phone)				BMK (Phone)			
Method	None	SSC	MT	NW	None	SSC	MT	NW	None	SSC	MT	NW	None	SSC	MT	NW
TPR	0.918	0.970	0.953	0.866	0.796	0.877	0.970	0.872	0.901	0.959	0.970	0.912	0.901	0.965	0.970	0.918
TPR (1-hop)	0.988	0.988	0.994	0.936	0.918	0.930	0.988	0.953	0.971	0.988	0.994	0.965	0.976	0.971	0.988	0.982
NLD	0.065	0.023	0.037	0.112	0.164	0.023	0.023	0.103	0.079	0.033	0.023	0.070	0.798	0.028	0.028	0.065
NLD (1-hop)	0.009	0.009	0.004	0.053	0.065	0.009	0.009	0.037	0.023	0.009	0.004	0.028	0.018	0.023	0.023	0.014

cases, for the reasons highlighted earlier: since NW may replace the misspelled word (e.g., "nusiness" for "business") with another word (e.g., "work") fitting in the sentence context, it may decrease the TPR in some cases, leaving some room for improvements.

5.3 Practical Attack

Now we introduce the results of a practical instance of SIA. The result of the practical attack demonstrates the applicability and generalizability of the attack to arbitrary individuals. For this attack, we trained the model using the training data recorded by USER A and used the test data collected from USER B as the test data. In such setting, USER A acts as the adversary targeting USER B.

First, we forwarded the test data through the attack pipeline. The background noises are canceled, and the keystroke events are detected. The MFCC features of each signal associated with a keystroke event are then extracted. The MFCC features are then scaled using the MinMax scaler parameters of the *training data collected from* USER A. Then, the features are forwarded to the SVM model in the *Key Identification* stage that is previously trained with the data of USER A. The predictions of the Selected and Random test data are obtained after the *Key Identification*. Finally, E-mail test data is forwarded to the *Word Correction* stage and the last predictions are obtained. The evaluations of each stage of the pipeline are elaborated on below. For this set of experiments, the same evaluation metrics explained in Section 5.2 are utilized.

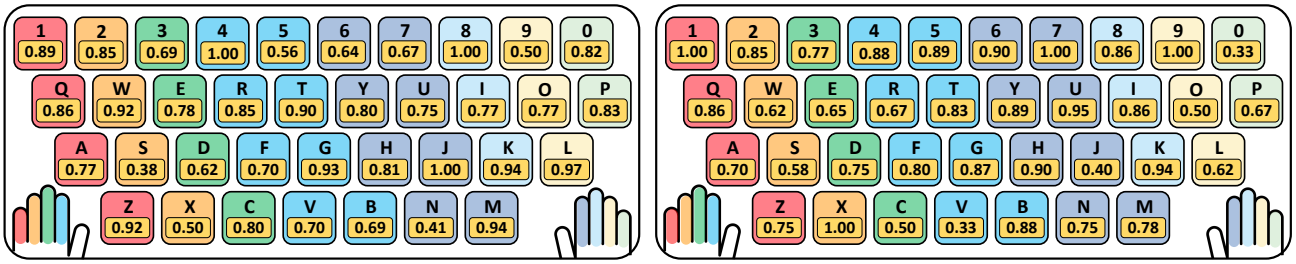
5.3.1 Keystroke Detection. The performance of the *Keystroke Detection* for the practical instance of SIA was almost identical with the results provided earlier in Table 5— we achieved about 0.98 in terms of average TPR. Therefore, we did not include another table for the *Keystroke Detection* evaluations for the lack of space.

5.3.2 Key Identification & Word Correction. Figure 8 demonstrates the character-wise TPR for the practical attack. When we compare the results in Figure 7 and Figure 8, we observe a slight decrease in the overall TPRs for the BMK (0.798 → 0.764) and a relatively large decrease for the MBA (0.937 → 0.776). The difference between the unique typing styles of the subjects leads to a decrease in the detection performance. Table 11 shows the cross-entropy decreases through the practical attack. Although the decrease is not as much as in Table 7, we are able to significantly reduce the difference against the actual probability distribution. The reduced difference can be utilized to further reduce the search space.

Table 10 shows the string-wise evaluations for each dataset—after *Key Identification*—and each word correction method for the E-mail dataset. When comparing Table 10 with Table 8 and Table 9, we can observe a decrease in the TPR; i.e., as observed in the character-wise TPRs. However, the difference between the tables is reduced when the one-hop variances are considered. This shows that, in the practical attack, most of the mispredicted characters are within the one-hop distance of the correct character. This observation gives some hints about the mispredicted characters, which further reduces the search space.

Table 10: String-wise evaluation of *key identification* and *word correction* (practical attack). The predictions obtained from identification (Random(R)/Selected(S)/E-mail(E)) and from correction (SSC/MT/NW) are compared with the ground truth.

Device	BMK						MBA					
Method	**R**	**S**	**E**	**SSC**	**MT**	**NW**	**R**	**S**	**E**	**SSC**	**MT**	**NW**
TPR	0.769	0.762	0.761	0.848	0.709	0.734	0.792	0.758	0.680	0.790	0.709	0.734
TPR (1-hop)	0.921	0.933	0.930	0.959	0.883	0.878	0.949	0.912	0.877	0.918	0.883	0.878
NLD	0.217	0.215	0.192	0.122	0.234	0.214	0.195	0.210	0.258	0.169	0.234	0.214
NLD (1-hop)	0.073	0.060	0.056	0.032	0.093	0.098	0.047	0.070	0.098	0.066	0.093	0.098



(a) The character-wise TPR for the MBA keyboard. **(b) The character-wise TPR for the BMK keyboard.**

Figure 8: The character-wise evaluation results for the practical attack.

Table 11: The cross-entropy changes for the practical attack.

Device	Dataset	Ideal	Frequency	Ours
MBA	E-mail	ϵ	14.02	1.19
	Random	ϵ	3.58	1.29
	Selected	ϵ	18.98	1.09
BMK	E-mail	ϵ	14.02	1.29
	Random	ϵ	3.58	1.12
	Selected	ϵ	18.98	1.18

6 LIMITATIONS AND FUTURE WORK

Despite the very promising results, which show the successful execution of SIA, this research is subject to several limitations which we outline in the following. First, due to COVID-19 restrictions, this study had to be performed on a limited number of test subjects. In the future, we are going to perform the experiments on multiple subjects to show the applicability of the attack at scale—guided by the robustness of the features employed, and the subject generalization results provided in this study, we expect that the results will hold on a large number of subjects. Second, although we demonstrated the performance difference across different widely used keyboards, the variety of the recording devices and keyboard models are limited. In our future work, we will utilize different smartwatch and keyboard models, specifically the ones that are widely used. Third, similar to the previous studies, the data is collected in a relatively quiet environment (i.e., in a room with a wall adjacent to a public street), when compared to a recording while actually on the street, or at a café. Therefore, the effect of excessive background noise is not considered in this study, which will be explored in the future. Finally, although the same type of data is used for both of the attack scenarios, the cellular network characteristics (and imperfections) are not considered in this work. In our future work, we will examine the effect of the potential cellular network problems on the recordings, including connection problems, latency, hot-cuts, etc. While anticipating that those issues will minimally affect the underlying principles of the attack, and the attack will still hold nevertheless, quantifying that effect is an open direction.

7 CONCLUSION

In this work, we demonstrated a keylogging attack framework through the acoustic emanations captured by a smartwatch—SIA. We proposed a system and threat model supported by two plausible attack scenarios. The threat model leverages the smartwatch microphone as a recorder and collects data of the acoustic emanations of a physical keyboard. By neutralizing the effect of background noises, which vary depending on the environmental settings, we lay a base for a robust identification framework. We then utilize digital signal processing techniques to locate keystroke events in data and extracted the most descriptive feature among the alternatives; MFCC. MFCC features are then scaled and forwarded to the best performing learning technique (SVM) for identification. Finally, we further increased our prediction accuracy using various state-of-the-art NLP techniques. We performed two types of experiments: user profiling and practical attack. With user profiling, we are able to recover up to 98% of the typed text. For the practical attack, we are able to recover up to 85% of the typed text.

We conducted our experiments on popular devices and verified the potential privacy leakage. We believe the high accuracy and the practicality of such an attack should be yet an alarming tale to smartwatch users regarding the privacy issues introduced with the integration of new technologies into our daily lives.

Acknowledgement. This work was supported by NRF grant number 2016K1A1A2912757, a seed grant from CyberFlorida, and a GPU grant from NVIDIA.

REFERENCES

[1] 2020. Smartwatch Market - Growth, Trends, Forecasts (2020 - 2025). https://www.researchandmarkets.com/reports/4591978/smartwatch-market-growth-trends-forecasts

[2] Manal Al-Sharrah, Ayed Salman, and Imtiaz Ahmad. 2018. Watch Your Smartwatch. In *2018 International Conference on Computing Sciences and Engineering (ICCSE)*. 1–5. https://doi.org/10.1109/ICCSE1.2018.8374228

[3] Kamran Ali, Alex X. Liu, Wei Wang, and Muhammad Shahzad. 2015. Keystroke Recognition Using WiFi Signals. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking* (Paris, France) *(MobiCom '15)*. Association for Computing Machinery, New York, NY, USA, 90–102. https://doi.org/10.1145/2789168.2790109

[4] D. Asonov and R. Agrawal. 2004. Keyboard acoustic emanations. In *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004.* 3–11. https://doi.org/10.1109/SECPRI.2004.1301311

[5] Ibrahim Baggili, Jeff Oduro, Kyle Anthony, Frank Breitinger, and Glenn McGee. 2015. Watch What You Wear: Preliminary Forensic Analysis of Smart Watches. In *2015 10th International Conference on Availability, Reliability and Security*. 303–311. https://doi.org/10.1109/ARES.2015.39

[6] D. Balzarotti, M. Cova, and G. Vigna. 2008. ClearShot: Eavesdropping on Keyboard Input from Video. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*. 170–183.

[7] Bianco D. Barisani A. 2009. Sniffing Keystrokes With Lasers and Voltmeters. In *In Proceedings of Black Hat USA*.

[8] Vincent Becker, Linus Fessler, and Gábor Sörös. 2019. GestEar: Combining Audio and Motion Sensing for Gesture Recognition on Smartwatches. In *Proceedings of the 23rd International Symposium on Wearable Computers* (London, United Kingdom) *(ISWC '19)*. Association for Computing Machinery, New York, NY, USA, 10–19. https://doi.org/10.1145/3341163.3347735

[9] Yigael Berger, Avishai Wool, and Arie Yeredor. 2006. Dictionary Attacks Using Keyboard Acoustic Emanations. In *Proceedings of the 13th ACM Conference on Computer and Communications Security* (Alexandria, Virginia, USA) *(CCS '06)*. Association for Computing Machinery, New York, NY, USA, 245–254. https://doi.org/10.1145/1180405.1180436

[10] Bo Chen, Vivek Yenamandra, and Kannan Srinivasan. 2015. Tracking Keystrokes Using Wireless Signals. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services* (Florence, Italy) *(MobiSys '15)*. Association for Computing Machinery, New York, NY, USA, 31–44. https://doi.org/10.1145/2742647.2742673

[11] Alberto Compagno, Mauro Conti, Daniele Lain, and Gene Tsudik. 2017. Don't Skype & Type! Acoustic Eavesdropping in Voice-Over-IP. In *Proceedings ACM on Asia Conference on Computer and Communications Security* (Abu Dhabi, United Arab Emirates). ACM, New York, NY, USA, 703–715. https://doi.org/10.1145/3052973.3053005

[12] Stanford Center for Digital Health and Rock Health. 2019. https://rockhealth.com/reports/digital-health-consumer-adoption-report-2019/

[13] Jeffrey Friedman. 1972. Tempest: A signal problem. *NSA Cryptologic Spectrum* 35 (1972), 76.

[14] Jun Gong, Zheer Xu, Qifan Guo, Teddy Seyed, Xiang 'Anthony' Chen, Xiaojun Bi, and Xing-Dong Yang. 2018. *WristText: One-Handed Text Entry on Smartwatch Using Wrist Gestures.* Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, New York, NY, USA, 1–14. https://doi.org/10.1145/3173574.3173755

[15] M. Hagiwara. 2021. *Real-World Natural Language Processing: Practical applications with deep learning*. Manning Publications. https://books.google.com.tr/books?id=A92_zQEACAAJ

[16] Tzipora Halevi and Nitesh Saxena. 2014. Keyboard acoustic side channel attacks: exploring realistic and security-sensitive scenarios. *International Journal of Information Security* 14 (09 2014), 1–14. https://doi.org/10.1007/s10207-014-0264-7

[17] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014). arXiv:1412.6980 http://arxiv.org/abs/1412.6980

[18] Bryan Klimt and Yiming Yang. 2004. The Enron Corpus: A New Dataset for Email Classification Research. In *Proceedings of the 15th European Conference on Machine Learning* (Pisa, Italy) *(ECML'04)*. Springer-Verlag, Berlin, Heidelberg, 217–226. https://doi.org/10.1007/978-3-540-30115-8_22

[19] Gierad Laput and Chris Harrison. 2019. Sensing Fine-Grained Hand Activity with Smartwatches. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3290605.3300568

[20] Ho-Man Colman Leung, Chi-Wing Fu, and Pheng-Ann Heng. 2018. TwistIn: Tangible Authentication of Smart Devices via Motion Co-Analysis with a Smartwatch. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 2, Article 72 (July 2018), 24 pages. https://doi.org/10.1145/3214275

[21] Vladimir I Levenshtein. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady* 10 (February 1966), 707.

[22] S. Li, A. Ashok, Y. Zhang, C. Xu, J. Lindqvist, and M. Gruteser. 2016. Whose move is it anyway? Authenticating smart wearable devices using unique head movement patterns. In *2016 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. 1–9. https://doi.org/10.1109/PERCOM.2016.7456514

[23] Xiangyu Liu, Zhe Zhou, Wenrui Diao, Zhou Li, and Kehuan Zhang. 2015. When Good Becomes Evil: Keystroke Inference with Smartwatch. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security* (Denver, Colorado, USA) *(CCS '15)*. Association for Computing Machinery, New York, NY, USA, 1273–1285. https://doi.org/10.1145/2810103.2813668

[24] Chris Xiaoxuan Lu, Bowen Du, Peijun Zhao, Hongkai Wen, Yiran Shen, Andrew Markham, and Niki Trigoni. 2018. Deepauth: In-Situ Authentication for Smartwatches via Deeply Learned Behavioural Biometrics. In *Proceedings of the 2018 ACM International Symposium on Wearable Computers* (Singapore, Singapore) *(ISWC '18)*. Proceedings of the 2018 ACM International Symposium on Wearable Computers, New York, NY, USA, 204–207. https://doi.org/10.1145/3267242.3267252

[25] Anindya Maiti, Oscar Armbruster, Murtuza Jadliwala, and Jibo He. 2016. Smartwatch-Based Keystroke Inference Attacks and Context-Aware Protection Mechanisms. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security* (China) *(ASIA CCS '16)*. Association for Computing Machinery, New York, NY, USA, 795–806. https://doi.org/10.1145/2897845.2897905

[26] Philip Marquardt, Arunabh Verma, Henry Carter, and Patrick Traynor. 2011. (Sp)IPhone: Decoding Vibrations from Nearby Keyboards Using Mobile Phone Accelerometers. In *18th ACM Conference on Computer and Communications Security* (Chicago, Illinois, USA) *(CCS '11)*. ACM, New York, NY, USA, 551–562. https://doi.org/10.1145/2046707.2046771

[27] Ülkü Meteriz, Necip Fazıl Yıldıran, Joongheon Kim, and David Mohaisen. 2020. Understanding the Potential Risks of Sharing Elevation Information on Fitness Applications. In *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*. 464–473. https://doi.org/10.1109/ICDCS47774.2020.00063

[28] J. V. Monaco. 2018. SoK: Keylogging Side Channels. In *2018 IEEE Symposium on Security and Privacy (SP)*. 211–228.

[29] Peter Norvig. 2007. https://norvig.com/spell-correct.html

[30] Nicole Odom, Jesse Lindmar, John Hirt, and Josh Brunty. 2019. Forensic Inspection of Sensitive User Data and Artifacts from Smartwatch Wearable Devices. *Journal of Forensic Sciences* 64 (06 2019). https://doi.org/10.1111/1556-4029.14109

[31] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. (2019).

[32] Mohd Sabra, Anindya Maiti, and Murtuza Jadliwala. 2020. Zoom on the Keystrokes: Exploiting Video Calls for Keystroke Inference Attacks. arXiv:2010.12078 [cs.CR]

[33] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15, 56 (2014), 1929–1958. http://jmlr.org/papers/v15/srivastava14a.html

[34] T. Sztyler and H. Stuckenschmidt. 2017. Online personalization of cross-subjects based activity recognition models on wearable devices. In *2017 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. 180–189. https://doi.org/10.1109/PERCOM.2017.7917864

[35] Aku Visuri, Zhanna Sarsenbayeva, Niels van Berkel, Jorge Goncalves, Reza Rawassizadeh, Vassilis Kostakos, and Denzil Ferreira. 2017. *Quantifying Sources and Types of Smartwatch Usage Sessions.* Association for Computing Machinery, New York, NY, USA, 3569–3581. https://doi.org/10.1145/3025453.3025817

[36] Tran Huy Vu, Archan Misra, Quentin Roy, Kenny Choo Tsu Wei, and Youngki Lee. 2018. Smartwatch-Based Early Gesture Detection 8 Trajectory Tracking for Interactive Gesture-Driven Applications. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 1, Article 39 (March 2018), 27 pages. https://doi.org/10.1145/3191771

[37] Martin Vuagnoux and Sylvain Pasini. 2009. Compromising Electromagnetic Emanations of Wired and Wireless Keyboards. *USENIX Security Symposium* (01 2009).

[38] Chen Wang, Xiaonan Guo, Yan Wang, Yingying Chen, and Bo Liu. 2016. Friend or Foe?: Your Wearable Devices Reveal Your Personal PIN. 189–200. https://doi.org/10.1145/2897845.2897847

[39] He Wang, Ted Tsung-Te Lai, and Romit Roy Choudhury. 2015. MoLe: Motion Leaks through Smartwatch Sensors. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking* (Paris, France) *(MobiCom '15)*. Association for Computing Machinery, New York, NY, USA, 155–166. https://doi.org/10.1145/2789168.2790121

[40] X. Yu, Z. Zhou, M. Xu, X. You, and X. Li. 2020. ThumbUp: Identification and Authentication by Smartwatch using Simple Hand Gestures. In *2020 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. 1–10. https://doi.org/10.1109/PerCom45495.2020.9127367

[41] Li Zhuang, Feng Zhou, and J. D. Tygar. 2009. Keyboard Acoustic Emanations Revisited. *ACM Trans. Inf. Syst. Secur.* 13, 1, Article 3 (Nov. 2009), 26 pages. https://doi.org/10.1145/1609956.1609959