EXPLORING THE PRIVACY DIMENSION OF WEARABLES THROUGH MACHINE LEARNING-ENABLED INFERENCE

by

ÜLKÜ METERIZ YILDIRAN
M.Sc. University of Central Florida, United States, 2020
B.Sc. Middle East Technical University, Turkey, 2018

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida,
Orlando, Florida

Spring Term
2022

Major Professor: David Mohaisen

ABSTRACT

Today's hyper-connected consumers demand convenient ways to tune into information without switching between devices, which led the industry leaders to the *wearables*. Wearables such as smartwatches, fitness trackers, and augmented reality (AR) glasses can be comfortably worn on the body. In addition, they offer limitless features, including activity tracking, authentication, navigation, and entertainment. Wearables that provide digestible information stimulate even higher consumer demand. However, to keep up with the ever-growing user expectations, developers keep adding new features and interaction methods to augment the use cases without considering their privacy impacts. In this dissertation, we explore the privacy dimension of wearables through inference attacks facilitated by machine learning approaches.

We start our investigation by exploring the attack surface introduced by fitness trackers. We propose an inference attack that breaches location privacy through the elevation profiles collected by fitness trackers. Our attack highlights that adversaries can infer the location from elevation profiles collected via fitness trackers.

Second, we investigated the attack surface introduced by the smartwatches. We introduce an inference attack that exploits the smartwatch microphone to capture the acoustic emanations of physical keyboards and successfully infers what the user has been typing. With this attack, we showed that smartwatches add yet another privacy dimension to be considered.

Third, we examined the privacy of AR domain. We designed an inference attack exploiting the geometric projection of hand movements in air. The attack framework predicts the typed text on an in-air tapping keyboard, which is only visible to the user.

Our studies uncover various attack surfaces introduced by wearables that have not been studied in literature before. For each attack, we propose possible countermeasures to diminish the ramifications of the risks. We hope that our findings shed light to the privacy risks of wearables and guide the research community to more aware solutions.

To my better half.

ACKNOWLEDGMENTS

First and foremost, I am extremely grateful to my supervisor, Prof. David Mohaisen, for his invaluable advice, continuous support, and patience during my PhD study. David helped me throughout my academic research and daily life with his immense knowledge and experience. He always supported my ideas and encouraged me to pursue my own research interests and passions. He picked me up when I felt lost, and helped me to get back on track.

I would like to extend my sincere thanks to my doctoral dissertation committee members, Prof. Damla Turgut, Prof. Murat Yüksel, Prof. Cliff Zou, and Prof. Mary Jean Amon for their valuable feedback at every step of the doctoral milestones.

I am also grateful to my friends and collaborators in the Security and Analytics Lab (SEAL) over the years: Afsah, Ahmed, Amin, Ayo, Hisham, Jabbar, Jeman, Jinchun, Marwan, Mo Abuhamad, Mohammad, Necip, Ran, Rhongho, Saad, Soohyeon, and Sultan.

I would like to express my gratitude to my parents and my brother. Without their tremendous support and encouragement in the past four years, it would have been impossible for me to complete my dissertation.

Last but not least, I cannot begin to express my thanks to my best friend and husband, who stood by me through all the ups and downs, helped me to stay strong, always supported me, and shared my journey. You are my rock.

TABLE OF CONTENTS

LIST OF FIGURES viii

LIST OF TABLES xv

CHAPTER 1: INTRODUCTION 1

Statement of Research 2

CHAPTER 2: RELATED WORK 6

Location Breach 6

Keylogging 8

CHAPTER 3: EXPLOITING SHARED ELEVATION PROFILES FOR PRIVACY INFERENCE THROUGH REPRESENTATION 13

Background 14

Threat Models 17

Approach: High-Level Overview 18

Implementation Details 20

Evaluation, Results, and Discussion 33

Countermeasures 43

Summary of the Completed Work 44

CHAPTER 4: EXPLOITING ACOUSTICS FOR PRIVACY INFERENCE THROUGH MODELLING 46

System and Threat Model 47
Methodology 49
Evaluation 60
Countermeasures 74
Summary of the Completed Work 76

CHAPTER 5: EXPLOITING GEOMETRIC PROJECTIONS FOR PRIVACY INFERENCE THROUGH MAPPINGS 78
System and Threat Models 79
AiRType: AR Keyboard Design 83
Approach Overview 90
Technical Details and Methods 94
Evaluation 103
Countermeasures 114
Summary of the Completed Work 117

CHAPTER 6: CONCLUSION AND FUTURE WORK 118

APPENDIX A: COPYRIGHT INFORMATION 120

APPENDIX B: IRB LETTERS 134

LIST OF REFERENCES 137

LIST OF FIGURES

3.1 Survey results for understanding users behavior with starting point statistics and finishing point statistics. While 90% of the 60 participants indicated their start of activity is either home, school, or work, an overwhelming 98% of the participant indicated those to be the finish (end) point of their activities. . . . 14

3.2 The end-to-end pipeline of the approach consisting of four main stages: (i) data collection, (ii) preprocessing, (iii) feature extraction, and (iv) classification. There are two types of data representations, each of which is processed differently in the feature extraction and classification stages. The feature extraction of the image-like representation is internally handled in the convolution layers of the classification phase. The convolution illustration in the feature extraction step is for the sake of modularization and for consistency with the other pipeline instance. 19

3.3 An illustration of the data mining pipeline. A geolocation boundary, B, is segmented into small boundaries, each of which is then forwarded to the Segment Exploration step to obtain the most frequented 10 route segments for that particular boundary. Finally, the elevation profile of each route segment is retrieved from the Google Maps API. 22

3.4 An illustration of the tight rectangle encapsulating an example route. The rectangle is created by fitting the route in between the minimum and maximum $(latitude, longitude)$ pairs of the given route. The minimum and maximum $(latitude, longitude)$ pairs correspond to the bottom left (*i.e.* South West) corner and the top right (*i.e.* North East) corner of the rectangle respectively. 23

3.5 Illustration of the flow of text-like preprocessing. The signal is discretized by eliminating the small elevation fluctuations. The discretized signal is also used for deciding the word size of the encoding. The discrete signal is then encoded in text and a vocabulary is built. 27

3.6 Illustration of bi-gram creation where the word size is $w = 2$ and window size is $W = 4$. 27

3.7 Elevation profile graphs by fixing the *y*-axis range and using only black versus elevation profile graphs by fitting and using color encoding. As can be seen, with the black option the elevation range can be represented by the position of the signal, but changes in the signal are not visible, which is an information loss. However, with the color option enabled, both alterations in the signal and the signal range are represented in one image. 28

3.8 The architecture of the LSTM network consisting of an LSTM unit with four hidden layers and two fully-connected layers. The input samples are passed through the LSTM unit individually and the last hidden state of the LSTM unit is forwarded to the fully-connected neural network. The fully-connected neural network utilizes softmax activation at the output layer which outputs the class probabilities. 33

3.9 The CNN architectures used for classification. A 1-D CNN is used for the one-dimensional data representations, i.e., n-grams, tf-idf, and raw data. A 2-D CNN is used for the image-like representation. The numbers in the figures depict the dimensions of the input throughout the learning/prediction process. 34

3.10 Some selected simulation results of TM-2 and TM-3. The maximum achieved accuracy results are compared. 39

3.11 An illustration of round creation from an unbalanced dataset of three classes. 41

3.12 An illustration of the fine-tuning pipeline for an unbalanced dataset of three classes. 41

4.1 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. 48

4.2 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. 49

4.3 The significant frequency band overlap between the white noise and the acoustics of a keystroke event. 50

4.4 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. 52

4.5 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. 55

4.6 The cumulative character-wise distribution of the test datasets. The colors on the keys encodes the associated hand and finger. 61

4.7 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. 66

4.8 The character-wise evaluation results for the practical attack. 72

5.1 First person view of the AR keyboard and hand meshes operated by the user from the Magic Leap 1. 79

5.2 Illustrations of the attack scenarios with varying adversarial capabilities. . . . 81

5.3 Illustrations of possible positioning options of AiRType in the augmented reality environment. 84

5.4 The position and orientation of coordinate systems of Magic Leap 1 (left) and Leap Motion Controller (right) is different. 85

5.5 The system keyboard of Magic Leap 1, i.e., baseline keyboard. The targeting is done through moving the cursor via the ray coming out of the remote controller. The selection is done by pulling the trigger button at the back of the remote. 87

5.6 The five-step pipeline for the attack, which takes hand tracking data as input and outputs inferences ordered from best to worst. The first two steps are for keystroke detection stage while the subsequent three are for key identification stage. The attack starts with the pre-processing of the data coming from the hand tracker device, and localization of the key tap windows into time slices in the input signal at *Deep Key Tap Localization*. Then, the key taps are refined through a set of processes done with the help of an estimation of the keyboard plane at *Key Tap Localization Refinement*, outputting 2D key tap points. At the next step, *Candidate Key Center Generation* deduces the set of candidate key centroids through clustering of the key tap points. *Candidate Keyboard Reconstruction* composes a set of candidate keyboard reconstructions through computing reconstruction transformation using the candidate key centers. Finally, *Best to Worst Ordering* evaluates, filters and orders the candidate keyboard reconstructions to build a best to worst ordering of the reconstructions through a set of inference measurements. 90

5.7 The multi-head CNN architecture used for key tap localization. The input is a segment from the pre-processed low-level hand tracking data, while the output is the confidence score for the segment belonging to a key tap. 95

5.8 Refining through the estimated keyboard plane and the trace of the index fingertip. The dots and the curve connecting the dots represents the trace of the index fingertip. The green window is considered as true positive since the window encapsulates a trace crosses the estimated plane. The associated key tap point is refined as the green circle instead of the red circle. The red window is considered as false positive and eliminated since the encapsulated trace does not cross the plane. 97

5.9 The demonstration of the maximum and the minimum scaling factors ($\mathbf{IM_1}$) used to filter the candidate keyboard reconstructions. For concise visualization, the effect of the scaling factor T_s is shown by inversely scaling the hand model rather than scaling the keyboard itself. In the virtual environment, however, the keyboard model gets bigger as the scaling factor T_s grows while the size of the hand model is fixed. 102

5.10 The setup used to carry out the experiments. 104

5.11 The precision-recall curve of the multi-head CNN in *Deep Key Tap Localization*. Different networks are trained for each hand, and both of them perform similarly well. 107

5.12 The effect of the *Key Tap Localization Refinement* on the number of true positive (TP), false positive (FP), and false negative (FN) samples for different users. 108

5.13 The accuracy distribution of the E-mail keyboard reconstructions before and after best to worst ordering. "All" corresponds to reconstructions after $\mathbf{IM_1}$ filtering. Top-15 ordering and all reconstructions after $\mathbf{IM_1}$ filtering categorized by their h_0, h_1, and h_2 accuracy quarters. 109

5.14 The top-k h_0, h_1, and h_2 accuracy for varying k considerations for each data set obtained by utilizing the key identification steps on the ground truth key tap points. 110

5.15 The inference found in the ordered list which is obtained utilizing the end-to-end pipeline on the E-mail. The end-to-end pipeline inference results on the first three sentences of the E-mail, which is found at the 11^{th} place in the ordered list, which were obtained by utilizing the end-to-end pipeline. The spaces are inserted for clearer visualization. Although the experiment did not include the usage of space key, spaces are inserted in the figure for clearer visualization of the difference. 113

5.16 With the dynamically positioned keyboard defense, the probability for each key that the adversary's guess on the key is a true positive. Outward keys have more chance to be correctly guessed since they have fewer target positions after displacement that conflict with the other keys. 115

LIST OF TABLES

1.1 Overview of the inference attacks. 2

2.1 A summary of the related keylogging inference attacks. 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. Top-X word means that the study matches words instead of keys and top-X PIN means that the study infers only the numerical characters. 8

3.1 Popular fitness applications and their features. ET: Exercise tracking. SS: Ability to share to social media. SNS: Social networking capabilities in the service. PR: Private records. BU: User blocking capability. 15

3.2 User-specific dataset sample size distribution. 24

3.3 City-level dataset sample size distribution. 24

3.4 Borough-level dataset sample size distribution. 25

3.5 The overall evaluation results for TM-1. Accuracy (%) with different data representations and classification models. In this table, the following abbreviations are used: SVM: Support Vector Machine; RF: Random Forest; MLP: Multi-Layer Perceptron; C1D: 1D Convolution; LSTM: Long Short-Term Memory. **C** column indicates the number of classes in the classification problem. The following settings are used: 4-class = [WDC, ORL, NYC, SD], 3-class = [WDC, ORL, NYC], 2-class = [WDC, ORL]. 35

3.6 The overall evaluation results for TM-2. Accuracy (%) with different data representations and classification models. In the table, we use the following abbreviations: LA: Los Angeles; MIA: Miami; NJ: New Jersey; NYC: New York City; SF: San Francisco; WDC: Washington, D.C. 36

3.7 The overall evaluation results for TM-3. Accuracy (%) with different data representations and classification models. 38

3.8 Comparison of maximum achieved accuracy across different methods. The Unweighted Loss (UWL) column is not considered while deciding the maximum accuracy, as the results are biased. The maximum accuracy of each evaluation is written **bold**, the results that are not considered are written *italic*. 42

3.9 The fine-tuning results for TM-1 and TM-3 as the epoch size changes. 42

3.10 The fine-tuning results for TM-2 as the epoch size is 1000 and learning rate is 0.001 for all rounds. 43

4.1 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. 55

4.2 Comparison of different multi-class classifiers with MFCC features in terms of TPR. Support Vector Machine (SVM) is the best classifier alternative. . . . 57

4.3 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. 62

4.4 Keystroke detection results where 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 (Samsung Galaxy Watch Active 2 / Apple Watch Series 6 / Samsung Galaxy s10e), and target keyboard (MBA / BMK) combinations. 63

4.5 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. 67

4.6 Cross entropy shows the difference between two probability distribution. **Ideal** column is the cross entropy of the actual distribution (one-hot vector) with itself ($\epsilon = -1 \times 10^{-9}$). **Frequency** column is the cross entropy between the actual distribution and the frequency distribution associated with each dataset (E-mail $\rightarrow$ English letter frequency, Random $\rightarrow$ Uniform distribution, Selected $\rightarrow$ Character distribution of RockYou leaked passwords) SIA column shows the cross entropy between the actual distribution and the distribution estimated by our method. 68

4.7 String-wise evaluation results of Key Identification for all device-keyboard-dataset combinations. The predictions obtained from Key Identification stage are compared with the ground truth strings. 70

4.8 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. 71

4.9 The cross-entropy changes for the practical attack. 73

4.10 String-wise evaluation results of Key Identification for all keyboard-dataset combinations. The predictions obtained from Key Identification stage of the practical attack are compared with the ground truth strings. 74

4.11 String-wise evaluation results of Word Correction on E-mail dataset for all keyboard-method combinations for practical attack. The output of Word Correction stage is compared with the ground truth strings. 75

5.1 The average ISO usability test results obtained from 5 participants per group, 10 participants in total. 89

5.2 The maximum h_0, h_1, and h_2 accuracy achieved for each data set across varying top-k orderings and brute-force attack. 111

CHAPTER 1: INTRODUCTION

Wearable devices 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., healthcare support, health and fitness monitoring, navigation, voice commands, activity recognition [71, 114], gesture recognition [37, 56, 120, 130], and authentication [73, 77, 83, 130]. According to a recent survey by Rock Health [50], the adoption of wearable devices increased from 24% in 2018 to 33% in 2019. Per another report [23], 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 [119] that quantifies smartwatches usages suggests that smartwatches are nowadays used more frequently than smartphones.

After decades of research and development [32, 33, 43, 53, 64], augmented reality (AR) technologies are now available for consumers, offering immersive experiences with a blend of virtual and real-world contents. Having got the attention of users via smartphone AR [115, 125], the development of more sophisticated AR technologies, such as head-mounted displays (HMDs), healthcare, e-commerce, and navigation systems [42, 92], and automotive AR windshields [47, 116], are also gaining speed. The applications offered within the recently available AR HMD devices, such as Magic Leap 1 [19] and Microsoft HoloLens [20], show the diverse use cases of these technologies, including simulation, entertainment, training, and assistance. With AR's involvement in response to the needs that emerged from the recent pandemic COVID-19 outbreak [16, 46, 63, 89], the benefits of the AR technology are even more evident. AR HMD technologies are expected to be an asset for daylong use [76, 103].

Despite the limitless benefits that wearables offer to consumers, security and privacy issues associated with them restraint their rising projection. Especially the new interaction methods with those devices, such as voice-enabled features of smartwatches or gesture-enabled features of AR HMDs,

Table 1.1: Overview of the inference attacks.

C	Device/Context	Feature	Technique	Outcome	Defense
C-3	Fitness Tracker	Elevation Profiles	Representation	Location Breach	Aggregation
C-4	Smartwatch	Acoustics	Modeling	Keylogging	Noising
C-5	AR HMD	Geometric Projections	Mapping	Keylogging	Counter-projection

brings about lots of controversial aspects of these interaction methods.

From smartphones to wearables, an increasing number of Internet of Things (IoT) devices are also equipped with Global Positioning System (GPS), accelerometers, and gyroscopes to allow applications to function or to present a better user experience using *geodata*, such as location and elevation information. More recently, fitness applications that run on wearable trackers and smartwatches used these components to collect spatial, temporal, and activity-specific information to analyze, summarize, and visualize users' activities. By analyzing those activities, many of those applications deliver personalized motivations and challenges for users to meet their goals.

Despite the broad set of advantages of integrating geodata with wearables, geodata usage and uncontrolled sharing can pose a significant privacy risk that can be further exploited in multiple attacks, including stalking [101] and cybercasing [51].

To keep up with the fast-growing nature of wearables, manufacturers and developers turned a deaf ear to associated security and privacy risks. Motivated by the deficiency of the security and privacy considerations, we find it timely and significant to investigate the potential security and privacy risks introduced by the wearables.

Statement of Research

In this dissertation, we take a step towards to exploring the security and privacy risks introduced by wearables through various inference attacks. Table 1.1 summarizes the device, feature, technique, outcome, and defense of each inference attack. We further elaborate the attacks below.

Exploiting Elevation Profiles for Privacy Inference Through Representation (Chapter 3). The extensive use of smartphones and wearable devices has facilitated many useful applications. For example, with Global Positioning System (GPS)-equipped smart and wearable devices, many applications can gather, process, and share rich metadata, such as geolocation, trajectories, elevation, and time. Fitness applications, such as Runkeeper and Strava, utilize location information for activity tracking and have recently witnessed a boom in popularity. Those fitness tracker applications have their own web platforms, and allow users to share activities on such platforms, or even with other social network platforms. To preserve the privacy of users while allowing sharing, several of those platforms may allow users to disclose partial information, such as the elevation profile for an activity, which supposedly would not leak the location of the users. In this work, and as a cautionary tale, we create a proof of concept where we examine the extent to which elevation profiles can be used to predict the location of users. To tackle this problem, we devise three plausible threat settings under which the city or borough of the targets can be predicted. Those threat settings define the amount of information available to the adversary to launch the prediction attacks. Establishing that simple features of elevation profiles, e.g., spectral features, are insufficient, we devise both natural language processing (NLP)-inspired text-like representation and computer vision-inspired image-like representation of elevation profiles, and we convert the problem at hand into text and image classification problem. We use both traditional machine learning- and deep learning-based techniques and achieve a prediction success rate ranging from 59.59% to 95.83%. The findings are alarming, and highlight that sharing elevation information may have significant location privacy risks.

Exploiting Acoustics for Privacy Inference Through Modelling (Chapter 4). The convergence of various technologies, such as smartwatches, smartphones, etc. has proven to be beneficial, although poses various security and privacy risks. In this work, 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.

Exploiting Geometric Projections for Privacy Inference Through Mappings (Chapter 5). Enabling users to push the physical world's limits, augmented and virtual reality platforms opened a new chapter in perception. Novel immersive experiences resulted in the emergence of new interaction methods for virtual environments, which came with unprecedented security and privacy risks. This chapter presents a spatial keylogging inference attack to infer user inputs typed with in-air tapping keyboards. We observe that hands follow specific patterns when typing in the air and exploit this observation to carry out our attack. Starting with three plausible attack scenarios where the adversary can obtain the hand trace patterns of the victim, we build a pipeline to reconstruct the user input. Our attack pipeline takes the hand traces of the victim as an input and outputs a set of input inferences ordered from the best to worst. Through various experiments, we showed that our inference attack achieves a pinpoint accuracy ranging from 40% to 87% within at most the top-500 candidate reconstructions. Finally, we discuss possible countermeasures, while the results presented provide a cautionary tale of the potential security and privacy risk of the immersive mobile technology.

Organization. This dissertation is organized as follows: We visit the literature and outline related work in Chapter 2. In Chapter 3, we examine the location privacy dimension of fitness trackers through the elevation profiles they collect. In Chapter 4, we study the attack surface introduced by the microphones in smartwatches. In Chapter 5, we explore the input privacy dimension of

in-air tapping keyboards in AR HMDs. Chapter 6 summarizes the main points outlined in this dissertation along with the future research directions.

CHAPTER 2: RELATED WORK

In this chapter, we categorize the literature based on the outcome as outlined in Table 1.1: Location Breach and Keylogging. In the first section, we narrate the related studies in the location privacy domain. In the second section, we summarize the keylogging attacks in the literature that exploits numerous mediums.

Location Breach

In this dissertation, we address the problem of *location privacy* in activity trackers using publicly shared elevation profiles. While, to the best of our knowledge, there is no work that investigates the exact research question, there are various related studies in the broad literature.

Most location privacy breaches are caused since users do not know why or how to preserve location privacy. Aktypi *et al.* [27] developed a tool to examine possible privacy exposures of users in their social networks where the data is mostly collected from wearable devices. Using this tool, the authors aimed to enhance the awareness of information leakage in social networks, particularly fitness applications in which the data retrieved from wearable devices is shared on social networks. Abdelmoty and Alrayes [26] aimed to increase awareness of location privacy on geo-social networks by surveying 186 users, where 77% of them indicated they use location-based services often, several times a day, and 47% of them reported that they were not aware that the location-based apps collect and store location information even when users select the private location option. Moreover, 43% of respondents were not aware that applications may share location information with third parties.

Despite the methods employed to preserve location privacy, several attacks are devised to uncover supposedly protected locations. Experiments for revealing exact locations from trajectories with private zones are conducted on a fitness tracking social network, Strava [60]. Researchers found the exact endpoints associated with users, even when such users selected the private zone option

when sharing the training route. In another study, location trajectories of users are recovered from publicly available aggregated mobility data obtained from GSM operators [117]. The attack relies on tracking the regularity—i.e., coming across the same location trace in the aggregated data regularly—and uniqueness—i.e., the location trace belongs to a unique user—of the user mobility traces to recover trajectories.

As our study exemplifies, online social networks amplify the scope of privacy breach risks for users. Zheng *et al.* [132] shows that sharing data that reveals spatiotemporal features of users' mobility patterns on online social networks reveals sensitive information such as home location, using a different form of data, *i.e.* multimedia. Rossi *et al.* [105] show that location-based social networks are vulnerable to identity privacy breaches by revealing the identity of users by observing their mobility patterns.

Several attacks against general location privacy methods are proposed [124]. The homogeneity attack [84] is an attack on k-anonymity to infer data of interest from other shared data. Machanavajjhala *et al.* [84] illustrated a scenario where an adversary infers the illness of a target person from available information, the zip code, age, etc. The same method can be applied to infer location data. In location distribution attacks [95], the adversary exploits the fact that users are mostly not uniformly distributed in the location space. Another attack by Shokri *et al.* [109] utilized the aggregated traffic statistics and environmental context information. The attack scenario includes an adversary who tries to reveal the possible location of the target by making use of the fact that the probability of the target's whereabouts is not uniformly distributed. Map matching methods [70] aim to restrict the obfuscated area to a smaller but plausible area by removing irrelevant areas. Movement boundary attacks were explored [54], where the adversary aims to calculate the movement boundary of a target by chasing the position queries and updates of the target. After calculating the boundary, the location of interest, such as home or workplace, is inferred and the irrelevant locations are discarded.

Although we did not directly touch upon preserving the location privacy in our study, there have

Table 2.1: A summary of the related keylogging inference attacks. 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. Top-X word means that the study matches words instead of keys and top-X PIN means that the study infers only the numerical characters.

Reference	Year	Exploited Medium	Proximity	Training?	Performance	
					Detection	Identification
Asonov et al. [31]	2004	Keyboard acoustics	1m	Yes (UP)	-	79% (top-1 key)
Berger et al. [38]	2006	Differences of keyboard acoustics	-	No	-	73% (top-50 word)
Balzarotti et al. [35]	2008	Video of the typing hands	$<$1m	No	-	46% (top-1 word)
Zhuang et al. [133]	2009	Bootstrapped keyboard acoustics	-	No	-	90% (top-1 word)
Marquardt e al. [90]	2011	Vibrations sensed via smartphone	50mm	Yes (UP)	-	43% (top-10 word)
Halevi et al. [59]	2014	Keyboard acoustics	-	Yes	-	64% (top-1 key)
Ali et al. [29]	2015	WiFi CSI distortion	4m	Yes (UP)	98%	96% (top-1 key)
Chen et al. [44]	2015	WiFi Multipath localization	5m	Yes (UP)	-	92% (top-5 key)
Wang et al. [123]	2015	Dislocation of hand	Smartwatch	Yes	57%	30% (top-5 word)
Liu et al. [81]	2015	HM+Acoustic emanations	Smartwatch	Yes	-	55% (top-5 word)
Maiti et al. [87]	2016	HM+Acoustic emanations	Smartwatch	Yes	-	51% (top-10 word)
Wang et al. [122]	2016	Hand movement	Smartwatch	No	-	80% (top-1 PIN)
Compagno et al. [45]	2016	Acoustics via VoIP	Remote	Yes (UP)	-	83% (top-1 key)
Sabra et al. [107]	2020	Video feed	Remote	No	92%	35% (top-50 word)
SIA (Chapter 4)	2021	Acoustics via Smartwatch	Remote	Yes (UP)	99%	98% (top-1 key)
SIA (Chapter 4)	2021	Acoustics via Smartwatch	Remote	Yes	99%	85% (top-1 key)

been a few related studies in this space. The fast-growing need of preserving location privacy over the aforementioned attacks excited researchers' attention. Researchers introduce obfuscation methods such as decreasing the quality of the location by introducing inaccuracy and imprecision [48]. Additionally, the term k-anonymity is defined as obscuring the location information of individuals with k number of other individuals within the region [55, 113].

Keylogging

There has been a significant number of studies on side channels and their utilization for keylogging inference, which we review in this section. We start by a brief background, followed by various techniques falling under a broad set of mediums exploited 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 attack was introduced in the early 1970s [52] 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 [96]. 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* [31, 38, 45, 59, 133], *electromagnetic emanations* [121], *vibrations* [36, 90], *motion* [81, 87, 122, 123], *visually observable clues* [35, 107], and *WiFi signal distortions* [29, 44], etc.

Acoustic Emanation for Inference. The first keylogging attack through acoustic emanation was presented by Asonov and Agrawal [31], 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. [133] 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. [38], 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 [59] 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. [45] 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. [81] 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. [87] 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. [123] utilized a language model to further increase the leakage and achieved 30% top-5 word accuracy. Wang et al. [122] 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 [36] or a hijacked smartphone with capable sensors within proximity [90].

Visual Cues. Visual cues are also considered as side-channel information in the literature. Sabra et al. [107] 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 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. [29] 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. [44] 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 2.1.

Attacks on AR/VR Keyboards. The built-in system keyboards in Magic Leap 1, Microsoft HoloLens, and other AR/VR HMDs share a similar layout. However, the mechanisms to use the keyboard, i.e., to target and select the keys, vary. Kreider [69] explores the feasibility of keylogging inference attacks on the Microsoft HoloLens system keyboard, which is used through targeting by the headset and selecting by tapping gesture. In their modeling, the adversary obtains the drawmetric profiles of a set of possible passwords in advance (10 passwords in this case) and infers which one the victim types through the analysis of the drawmetric profile extracted by manually inspecting the victim's video recording. Although the study is based on manual processing, which is impractical for random length inputs from unknown, intractably large input sets, the evidence underlines the potential privacy leakage due to the visual side-channel for AR/VR HMDs.

Ling et al. [80] demonstrate computer vision- and motion sensor-based attacks to infer passwords for two modes: (I) targeting via headset and selecting via controller (Mode 1) and (ii) targeting and selecting via a controller (Mode 2). The computer vision-based attack relies on visual data from a stereo camera and attacks on Mode 1. On the other hand, the motion sensor-based attack assumes malware installed in the victim's device to obtain motion sensor readings and can attack both modes. The attacks assume correctly guessing the keystroke to a special key (*go* key in Samsung Gear VR) or otherwise perform a brute force approach. Both approaches, however, rely on the assumption that the keyboard is fixed in position and the size in the virtual environment to

use pre-computed rotation angles for inference. On the contrary, in our study, we account for the possibility that the virtual objects can be freely positioned and scaled in the virtual environment, which otherwise limits the flexibility of virtual applications.

CHAPTER 3: EXPLOITING SHARED ELEVATION PROFILES FOR PRIVACY INFERENCE THROUGH REPRESENTATION

Geodata recorded by fitness trackers are crucial and valuable for the operation of fitness applications. However, they can also be exploited for launching attacks on users by breaching their privacy, since sensitive information of users such as home or workplace location can be easily inferred from such data. Even worse, a large number of users, when sharing such information, would be unaware of the ramifications of sharing, and the potential risk of inferring such contextual information, such as home, work location, etc., from such shared location data. To support this argument, we conducted an online survey with 60 participants who regularly use fitness trackers outdoors. The results of the survey, summarized in Fig. 3.1, reveal that 51% of the participants start their training from their homes, 36% start from their school, and 3% start from their workplace, while 76% of the participants finish their training at their homes. Moreover, for the same set of users (results are not shown in Fig. 3.1), 42% of those users have indicated that not sharing location information implies privacy protection, while 30% of the respondent were uncertain, and 28% were certain that not sharing would not necessarily mean their privacy is protected. The mixed responses highlight the gap between reality and expectations of privacy when sharing location information online and calls for further investigation.

Although it is possible to hide the location trajectory by removing the activity map in the fitness applications, users still want to share elevation profile or certain statistics of the activity to show the roughness, technicality, and difficulty of the routes they took as a measure of their workout. For example, up until recently, users have been demanding those fitness applications to allow for fine-grained and customized access control by allowing them to share, for example, the elevation profile of an activity while masking the map that highlights the actual trajectory, which is deemed of high privacy value to them [4–6, 8].

This work has been published in IEEE 40th International Conference on Distributed Computing Systems (ICDCS) 2020.

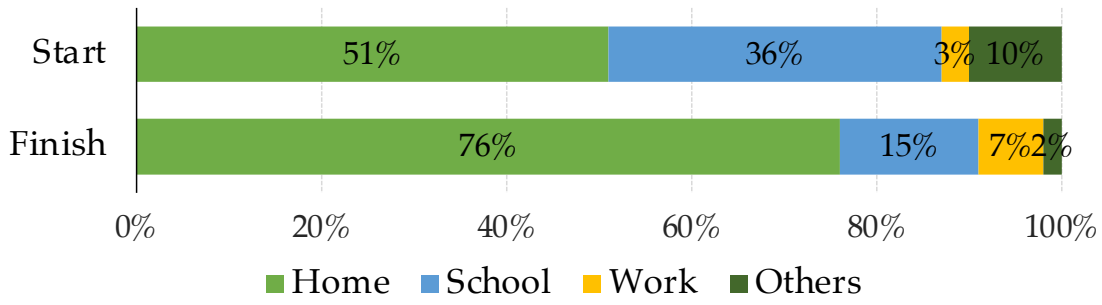


Figure 3.1: Survey results for understanding users behavior with starting point statistics and finishing point statistics. While 90% of the 60 participants indicated their start of activity is either home, school, or work, an overwhelming 98% of the participant indicated those to be the finish (end) point of their activities.

In the same survey we conducted earlier, we asked our 60 subjects "while sharing an outdoor workout record, do you think hiding the map and sharing only the statistics of your training (such as speed and elevation changes) is enough for protecting your privacy?". The results were overwhelmingly positive, with 25 of them indicating "yes", 18 indicating "maybe" (together accounting for more than 71%), while only 17 indicating "no".

Is sharing the elevation profile of an activity enough to maintain the privacy of users? In this work, we argue that an approximate location, extracted from the contexts of activities, and at different levels of location granularity, could still be revealed from the elevation profile information. We examine this problem comprehensively and develop techniques that can be used to accurately associate an elevation profile with contextual information, such as the location.

Background

In this section, we provide some background information highlighting the significance of elevation profiles for athletes, the use cases, some properties of the fitness applications on the market today, and some reported privacy breach incidences of fitness applications to further contextualize the presented study.

Table 3.1: Popular fitness applications and their features. ET: Exercise tracking. SS: Ability to share to social media. SNS: Social networking capabilities in the service. PR: Private records. BU: User blocking capability.

Service	ET	SS	SNS	PR	BU
Strava	●	●	●	●	●
Runtastic	●	●	●	○	●
Runkeeper	●	●	●	●	○
Nike+ Running	●	●	●	●	○
MapMyRun	●	●	●	●	○

Importance of Elevation Profiles for Athletes. Athletes who keep track of their activities records measure various modalities and attributes associated with the activities, including the distance, speed, overall time, and heart rate over the course of the activity. Based on these attributes, they adjust their training strategies to reach their goals. Elevation changes, often reported in the form of elevation gain, are one of the most significant attributes measuring the performance of a cyclist/runner, and often depicts how hard the run or ride is. For example, riding a bike for a 20-mile ride while climbing 1000 feet in total is significantly more challenging than biking on a flat terrain [100]. Therefore, when recording or sharing a ride/run, athletes care about the changes in the elevation, thus elevation profiles.

Fitness Applications & Privacy Breach Incidents. Fitness applications allow users to track their workout history and provide them with statistics. Moreover, some fitness applications have social network capabilities, as shown in Table 3.1, and allow users to share workout summaries that are known to motivate users and their social network connections to achieve their goals [99]. Some of the fitness applications also inherit user blocking feature and capabilities from social network platforms, including user privacy options such as private records–the activity records that are only visible to the user.

Although fitness applications have configurable privacy options, there has been a lot of privacy incidents concerning location data obtained from those fitness applications. We review some of

those privacy breaches in the following to contextualize our work in the broader privacy literature.

Revealing Secret U.S. Military Bases. Strava, which is one the most popular fitness tracking applications in the market today, collects users' public data and publishes a heatmap of the aggregates to highlight routes frequented by users [10]. Although the aggregates in the heatmap do not explicitly contain any identity information, activities in desolate places revealed the location of many U.S. military bases, which is considered sensitive information [3, 12].

Deanonymization Through Strava Segments. In Strava, the heatmap feature shows "heat" made by the aggregated and public activities by Strava users over the last year. It is, however, shown that a dedicated adversary can deanonymize heatmap to find out users who ran in a specified route [82]. For example, by selecting a route from the heatmap, a registered user can manually create a GPS eXchange (GPX) track file and create a segment using it on Strava. A segment is a portion of a road or a trail where athletes compare their finishing times. Consequently, once this segment is created, the users who previously ran that route are shown on the leaderboard grouped by gender and age. This feature is then leveraged to identify individuals who ran that particular place.

Tracking and Bicycle Theft. Users of fitness applications can share information related to the equipment used for the activity, including the bicycles, tracking devices, shoes, etc., along with the routes frequented. The combined shared information making them a target to robbery, and several such incidents of bicycle theft are reported [1, 2, 7, 11].

Attack on Privacy Zone. To cope with the increasing privacy risks, Strava features *privacy zones*, a technique to obfuscate the exact start and end points of a route. A recent study [60] has demonstrated that is possible to reveal the exact start and end point of a route that utilizes the privacy zone feature. The same study also claimed that around 95% of the users are at risk of revealing their location information.

Live Activity Breach. In Runtastic, one of the popular activity-tracking applications, users can share their live activities. In theory, users should be able to configure the privacy settings for

their activities such that only privileged users, such as connections on the application platform, can track the shared live activity session. However, it has been demonstrated [9] that the selected privacy settings are not correctly applied to a live session. As a result, everyone can go through live sessions and track Runtastic users in real-time, even though the associated privacy options should have prevented this type of breach. Based on this incident, it would be easy to stalk and locate a user, e.g., lone runner or cyclist with an expensive equipment, in real-time.

Threat Models

We outline the potential threat models under which this study is conducted. We describe three models under which the location privacy is breached only from associated elevation profiles. We note that the following threat models are only hypothetical: no attacks were actually launched on any users. As mentioned earlier, this study in its entirety is motivated by the aforementioned demands of users to have more flexibility over-sharing partial data, such as elevation profiles, and examines the ramifications of such sharing in a hypothetical setting. We note, however, that those settings are also plausible if such sharing is enabled.

Our study utilizes three threat models: TM-1, TM-2, and TM-3, which we outline below with their justifications. The adversarial capabilities in TM-1 are greater than in TM-2 and TM-3, making it more a restrictive (powerful) model.

❶ **TM-1.** In TM-1, we assume an adversary with workout history records of a target user, and the goal of the adversary is to identify the last workout location of the target user from the recently shared elevation profiles. TM-1 is justified by multiple plausible scenarios in practice. For example, such an adversary might have been a previous social network connection of the target user that was later blocked. In such a scenario, the adversary may have previous workout records of the target from which the adversary may attempt to de-anonymize the target's activities. Another example might include group activities, in which two individuals (i.e., the adversary and target) may have shared the same route at some point. In either case, by knowing the target's previous fitness

activity records, the main goal of the adversary in this model is to identify recent whereabouts only from publicly shared elevation profiles in workout summaries, thus breaching the target's location privacy.

❷ TM-2**.** In TM-2, we assume an adversary with access to limited information such as the city where the target lives. Such information is easily accessible from public profile summaries, `athlinks.com`, public records, etc. The adversary's goal in TM-2 is to find out which region or part of a given city the target's activities are associated with. The TM-2 use scenario may include a targeted user sharing private activities, in which the route is hidden while the elevation profile is shown. The adversary, knowing the city where the target lives, would want to identify the region (e.g., a borough in the city) associated with the user's activity.

❸ TM-3**.** In TM-3, we assume an adversary trying to identify the target user's city using only publicly shared elevation profiles without any prior information. We assume, however, the adversary has the ability to profile the elevation of cities, with information that is easily obtained from public sources (e.g., Google Maps, OpenStreetMap). The use scenario of TM-3 may be used as a stepping stone towards launching the attack scenario in TM-2 upon narrowing down the search space to a city.

Approach: High-Level Overview

In this section, we give a brief overview of our pipeline, which consists of the data collection, preprocessing, feature extraction, and classification as illustrated in Fig. 3.2. Each phase of the pipeline is detailed in Implementation Details section.

Data Collection. We collected three datasets with varying and rich characteristics, namely (i) user-specific activity data collected from an athlete, (ii) mined training route segments grouped at city-level, and (iii) mined training route segments grouped at borough-level. For the user-specific dataset, we collected physical activity records of athletes and converted those activities to an inter-

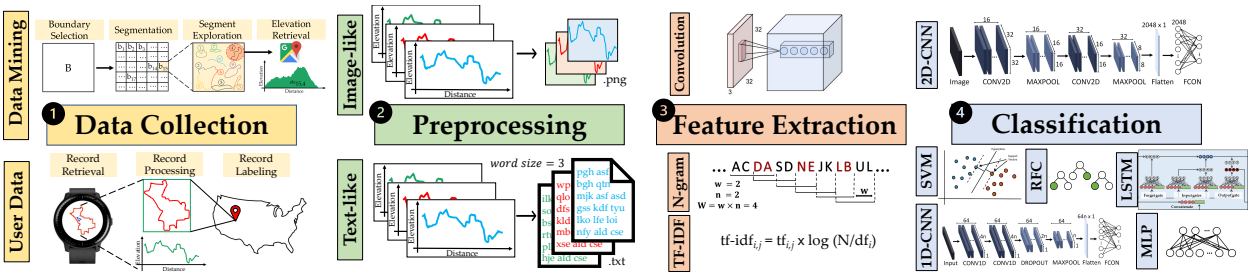


Figure 3.2: The end-to-end pipeline of the approach consisting of four main stages: (i) data collection, (ii) preprocessing, (iii) feature extraction, and (iv) classification. There are two types of data representations, each of which is processed differently in the feature extraction and classification stages. The feature extraction of the image-like representation is internally handled in the convolution layers of the classification phase. The convolution illustration in the feature extraction step is for the sake of modularization and for consistency with the other pipeline instance.

mediate format, the GPS Exchange Format (GPX). Then, we parsed the GPX files and manually labeled them according to the latitude and longitude information included within each file. For the second dataset, we mined training route segments from a popular fitness tracking website by specifying the location boundaries, *i.e.* the class label of the mined data, and augmented each segment with the corresponding elevation profiles obtained from Google Maps Elevation API. Finally, we constructed the borough-level dataset in a similar manner as in the city-level dataset.

Preprocessing. We employ Natural Language Processing (NLP) and computer vision techniques to convert the problem to text classification and image classification problems, respectively. To this end, we prepare the data accordingly in the preprocessing phase. Preprocessing consists of two parts: (i) text-like and (ii) image-like representations.

For text-like representation, we discretize the elevation signals and compute the minimum required *word* size. We then create a mapping between each unique discrete value and a string. By mapping the string correspondents to the unique discrete values, we encode the elevation profiles in text. We, then, form a *vocabulary* from the text sequences of each dataset using the n-grams.

To obtain image-like representations, we convert the elevation profiles to a fixed-sized line graph where the x-axis stands for time and the y-axis stands for the elevation values. We also color the

lines in the graphs to represent the elevation interval in which the elevation profiles range.

Feature Extraction. The classification algorithms operate on high-quality and discriminative features, obtained from the representations of elevation profiles. For feature extraction, we utilize NLP and computer vision approaches.

To employ NLP approaches using the vocabulary obtained in preprocessing phase, we represent each elevation profile as either a feature vector based on the frequency of the vocabulary in the text-like representation (bag-of-words vector) or as a term frequency-inverse document frequency (tf-idf) vector. To employ computer vision approaches, we utilize Convolutional Neural Network (CNN) over image-like representations. The optimal features of an image-like representation are efficiently extracted by the convolutional and pooling layers in the CNN architecture.

Multi-Class Classification. We utilize various machine learning and deep learning models for classification including Support Vector Machine (SVM) and Random Forest Classification (RF), Multi-Layer Perceptron (MLP), Long Short-Term Memory (LSTM), 1D Convolutional Neural Network (C1D), and 2D Convolutional Neural Network (CNN).

Implementation Details

The implementation details of data collection, preprocessing, feature extraction, and multi-class classification are addressed in the following subsections.

Data Collection. In this study, we compiled three datasets: the user-specific dataset, the city-level dataset, and the borough-level dataset. The user-specific dataset is retrieved from a voluntary athlete who frequently records activities through fitness applications. It offers dense and thorough coverage of regions frequented by the user; those regions are used as class labels. The city-level and borough-level datasets are created from scratch by collecting location trajectories that are created and frequented by the athletes. Both city-level and borough-level datasets provide sparse coverage of cities and boroughs.

User-Specific Dataset. For the user-specific dataset, we collected activity data including each activity's location trajectory and the corresponding elevation profile from a voluntary athlete who records activities frequently through fitness applications. First, the location trajectories included in the user-specific dataset are converted to GPX format to avoid confusion caused by different formats and settings across the activity records. Then, to label the samples, the maximum and minimum coordinates of each location trajectory are fetched. Each sample location trajectory is encapsulated with a tight rectangle whose top right (North East) and bottom left (South West) corners are computed from the maximum and minimum coordinates of the trajectory as illustrated in Fig. 3.4. To classify the samples, each rectangle encapsulating the trajectory is compared with the previously created regions. If the Euclidean distance between the center of the rectangle and the center of the existing region does not exceed a predetermined threshold, the rectangle and its corresponding sample are labeled with a unique identity of the region. If there is no region that includes the trajectory, a new region is created. The final sample size distribution of the user-specific dataset is shown in Table 3.2.

The user-specific dataset is prone to have similar location trajectory portions across its samples since the user may frequent the same set of places in his/her everyday activities, such as the location trace they follow while leaving/arriving home, or their favorite routes. Therefore, we calculated the average overlap ratio of the routes included in the user-specific dataset by comparing each sample with the other samples with the same class label. For each sample pair comparison, the overlap ratio is calculated as the intersection-over-union of the tight rectangles encapsulating the sample routes. The average overlap ratio of the user-specific dataset is calculated as 35%.

City-Level Dataset. For the city-level dataset, we mined **publicly available** training route segments in a popular fitness tracking application using its EXPLORESEGMENTS() functionality. We note that our experiments do not put any users at risk and are not in violation of the terms of use of the fitness tracking application: since both the trajectory (map) and elevation profiles are public information, we are also not obtaining any information beyond what is provided by the users

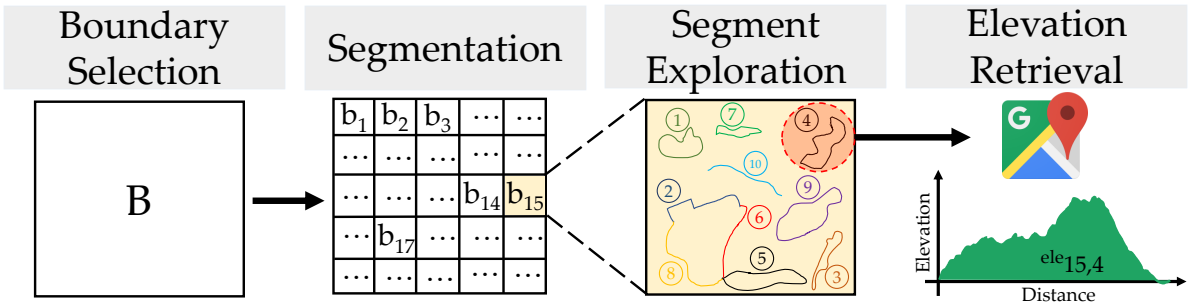


Figure 3.3: An illustration of the data mining pipeline. A geolocation boundary, B, is segmented into small boundaries, each of which is then forwarded to the Segment Exploration step to obtain the most frequented 10 route segments for that particular boundary. Finally, the elevation profile of each route segment is retrieved from the Google Maps API.

explicitly. We also note that the training route segments are user-created activity routes whose main purpose is to compare completion times among users who also completed the same route. They are particularly useful for our purposes since they include public location trajectories that are frequented by the actual users rather than randomly created location trajectories that may not necessarily be of privacy value. During mining the segments, the anonymity, thus the privacy, of the users who frequented the segments or created the segments is maintained.

The overall data mining procedure consists of three steps, as illustrated in Fig. 3.3. First, we define the cities of interest, which we also use as the class labels per our threat model. For each city, we define the rectangle geolocation boundary box B consisting of the top right and bottom left corner coordinates in the boundary selection phase. In the segmentation phase, and since EXPLORESEGMENTS() returns only the 10 most frequented segments encapsulated by a given boundary, and to obtain more data of a geolocation boundary box, we divide the large rectangle boundary of the city into smaller region boundaries, each denoted by b_i, by following a grid-like structure as shown in the second phase of the Fig. 3.3. For each region boundary b_i, we call EXPLORESEGMENTS() and receive the geolocation polyline path, $path_i^j$ where $j \in [1, 10]$, of the 10 most frequented segments encapsulated in b_i, as shown in the segment exploration phase. Finally, since the polyline paths do not include elevation profiles, we obtain the associated elevation

Figure 3.4: An illustration of the tight rectangle encapsulating an example route. The rectangle is created by fitting the route in between the minimum and maximum $(latitude, longitude)$ pairs of the given route. The minimum and maximum $(latitude, longitude)$ pairs correspond to the bottom left (*i.e.* South West) corner and the top right (*i.e.* North East) corner of the rectangle respectively.

profile $ele_{i,j}$ for each $path_i^j$ using the Google Maps Elevation API through the elevation retrieval phase. The sample size distribution of the city-level dataset can be found in Table 3.3.

Unlike the user-specific dataset, the city-level dataset does not include any overlapped samples since each region r_i is disjoint with the other regions. A segment route that is included by more than one neighboring region is not considered, since EXPLORESEGMENTS() returns the routes that are encapsulated within the given boundaries, b_i.

Borough-Level Dataset. For the borough-level dataset, we apply a similar mining procedure as we have done with the city-level dataset, using the borough boundaries instead of the city boundaries. Table 3.4 shows the sample size distribution of the borough-level dataset for different cities.

Preprocessing. A key design element in our pipeline is the representation modality of the elevation profile which will significantly impact the performance of our elevation-location mapping, as we

Table 3.2: User-specific dataset sample size distribution.

Regions	Abbreviation	Sample Size
Washington DC	WDC	366
Orlando	ORL	232
New York City	NYC	120
San Diego	SD	18

Table 3.3: City-level dataset sample size distribution.

Regions	Abbreviation	Sample Size
New York City	NYC	2437
Washington DC	WDC	2129
San Francisco	SF	743
Colorado Springs	CS	369
Minneapolis	MIN	363
Los Angeles	LA	280
New Jersey	NJ	266
Duluth	DUL	156
Miami	MIA	94
Tampa	TAM	83

show later. We transform the samples into text-like and image-like representations to facilitate feature extraction and feed into our classification module. In this section, we describe the details of the utilized preprocessing methods.

Text-like Representation. For our text-like representation, our approach follows four steps, as shown in Fig. 3.5: discretization, word size decision, text encoding, and vocabulary creation.

Discretization: In the discretization step, the original elevation signal is discretized along the y-axis, which represents the elevation values to avoid possible overhead by small differences in the precision causing longer string correspondences and, consequently, longer vocabulary and sparse feature vectors. The discretization is done as follows. Let e_i^j be the i-th elevation

Table 3.4: Borough-level dataset sample size distribution.

City/State	Region	Sample Size
Los Angeles (LA)	Downtown	280
	Santa Monica	128
	Chinatown	46
	Beverly Hills	38
Miami (MIA)	Downtown	67
	Miami Beach	44
	Virginia Key	18
New Jersey (NJ)	Jersey City	266
	West New York	23
	Newark	28
New York City (NYC)	Manhattan	2437
	Queens	353
	Brooklyn (South)	239
	Brooklyn (North)	205
	Bronx	142
	Staten Island	119
San Francisco (SF)	South West	743
	South East	144
	North West	130
	North East	86
Washington DC (WDC)	District of Columbia	2129
	Baltimore	218

value in j-th sample. The discretization functions are defined as $f(e_i^j) = \lfloor e_i^j \rfloor$ and $f(e_i^j) = \frac{\lfloor e_i^j \times 10^3 \rfloor}{10^3}$, where the first function is used for processing the user-specific dataset and the second function is used for processing the city-level and borough-level datasets. Since the user-specific dataset is dense in terms of sampling rate, using the floor function is enough to represent the routes. However, as the city-level and borough-level datasets are already sparse, losing information is undesired, so we used the second function to represent the elevations that differ in up to 3 decimal digits precision. To demonstrate the effect of discretization,

we measured the vocabulary size of the smallest class of the User-specific dataset, i.e., San Diego. For a class as small as the San Diego class, using the second function results in a vocabulary of size 12,870 and using the first function results in a vocabulary of size 3,155. Such difference in the vocabulary size demonstrates the effect and necessity of discretization.

Word size decision: For word size decision, we use $w = \log_l c$, where w is the word size, l is the length of the alphabet, and c is the number of unique value occurrences in the given signals.

Text encoding. For text encoding, each unique value in all the discrete signals is mapped to a unique string with length w, and each sample signal is encoded by referring to the string correspondences of each value in the discrete signal and concatenating these strings to construct a long text, *i.e.* corpus.

Vocabulary creation: To create our *vocabulary*, we consider the corpus created from all encoded signals regardless of labels. Each line in the corpus represents a sample signal, and each word in a line represents the text correspondence of an elevation value in the sample signal. We build a vocabulary from the unique word-based n-grams of the document. As illustrated in Fig. 3.6, a window with size $W = w \times n$ is slid throughout the corpus and each window content is appended to the vocabulary set. Since the vocabulary set does not contain duplicate entries by definition, we construct the vocabulary consisting of unique n-grams of the given dataset after traversing the corpus by n times with different window sizes.

Image-like Representation. In the image-like representation, the elevation signals are drawn as line graphs and saved as a 32×32 images[1]. To draw a line graph, the maximum and minimum values for the *y*-axis are set to be the maximum and minimum of each elevation signal, and the lines are colored to encode the value interval in which the elevation signal ranges. We note that the image-like representation with colors has multiple advantages over the black-white representation where the *y*-axis is set to the range of a whole dataset. First, as illustrated in Fig. 3.7, the alterations

[1]The size is chosen to strike a balance between the performance in terms of the required computations and produced accuracy.

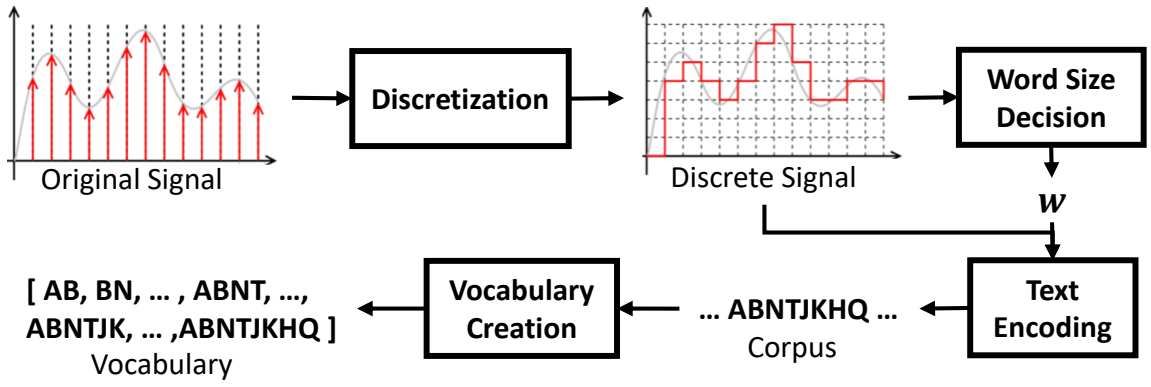


Figure 3.5: Illustration of the flow of text-like preprocessing. The signal is discretized by eliminating the small elevation fluctuations. The discretized signal is also used for deciding the word size of the encoding. The discrete signal is then encoded in text and a vocabulary is built.

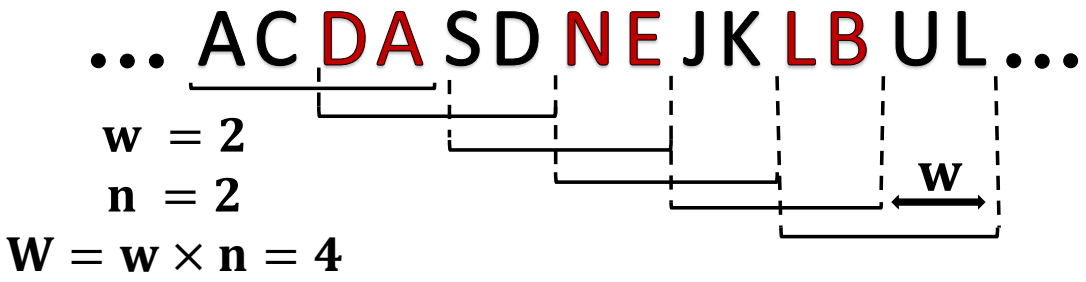
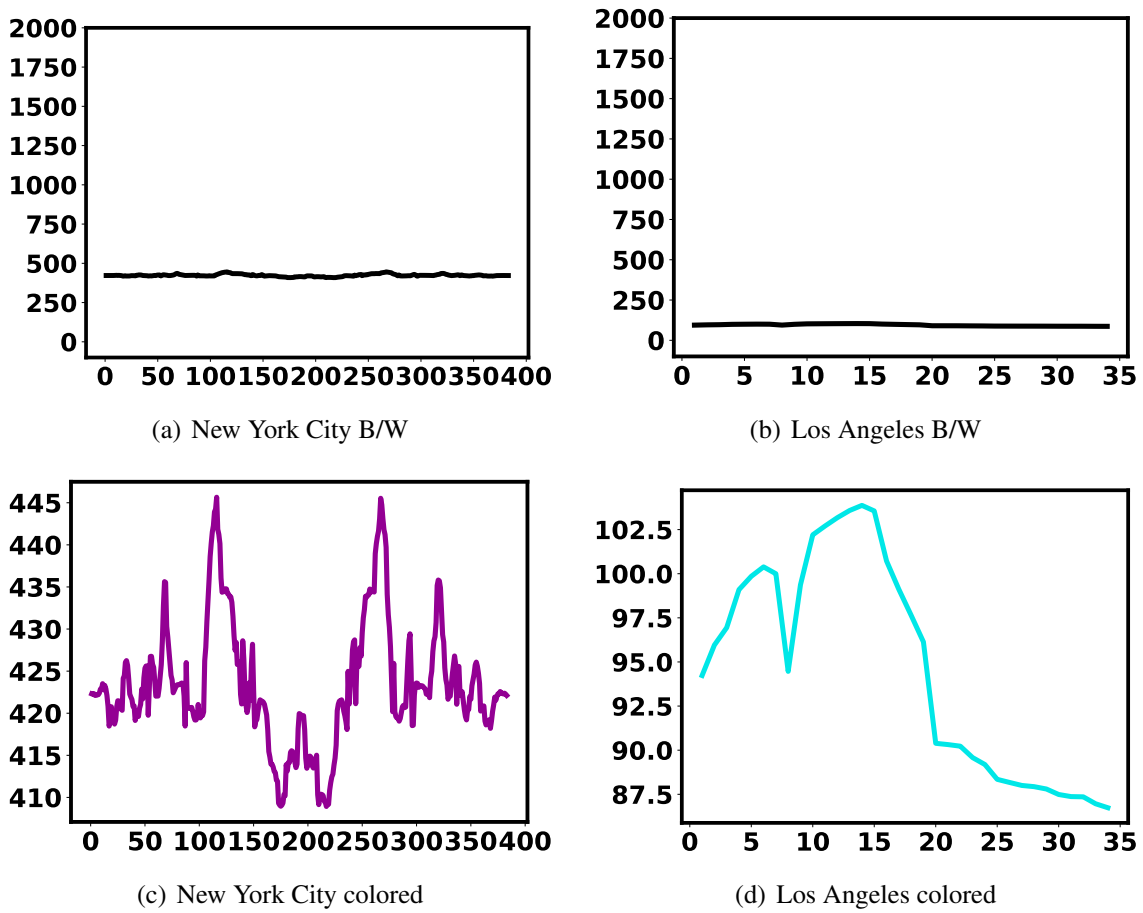


Figure 3.6: Illustration of bi-gram creation where the word size is $w = 2$ and window size is $W = 4$.

of an elevation signal are more visible with the color encoding method, which could be a more discriminative feature to learn. Second, the color encoding method results in an efficient utilization of the feature space. We use 200 elevation values for each image, obtained by dividing the elevation signal into equal-sized parts.

Feature Extraction. To classify elevation profiles accurately, we extract discriminative features from the elevation profile representations.

Text-like. In the text-like feature extraction, we utilized two methods: (i) n-grams, (ii) tf-idf.



(a) New York City B/W

(b) Los Angeles B/W

(c) New York City colored

(d) Los Angeles colored

Figure 3.7: Elevation profile graphs by fixing the *y*-axis range and using only black versus elevation profile graphs by fitting and using color encoding. As can be seen, with the black option the elevation range can be represented by the position of the signal, but changes in the signal are not visible, which is an information loss. However, with the color option enabled, both alterations in the signal and the signal range are represented in one image.

For the n-grams method, words and non-overlapping occurrences of word sequences are counted, a feature vector for each sample is created with each unique word sequence count being a feature. Finally, the feature vectors are normalized where each feature represents the occurrence probability of each word in the given sample.

The tf-idf is a statistical feature signifying the importance of a word in a document. The tf-idf values proportionally increase as the number of appearances of a word in a document increases. Technically, the tf-idf for a word is the multiplication of two metrics: (i) term frequency (tf) and

(ii) inverse document frequency (idf). Tf-idf of a word t in a document d included in the set of documents D of which cardinality is N is calculated as follows:

$$\text{tf-idf}(t, d, D) = \text{tf}(t, d) \,.\, \text{idf}(t, D), \tag{3.1}$$

$$\text{tf}(t, d) = \log(1 + \mathsf{freq}(t, d)), \tag{3.2}$$

$$\text{idf}(t, D) = \log\left(\frac{N}{\mathsf{count}(d \in D : t \in d)}\right). \tag{3.3}$$

The higher the tf-idf means that the word t is more relevant to the particular document d.

Feature Selection. When the dataset is large and diverse, the vocabulary and, consequently, the feature vector representation become too large to process, compute, and learn from. With a feature selection phase to address long feature vectors, some rarely occurring features in the vocabulary are discarded according to a pre-specified feature frequency threshold. For selection, the features are ordered by their term frequency across the corpus, the features whose term frequency is below a specified threshold are discarded, and a new vocabulary is created. For both feature extraction methods of the text-like representations, the term frequency threshold is set such that the size of the eventual vector representation is 5,000.

Image-like. We use the CNN for processing and classifying the 32×32 images, thus it is unnecessary to explicitly extract features since the convolutional layer kernels do that already by learning the filters optimally and efficiently. Therefore, the actual feature extraction mechanism for the image-like representation is discussed in the context of classification phase.

Multi-Class Classification. For classification, SVM, RF, MLP, LSTM, and CNN are used.

Support Vector Machine (SVM): SVM is a supervised classification technique. The main challenge in SVM is finding the best hyperplane that divides the classes from each other considering a given margin. SVM with linear kernel is able to distinguish the classes more successfully when the features are multi-dimensional and numerous [61, 86]. In linear kernel

settings, when an optimal hyperplane is found while representing a class, only the features around the hyperplane within the given margin are considered and the other features are simply ignored. As such, the complexity of SVM is independent of the number of features. Since we consider n-grams up to n=5 in the feature extraction phase, the number of features is large, which makes using SVM justified in terms of efficiency and success.

SVM with linear kernel is generally used for binary classification. To utilize it for the multi-class classification problem, we use the one-versus-rest method. In this method an individual model is trained for each class and the label of the most confident model is outputted. For the penalization, we use L2 norm, as it is a standard for linear SVMs. As for the loss function, we utilized square of hinge loss, and the hyperparameters are decided through grid search tuning.

Random Forest (RF): RF is based on decision trees, which are ensemble learning methods for classification. The main feature of ensemble methods is further improving the generality and robustness of a single estimator by combining several base estimators that are built with a given learning algorithm. In decision trees, features are represented by tree nodes and each branch between two nodes represents what the immediate ancestor node returned. Since building an optimal binary decision tree from given features is an NP-complete problem, using a Random Decision Forest with different tree configurations and efficient heuristics is a way to alleviate the NP-completeness for classification problems. While creating our random forest, perturb-and-combine techniques are used. Perturb-and-combine techniques are designed specifically for decision trees to improve their accuracy by creating several (different) versions of the estimator by *perturbing* the training set, then *combining* these different versions into a single estimator [39]. Further details on this approach can be found in [40] and [39].

In this study, we use 20 decision trees for the RF model. The final prediction is then done by averaging the tree predictions. We do not set any upper limit on the number of the leaf

nodes or the depth of the tree, i.e., there were no time optimization concerns during training the process, so we leave the trees grow to their maximal depth.

Multi-layer Perceptron (MLP): MLP is a feed-forward fully-connected neural network which utilizes backpropagation for training and is used for supervised learning. Recent studies on comparing multi-layer neural networks and decision trees [49, 78] concluded the following:

- Multi-layer neural networks allow incremental learning, in which the model's knowledge is continuously extended, more easily than the decision trees.
- The training time of the multi-layer neural networks is much longer than decision trees.
- The multi-layer neural networks predictions are generally as good as the predictions produced by the decision trees, although they can perform better in certain cases.

Given the aforementioned potential for MLP performing better than RF, we used MLP with 20 hidden layers in our experiments. We used Adam solver [65] for weight optimization, since it is shown to perform well in terms of both training time and validation score for large feature spaces. We also utilize ReLU as the activation function, 0.001 as the learning rate, and 200 as the epoch size. For regularization, we use L2 norm with 0.0001 penalty parameter. The hyperparameters are decided through grid search tuning.

Long Short-Term Memory (LSTM): LSTM is a recurrent neural network architecture. Unlike the standard neural networks, LSTMs are capable of keeping track of long-term dependencies in the input sequences using feedback connections. Such long dependencies are handled through feature extraction in n-grams and tf-idf vectors for standard neural networks. LSTM, on the other hand, handles dependencies internally, without requiring an explicit feature extraction. LSTM is also particularly useful for capturing the order dependence in sequence prediction problems. For LSTM, the input sequence can be a time series, a sentence from a given language, or a text-like representation as in our application case. Fig. 3.8 depicts the LSTM architecture employed in this work, where the input is directly passed through the

LSTM layer with four hidden layers, and two fully-connected layers following the LSTM layer. Each input sample (vector representation of an elevation profile for n-grams and tf-idf, or individual values for the raw data) is passed through the LSTM unit. When all sample vectors (or elevation values) are passed through the LSTM units, the final hidden state vector is passed to the subsequent fully-connected layer. In the LSTM unit, hyperbolic tangent and sigmoid (depicted as T and S, respectively, in Fig. 3.8) are used as the activation function. For the fully-connected layers, ReLU and softmax activation functions are used, respectively.

Convolutional Neural Network (CNN): CNN is similar to neural networks in mechanism. Both of them consist of neurons with learned parameters, weights, and biases. The improvement of CNN, however, is in the form of convolution layers, which apply forward passes that decrease the amount of parameters of the neural network considerably. Convolution layers also facilitate processing high-dimensional data, such as images. They prepare high-dimensional data for fully-connected layer, which cannot process high-dimensional data efficiently, by highlighting the important spatial features along the way.

In this study, we utilize two CNN architectures, differing in the convolution layers dimensions. Fig. 3.9 illustrates the employed CNN architectures.

In the first architecture, we use two consecutive 2D convolution layers (CONV2D) along with the ReLU activation function and MAX pooling layers (MAXPOOL) before a fully connected layer (FCON). For both of the convolution layers, kernel, stride, and padding sizes are determined as 5, 1, and 2, respectively, based on the performance. The distinctive features are selected at the max-pooling layers with a kernel and a stride size of 2, which reduce the dimensions from (32×32) to (8×8) at two passes.

In the second architecture, we used two consecutive 1D convolution layers (CONV1D) along with the ReLU activation function. A dropout (DROP) layer is added to alleviate the overfitting problem. Then, a max-pooling layer (MAXPOOL) and a fully connected layer (FCON) are added.

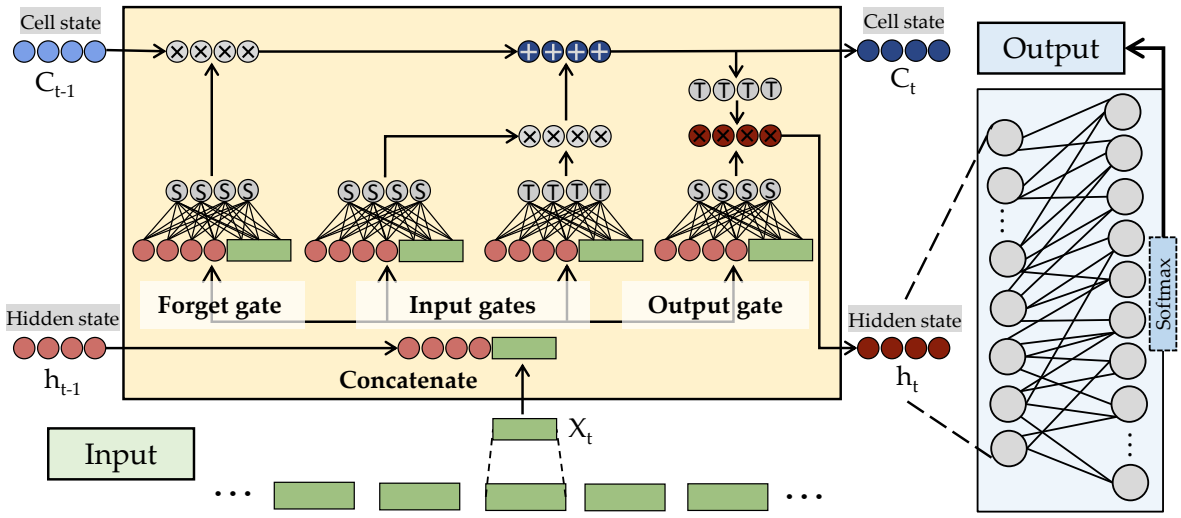


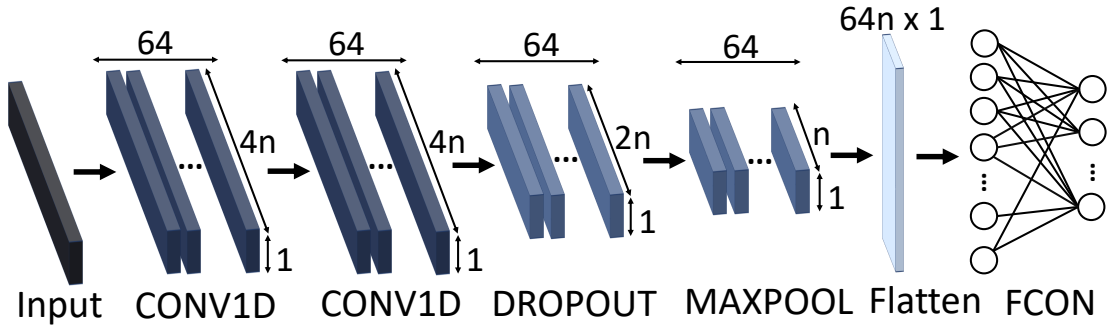
Figure 3.8: The architecture of the LSTM network consisting of an LSTM unit with four hidden layers and two fully-connected layers. The input samples are passed through the LSTM unit individually and the last hidden state of the LSTM unit is forwarded to the fully-connected neural network. The fully-connected neural network utilizes softmax activation at the output layer which outputs the class probabilities.

For both architectures, the softmax function is used as an activation function at the output layer and the Categorical Cross Entropy is used as a loss function. For parameters optimization, we used the Adam optimizer.

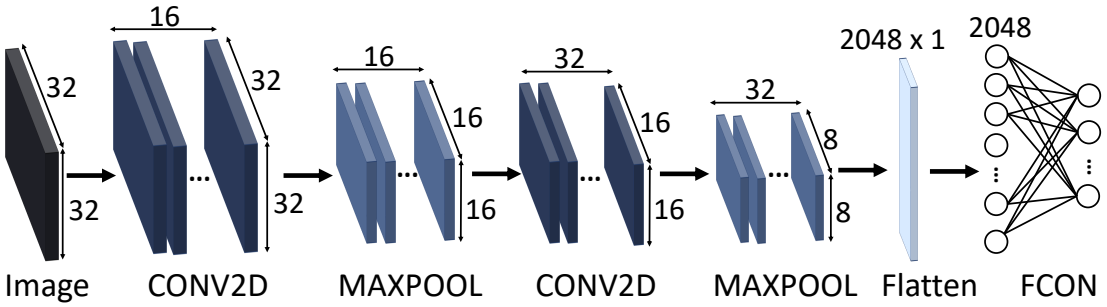
Evaluation, Results, and Discussion

We performed experiments for each dataset, data representations, and threat models. We categorize the evaluations into three: Raw, Text-like, and Image-like.

Raw. First, we performed evaluations on the raw data as a baseline. As the multi-class classification models require fixed input shape, we divided the elevation profiles into equal-length (32) chunks and use the raw data to train and test the models. For all datasets and threat models, we used a slightly modified version of the soft voting ensemble method while testing with raw data. Instead of passing a single input to different models, we passed the equal length chunks of a single



(a) 1-D CNN Architecture



(b) 2-D CNN Architecture

Figure 3.9: The CNN architectures used for classification. A 1-D CNN is used for the one-dimensional data representations, i.e., n-grams, tf-idf, and raw data. A 2-D CNN is used for the image-like representation. The numbers in the figures depict the dimensions of the input throughout the learning/prediction process.

elevation profile individually to a single model and decided the final prediction through soft voting. Soft voting sums the predicted probabilities for each class label and returns the class label with the highest probability as a final prediction.

Text-like. Second, we performed evaluations with text-like features: n-gram, and tf-idf. With n-gram features, we performed experiments using 10-fold cross-validation and by fixing the dimension of n-grams to 5 for all datasets and associated threat models. With tf-idf features, we performed 10-fold cross-validation and fixed the dimension of n-grams to 5 for all datasets and threat models.

The user-specific dataset contains overlapped and repetitive portions by nature. In the Simulations subsection, we simulated the same behavior on the mined datasets and performed the same

Table 3.5: The overall evaluation results for TM-1. Accuracy (%) with different data representations and classification models. In this table, the following abbreviations are used: SVM: Support Vector Machine; RF: Random Forest; MLP: Multi-Layer Perceptron; C1D: 1D Convolution; LSTM: Long Short-Term Memory. **C** column indicates the number of classes in the classification problem. The following settings are used: 4-class = [WDC, ORL, NYC, SD], 3-class = [WDC, ORL, NYC], 2-class = [WDC, ORL].

C	raw data					n-grams					tf-idf				
	SVM	RF	MLP	C1D	LSTM	SVM	RF	MLP	C1D	LSTM	SVM	RF	MLP	C1D	LSTM
2	95.29	98.43	96.60	97.45	96.61	97.35	98.22	99.11	99.74	49.67	98.89	97.56	98.89	**99.94**	51.11
3	77.56	98.64	95.88	96.20	96.46	96.79	97.53	97.93	99.23	45.83	98.86	97.91	98.48	**99.00**	42.60
4	70.39	96.51	71.87	74.17	75.78	93.33	87.99	95.66	**99.80**	38.67	92.33	90.66	92.33	99.54	33.33

evaluations for comparison.

Image-like. For the experiments on the image-like representations, we employed three methods in CNN: unweighted loss function, weighted loss function, and fine-tuning. In the unweighted and weighted loss function evaluations, we split the test data from the dataset by considering the sample size of the classes; we assigned probabilities for each class considering the inverse proportion to its size and then randomly selected test data with the associated probabilities. In fine-tuning evaluations, we performed 10-fold cross-validation at the last round where all the classes have the same sample size.

Raw and Text-like Data Evaluation.

❶ **Evaluating TM-1.** We trained and tested models with the user-specific dataset. As shown in Table 3.2, the user-specific dataset has an unbalanced sample size across classes. To mitigate bias, we use the same sample size for each class and change the number of classes at each step. The evaluation results are shown in Table 3.5. Due to the limited number of samples, the accuracy decreases as the number of classes increases. The only exception is C1D with n-grams and tf-idf. One dimensional convolutions were able to capture the characteristics of elevation profiles even with a limited number of samples. The results show 99.80% accuracy with C1D, n-grams and 4-class classification. With tf-idf and C1D, we obtained 99.00% and 99.94% accuracy with 3-class and 2-class classification, respectively.

Table 3.6: The overall evaluation results for TM-2. Accuracy (%) with different data representations and classification models. In the table, we use the following abbreviations: LA: Los Angeles; MIA: Miami; NJ: New Jersey; NYC: New York City; SF: San Francisco; WDC: Washington, D.C.

	raw data					n-grams					tf-idf				
Cities	SVM	RF	MLP	C1D	LSTM	SVM	RF	MLP	C1D	LSTM	SVM	RF	MLP	C1D	LSTM
LA	67.18	77.41	63.43	32.50	36.88	**78.02**	74.33	77.27	55.29	28.25	76.27	76.02	75.33	58.49	23.00
MIA	69.57	80.85	67.17	80.55	54.10	75.55	77.55	75.77	77.33	40.33	83.77	82.44	72.00	**99.54**	25.33
NJ	65.56	82.56	78.31	83.32	74.18	74.76	66.19	82.39	71.19	39.05	77.93	82.69	65.71	**92.42**	32.14
NYC	73.63	**84.26**	73.44	37.33	25.57	82.25	80.71	79.78	73.02	20.72	81.50	82.96	82.63	76.60	17.94
SF	65.92	74.66	65.89	42.21	32.53	74.71	76.15	**80.25**	54.08	25.88	76.13	75.71	74.71	58.07	27.92
WDC	53.08	77.30	60.44	64.01	56.05	76.13	70.63	67.20	**89.61**	58.74	75.44	74.34	55.50	85.24	51.62

LSTM gives better accuracy with raw data compared to the other text-like representations. LSTM performs better on the data where the ordering is decisive. With the text-like representations, the original ordering of the values is encoded disparately, thus LSTM could not extract much information through the ordering.

For TM-1, RF also performs better with raw data. Since extracting features from the range and the ordering of the values are less demanding for decision trees, it is reasonable to observe such pattern.

Other classification methods, i.e., SVM, MLP and C1D, benefit more from the n-grams and tf-idf features.

Since the user-specific dataset is compiled from actual users, exhibiting mobility patterns, about 35% of the routes are overlapped. In a repetitive and overlapped setting, both training and testing splits may contain similar patterns leading to the high accuracy scores. The results prove that a targeted attack on a person whose activity history is known will be successful with accuracy as high as 99.80%.

❷ **Evaluating TM-2.** While evaluating TM-2, the borough-level dataset is used. Individual models are created for each of the cities, by labeling the data as the name of the corresponding borough and evaluated separately. Similar to user-specific dataset, borough-level dataset also has unbalanced sample size across the classes. To avoid biased results, we fix the sample size to that of smallest

class for all classes. At each fold, we randomly select train and test data for the classes with more sample. Table 3.6 shows the accuracy results of each model.

Los Angeles model reaches up to 78.02% accuracy with n-grams and SVM. Similar results are obtained with other classification method and data representation pairs, such as raw and RF, n-grams and MLP, tf-idf and SVM. For Los Angeles, C1D could not become prominent; less complex models perform better on this dataset.

With **Miami**, we reach up to 99.54% accuracy with tf-idf and C1D. Overall, tf-idf is shown to be a better representation for this particular dataset. Combining a complex model with a representative feature, we achieved high accuracy.

For **New Jersey**, we achieve 92.43% accuracy with tf-idf and C1D. According to the results, tf-idf features better represents the New Jersey dataset.

In **New York City**, we reach up to 84.26% accuracy with raw data and RF. When we examine the dataset, we observed that the elevations fluctuate mostly between 13 ft and 95 ft. When such small range is considered, the decimal digit precision plays an important role. Since we do not discard any precision in the raw dataset, it is reasonable to have better accuracy with raw data. Although the highest accuracy is obtained with raw data and RF, tf-idf is a better choice for other classification methods.

In **San Francisco**, we achieve 80.25% accuracy with n-grams and MLP. Both text-like representations present similar accuracy patterns.

For **Washington DC**, we obtain 89.61% accuracy with n-grams and C1D. For both text-like representations, C1D shows better performance than other methods.

Overall, we can clearly observe the difference between TM-1 results and TM-2 results. The two main reasons for this performance gap are that (i) there is no overlapped or repetitive routes among the mined segments in the borough-level dataset, and (ii) the elevation differences and elevation sequences are not distinctive enough within a city to decide in which borough is the given test data

Table 3.7: The overall evaluation results for TM-3. Accuracy (%) with different data representations and classification models.

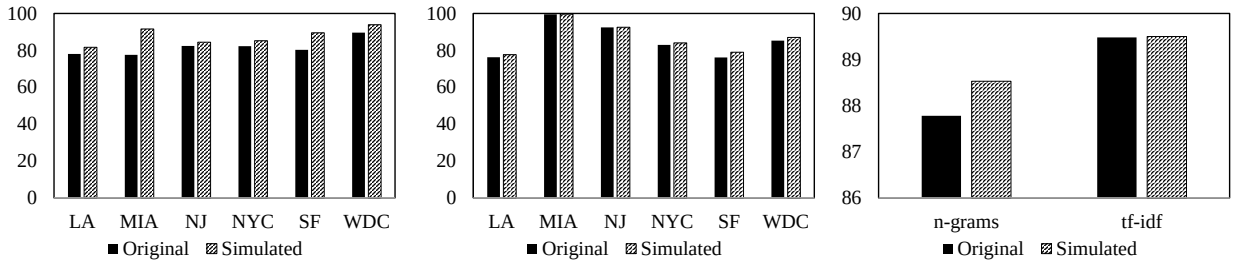
C	raw data					n-grams					tf-idf				
	SVM	RF	MLP	C1D	LSTM	SVM	RF	MLP	C1D	LSTM	SVM	RF	MLP	C1D	LSTM
3	61.53	**88.99**	64.29	82.53	56.15	76.70	78.04	77.20	65.23	33.92	85.22	81.33	81.81	65.99	33.20
5	73.14	**93.00**	73.53	65.65	70.11	80.33	78.67	79.53	53.92	20.28	86.18	82.78	84.11	52.38	20.11
7	77.58	**94.94**	80.60	52.58	64.98	85.22	84.73	84.82	43.50	13.86	88.59	88.01	87.27	48.14	14.23
8	80.60	**95.98**	80.50	44.57	67.55	84.77	84.85	85.12	43.79	14.03	87.19	86.24	86.55	50.28	12.50
10	83.72	**95.36**	84.11	40.81	59.20	87.46	87.78	87.12	31.22	16.95	89.48	87.99	88.41	43.86	12.56

is. The results of the simulated behavior will be discussed in the simulations subsection.

❸ **Evaluating** TM-3**.** In TM-3 evaluations, due to sample size differences across the labels in the city-level dataset, we follow the same procedure in TM-1 evaluations. Fixed number of samples are randomly selected from each class for training and testing. Table 3.7 shows the results of the evaluation. Per the reported results, we are able to predict the city of an elevation profile among 10 cities with an accuracy of 95.36%, among 8 cities with an accuracy of 95.98%, among 7 cities with an accuracy of 94.94%, among 5 cities with an accuracy of 93.00%, and among 3 cities with an accuracy of 88.99%. For TM-3, for all number of classes, we find the best performing configuration as raw data and RF. When we look into the reason for the fact that raw data with RF outperforms every other configuration, we observe that the elevation range of the different classes in this dataset plays an important role, similar to TM-2: NYC. Decision trees in RF are able to capture the features from the first hand, without any representation needed in the middle.

When we consider the text-like representations, we observe that tf-idf features better represent the dataset. The success of the city-level estimations, when compared to the borough-level estimations (TM-2), is due to the elevation range and sequence differences across cities, which is reasonable, even though the dataset is mined in a similar fashion as in the borough-level dataset. This mining indicates that the city-level dataset also does not contain comprehensive, repetitive, and overlapped samples. The results of the simulated evaluation will be discussed in the following.

Simulation: Raw and Text-like Data Evaluation. The mined datasets do not contain overlapped



(a) Simulation of TM-2 with n-gram features

(b) Simulation of TM-2 with tf-idf features

(c) Simulation of TM-3

Figure 3.10: Some selected simulation results of TM-2 and TM-3. The maximum achieved accuracy results are compared.

or duplicate samples as in the user-specific dataset. In this set of evaluations, we simulate overlapped mined datasets and perform evaluations under the same threat models.

Simulation of TM-2. For the city-level estimation evaluations, we rebuild a simulation dataset with a $30 - 34\%$ overlap ratio for each region within the cities. The same evaluation procedures are then followed as the original mined dataset, which is 10-fold cross-validation with a fixed n-grams size of 5. Figure 3.10(a) and Figure 3.10(b) show the comparison between the best achieved result in original evaluation and the best achieved result in the simulations. The increase in the accuracy confirms our previous hypothesis that having overlapped route samples would increase the accuracy. Since the mined dataset is not specific to any target user's mobility pattern, it is anticipated to result in less accuracy than the TM-1 evaluation accuracy scores.

Simulation of TM-3. For TM-3's simulated evaluations, we rebuild a simulation dataset with a 35% overlap ratio for each city and performed the same evaluation with 10-fold cross-validation and 5-grams. Figure 3.10(c) shows the comparison of the best achieved accuracy results in original evaluations and simulations. As expected, the accuracy is increased in the simulations proving our previous hypothesis that having similar patterns in a dataset affects the success of the attack.

Image-like Data Evaluations. In this set of evaluations, we perform experiments on the image-like representations of the data. Since the original data is unbalanced, the dataset built with the

image-like representation also inherits the problem. In this section, we explain the methods to avoid bias due to unbalanced dataset and discuss the associated results.

Dealing with Unbalanced Dataset. There are various methods to deal with unbalanced dataset, including downsampling, oversampling, and creating synthetic samples from existing ones. Among these methods, downsampling and oversampling are the easiest ones to explore, although downsampling leads to losing a great amount of data, and oversampling raises the chances of getting lower accuracy as the misclassified duplicated samples increase the false ratio. Therefore, we explore other alternatives: (i) weighted loss function and (ii) fine-tuning with different samples.

Weighted Loss Function. For the unbalanced dataset, we utilize a weighted loss function while training the CNN and use all the data in the dataset. By assigning a class weight that is inversely proportional to the sample size of the class, we signify samples of small classes while calculating the loss, thus their effect does not easily wear off.

Fine-Tuning with Different Samples. Fine-tuning is a common technique in deep learning and is used for re-training a complex pretrained model with another dataset. To address the unbalanced dataset, we take advantage of fine-tuning in a different manner. Namely, we introduce rounds and create a set of small datasets from the unbalanced datasets for each round. As illustrated in Fig. 3.11, several small and balanced datasets are created by randomly selecting samples. For each consecutive round, samples of one or more classes are discarded, and the round dataset is created from the remaining classes. After round dataset creation, the model is trained with the round dataset that contains *the least number of classes*, *i.e.* the lattermost created round dataset. At each step, the model is re-trained using the same or different hyperparameters until all the rounds expire. The dataset ordering of the rounds is reversed since the impact of the smallest dataset would wear off if the model is trained with the same order of round dataset creation, which conflicts with the whole idea. As illustrated in Fig. 3.12, while re-training, the parameters of the previous model are passed to the model of the next round. The hyperparameters of each round can be tuned accordingly. For instance, for the last round, where we include all of the classes, the

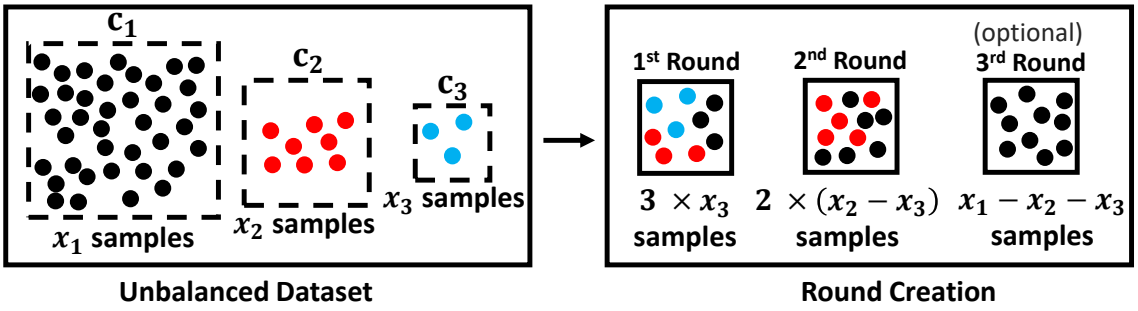


Figure 3.11: An illustration of round creation from an unbalanced dataset of three classes.

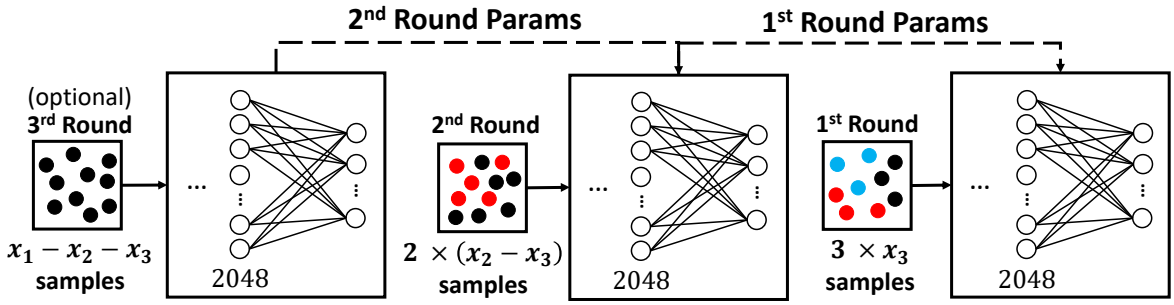


Figure 3.12: An illustration of the fine-tuning pipeline for an unbalanced dataset of three classes.

learning rate is reduced to find the loss minima.

To evaluate our attacks on the image-like data, the elevation profiles are converted into a dataset of images and rounds using the configurations and steps discussed above. Table 3.8 highlights the maximum achieved prediction accuracy along with comparisons with other methods.

Weighted vs. Unweighted Loss Function. To observe the impact of the weighted loss function, we conduct evaluations without giving any weight to the classes in the loss function while using an unbalanced dataset. We note that the unweighted loss function evaluation results are biased due to the unbalanced dataset. Table 3.8 shows the maximum achieved accuracy for each dataset and method. Even though the weighted loss function evaluation results are biased, which *seems* successful in outputting the largest class used during training and testing, the biased results remain behind 4 evaluations out of 8. In TM-1 and TM-3, the accuracy scores of unweighted and weighted

Table 3.8: Comparison of maximum achieved accuracy across different methods. The Unweighted Loss (UWL) column is not considered while deciding the maximum accuracy, as the results are biased. The maximum accuracy of each evaluation is written **bold**, the results that are not considered are written *italic*.

Methods	**Raw**	**Text-like**	**Image-like**		
		n-grams/ tf-idf	UWL *(biased)*	WL	FT
TM-1	96.51	**99.94**	*96.98*	95.23	87.93
TM-2: LA	77.41	**78.02**	*68.85*	68.39	63.63
TM-2: MIA	80.85	**99.54**	*88.96*	86.80	62.50
TM-2: NJ	83.32	**92.42**	*93.45*	79.42	57.14
TM-2: NYC	**84.26**	82.96	*74.20*	79.37	72.79
TM-2: SF	74.66	**80.25**	*67.20*	78.70	65.38
TM-2: WDC	77.30	**89.61**	*62.79*	70.28	71.50
TM-3	**95.36**	89.48	*92.51*	92.82	89.00

Table 3.9: The fine-tuning results for TM-1 and TM-3 as the epoch size changes.

	TM-1			TM-3		
Epoch Size	**500**	**1000**	**2000**	**500**	**1000**	**2000**
Accuracy	79.31	87.96	82.73	86.04	89.00	87.85
Recall	55.87	67.54	63.12	29.76	45.34	38.91
Specificity	86.33	92.65	88.46	92.27	93.98	93.29
F1 Score	58.62	68.25	63.37	36.23	45.45	41.12

loss functions are considerably close. Thus, we conclude that the weighted loss function improved the prediction performance primarily for TM-2.

Fine-tuning vs. Weighted Loss Function. For the fine-tuning evaluations, round datasets are created from the original data. For TM-1, with 4 classes, 3 rounds are created. For TM-3, with 10 classes, 5 rounds were created by eliminating 1, 2, 1, and 2 classes at each round, respectively. The dataset of TM-2 can be considered as a compilation of the dataset of 6 cities: Los Angeles (3 rounds), Miami (3 rounds), New Jersey (2 rounds), New York City (4 rounds), San Francisco (2

Table 3.10: The fine-tuning results for TM-2 as the epoch size is 1000 and learning rate is 0.001 for all rounds.

	LA	**MIA**	**NJ**	**NYC**	**SF**	**WDC**
Accuracy	63.61	62.52	57.13	72.84	65.34	71.55
Recall	28.02	25.66	40.03	18.15	30.76	73.27
Specificity	75.84	75.97	66.75	83.43	76.35	73.22
F1 Score	28.83	28.64	37.55	18.46	31.47	73.44

rounds), and Washington DC (1 round). Even though the main idea is to use all the data we have, we decided to downsample the classes with a large sample size. For instance, in the evaluation of TM-2: New York City, the biggest class has 5,455 samples where the second biggest class has 960 samples. In such cases, we did not create an additional round for only one class as this round would have a strong influence over the predictions, *i.e.* overfitting.

Table 3.8 shows the fine-tuning method outperformed the weighted loss function method only for TM-2: WDC. The difference between the fine-tuning evaluation of Washington DC and others is that we were able to create only one round from the data. Overall, according to the results shown in Table 3.9 and Table 3.10, the fine-tuning evaluation is not as successful as the weighted loss function evaluation, since we still lose some data while creating rounds.

Text-like vs. Image-like Evaluations. When we compare text-like and image-like representations, we can conclude that text-like representation is a better choice for such attack. For all evaluations except TM-3, text-like representation outperformed image-like representation. For TM-3 and TM-2:NYC, the raw data and RF configuration is the best choice.

Countermeasures

Having proven that sharing the elevation profile introduces a threat for users' location privacy, we offer four techniques to bypass this risk while maintaining the key role of elevation profiles, which is demonstrating the roughness, technicality, and difficulty of the routes as a measure of a workout.

Obfuscation. Fitness applications can add an option to obfuscate the elevation profiles. This obfuscation can (i) shift the elevation values up/down or (ii) rotate the signal left/right. For the first option, as the relative elevations will remain the same, the changes will still demonstrate the characteristics of the route. However, for the athletes caring about the oxygen levels associated with the altitude, this solution might be applicable. The second option is most effective to the routes that starts and ends at similar altitudes. When we consider a route that start at low altitude and ends at high altitude rotating the signal right will change the technicality of the signal. The elevation profile will show a strict incline, which is not the case.

Adding Dummy Events. Athletes that are mostly interested in showing the elevation gain rather then the length of the route can insert dummy portions to the flat elevation profile segments to confuse the prediction models. Fitness applications can automatize this process with offering parameters such as, length of dummy events, altitude range of the dummy events etc.

Aggregation. For the athletes that wants to show the exact details of the route and maintain their privacy, fitness applications can offer an option to aggregate the elevation profile. The aggregated version can show the overall elevation range, the elevation gain, length of the route, duration, velocity etc. Additionally, with the help of statistical measures, such as standard deviation, athletes can show the technicality of the routes.

Summary of the Completed Work

In this work, we presented a new inference attack on location privacy using only elevation profiles collected by wearable fitness trackers. The attacks are categorized into three types: predicting location by knowing the activity history of the target, predicting the borough by knowing the city of the target, and predicting the city of the target without any prior knowledge. The key contributions of our work are proving the concept that hiding the route of a workout and sharing only the elevation profile is not sufficient to preserve location privacy, defining a new attack surface by creating scenarios for possible threat models, and providing a machine-learning approach to

realize such threat as attacks. To validate our attacks we created three datasets by collecting data from athletes and mining data from a popular fitness tracking website and Google Elevation API. We preprocessed the datasets by employing Natural Language Processing and Computer Vision approaches and then employed classification techniques to predict the location from elevation profiles. En route, we defined three threat models and evaluated each of them individually on the different datasets. As a result of the evaluations, we are able to identify the corresponding location of an elevation profile with accuracy between 59.59% and 95.83%.

CHAPTER 4: EXPLOITING ACOUSTICS FOR PRIVACY INFERENCE THROUGH MODELLING

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 [28, 34, 93, 98], 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 work explores the attack surface due to a victim's smartwatch, which addresses those shortcomings. Our contribution is SIA, an inference attack that is facilitated

This work has been published in Proceedings of the 20th Workshop on Workshop on Privacy in the Electronic Society (WPES '21) held in conjunction with The ACM Conference on Computer and Communications Security (CCS) 2021.

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

System and Threat Model

In this section, 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 4.1: (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

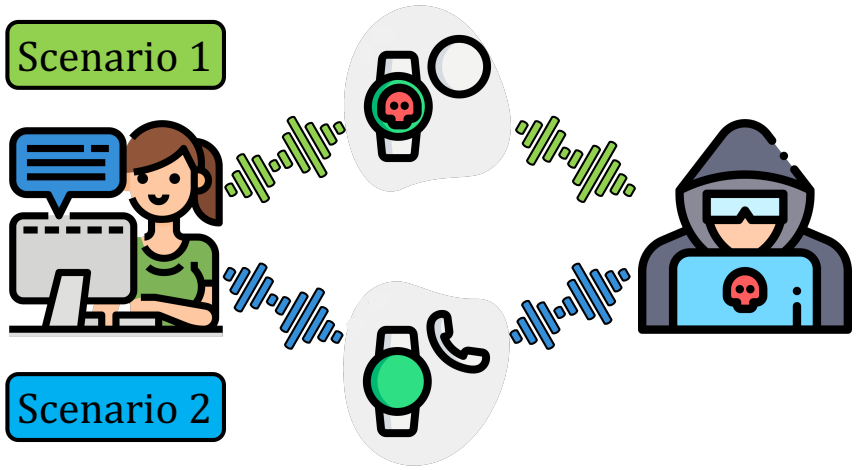


Figure 4.1: System and Threat Model Scenarios. Scenario 1 assumes an infected smartwatch for data transmission, while scenario 2 exploits the acoustic emanations in a phone call.

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 where the targeted user talks through the smartwatch and the adversary records the 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, as follows:

Challenge 1: The smartwatch mobility adds another layer of challenge, impacting the observed signal quality and consistency.

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

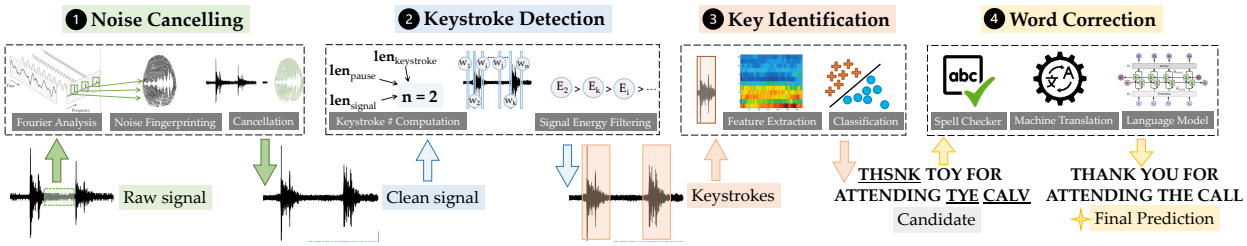


Figure 4.2: SIA consists of four main stages: *Noise Cancelling*, *Keystroke Detection*, *Key Identification*, and *Word Correction*. *Noise Cancelling* takes the raw signal, cancels any ambient background noise or white noise (hiss) from the signal, and returns the clean signal. *Keystroke Detection* 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.

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

Challenge 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, we address those challenges.

Methodology

In this section, we introduce the methodology of SIA, followed for implementing the attack objectives in our threat model. Figure 4.2 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.

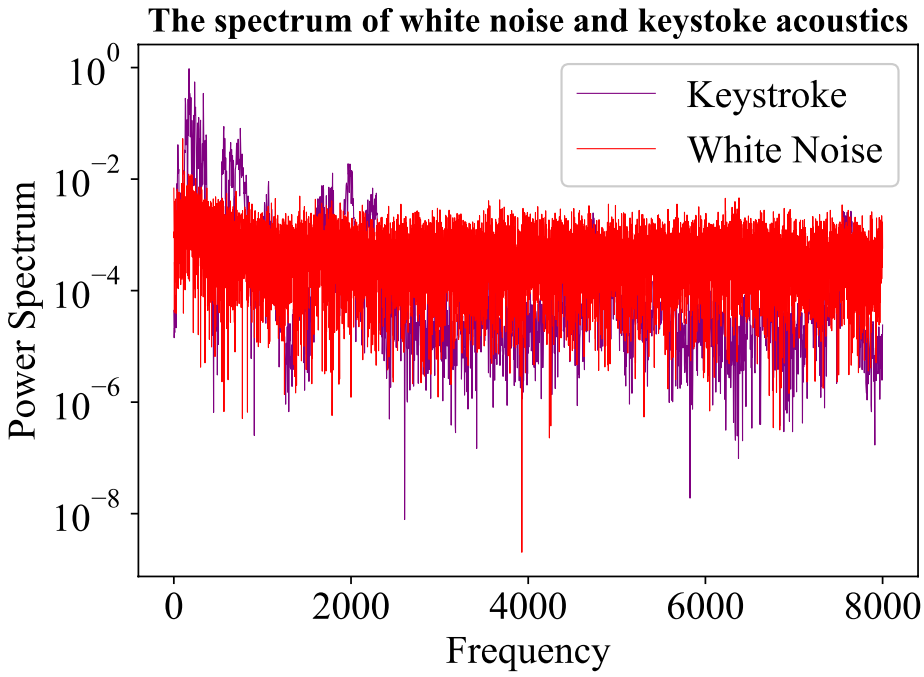


Figure 4.3: The significant frequency band overlap between the white noise and the acoustics of a keystroke event.

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.

White Noise Cancelling. Our white noise cancelling 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 4.3 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.

Ambient Noise Cancelling. Due to the unpredictable nature of ambient noise, i.e., what quantity

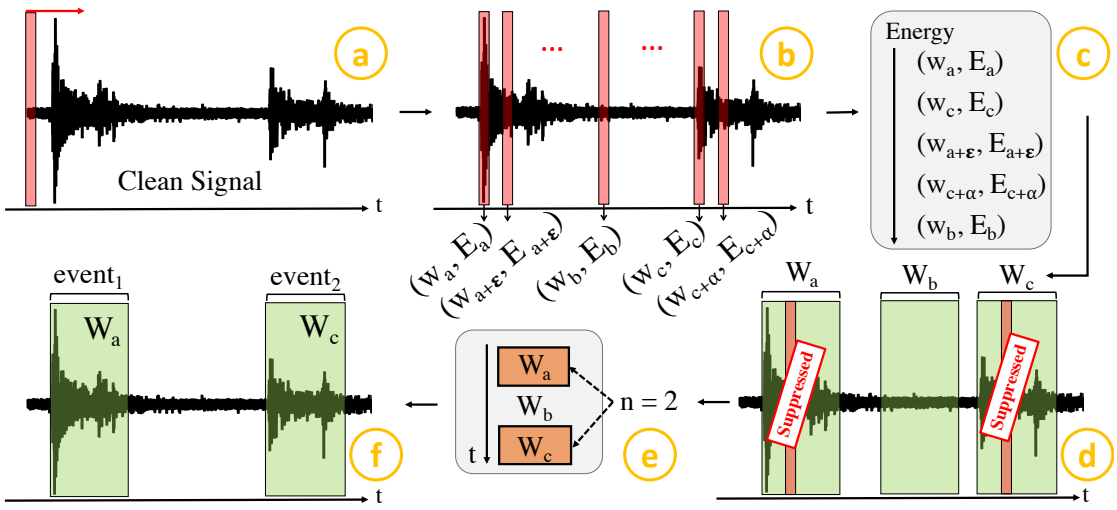


Figure 4.4: The maximum number of keystrokes, n, is computed considering the length of the clean signal coming from *Noise Cancelling* stage. **(a)** A 10 ms window is slid over the clean signal. **(b)** The spectrum energy, E_a, of each 10 ms windows w_a is calculated. **(c)** The window-energy tuples, (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.

of noise (what) and wherein the time domain it is injected (when), cancelling it is more challenging than cancelling 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".

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 [31,45,90,96,133], (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 4.4). 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 4.4). 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 from the `energy_list` (Part d in Figure 4.4). 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.

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.

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

$$C_p = |\mathsf{FFT}(\log(|\mathsf{FFT}(f(x))|^2))|^2. \tag{4.1}$$

The inner FFT function in Equation 4.1 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. 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.

Table 4.1: 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

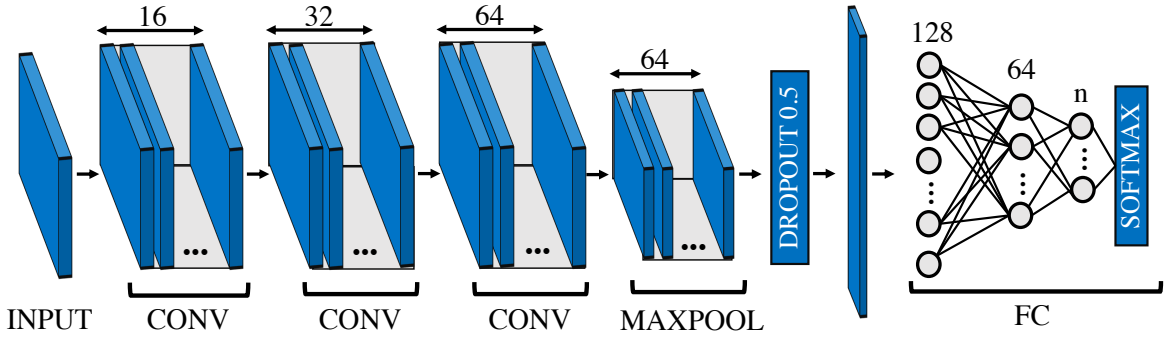


Figure 4.5: The Convolutional Neural Network (CNN) architecture employed during the classifier selection. Three subsequent convolutional layers (*CONV*) with the depth of 16, 32, 64 are utilized. Then, a *MAXPOOL* layer selects the maximum weights in the convolutions. Before the fully-connected layer (*FC*), a dropout regularization is performed to avoid overfitting. The softmax at the end of *FC* returns the class probabilities for the given input.

> 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 4.1).

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 4.2).

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 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 4.5. 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

Table 4.2: 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

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 [110] 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 [66] 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:

$$f(\hat{y})_i = \frac{e^{\hat{y}_i}}{\sum_j^C e^{\hat{y}_j}} \quad CCE = -\sum_i^C y_i \log(f(\hat{y})_i), \tag{4.2}$$

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 4.2) 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 of an advantage for our task. Therefore, CNN

could not perform better when compared to others.

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.

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 [97]. Levenshtein distance (LD) [74] 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.

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` [58] 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.

Next Word Prediction: 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 [102] 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.

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.

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, Apple Watch Series 6, 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 2021.

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 smartwatches 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.

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

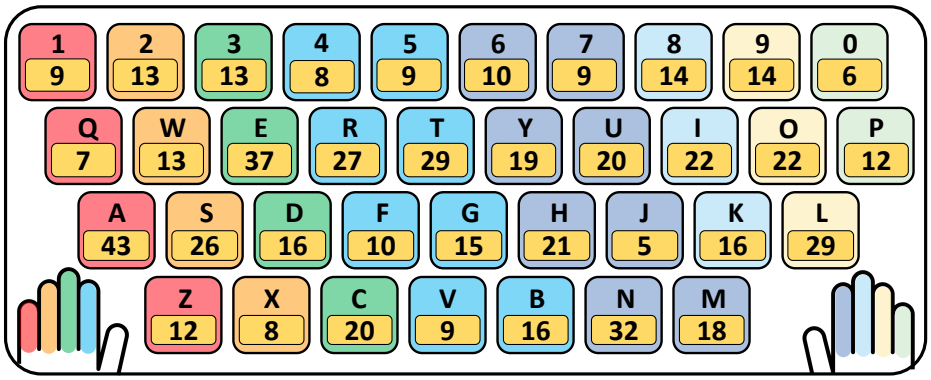


Figure 4.6: The cumulative character-wise distribution of the test datasets. The colors on the keys encodes the associated hand and finger.

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.

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.

- **E-mail:** A randomly selected e-mail sample from Enron mail dataset [67]. The e-mail consists of 213 letters and 42 words.
- **Random Password:** 32 randomly generated passwords of length 8. Hereafter, this data is referred to as “Random”.
- **Selected Password:** 20 randomly selected passwords from the RockYou password dataset. Hereafter, this type of data is referred to as “Selected”.

Figure 4.6 shows the character-wise distribution of the characters in the test dataset.

Table 4.3: 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.

Reference	Method	5-fold	Practical
SIA	Clean Data + SVM	0.99	0.85
Compagno et al. [45]	Noisy Data + LR	0.94	0.53

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*. The experiments on the USER B's test data are considered as *practical attack*.

Unlike in the prior studies [45] 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. [45] collected a single dataset and directly evaluated the performance using k-fold cross-validation without noise cancelling. For comparison, we implemented their method and tested our data on it. Table 4.3 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 4.3). 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 4.3 also demonstrates that SIA outperforms Compagno et al. [45] (0.85 for SIA vs 0.53 for their model). We emphasize that our better results in comparison with [45] are obtained in a more practical system setting: Compagno et al. [45] 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 4.4: Keystroke detection results where 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 (Samsung Galaxy Watch Active 2 / Apple Watch Series 6 / Samsung Galaxy s10e), and target keyboard (MBA / BMK) combinations.

Keyboard (Recorder)	**Metric**	**E-mail**	**Random**	**Selected**
MBA (Samsung Galaxy Watch Active 2)	TPR	0.994	0.996	0.994
	FNR	0.006	0.004	0.006
	P	0.900	0.988	0.900
MBA (Apple Watch Series 6)	TPR	0.995	0.993	0.994
	FNR	0.005	0.007	0.006
	P	0.900	0.908	0.900
MBA (Samsung Galaxy s10e)	TPR	0.994	0.988	0.988
	FNR	0.006	0.012	0.012
	P	0.855	0.941	0.895
BMK (Samsung Galaxy Watch Active 2)	TPR	0.965	0.984	0.983
	FNR	0.035	0.016	0.017
	P	0.754	0.933	0.818
BMK (Apple Watch Series 6)	TPR	0.985	0.974	0.980
	FNR	0.015	0.026	0.020
	P	0.754	0.943	0.907
BMK (Samsung Galaxy s10e)	TPR	0.994	0.996	0.967
	FNR	0.006	0.004	0.033
	P	0.855	0.910	0.931

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.

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:

$$\textit{IoU}(E_1, E_2) = \frac{\min(0, t_2 - t_3)}{(t_4 - t_1)} \in [0, 1]. \tag{4.3}$$

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 4.4 shows the TPR, False Negative Rate (FNR), and Precision (P) of the *Keystroke Detection* on each dataset-keyboard-recorder combinations.

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 facilitate 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.

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 cross-entropy and NLD evaluation metrics.

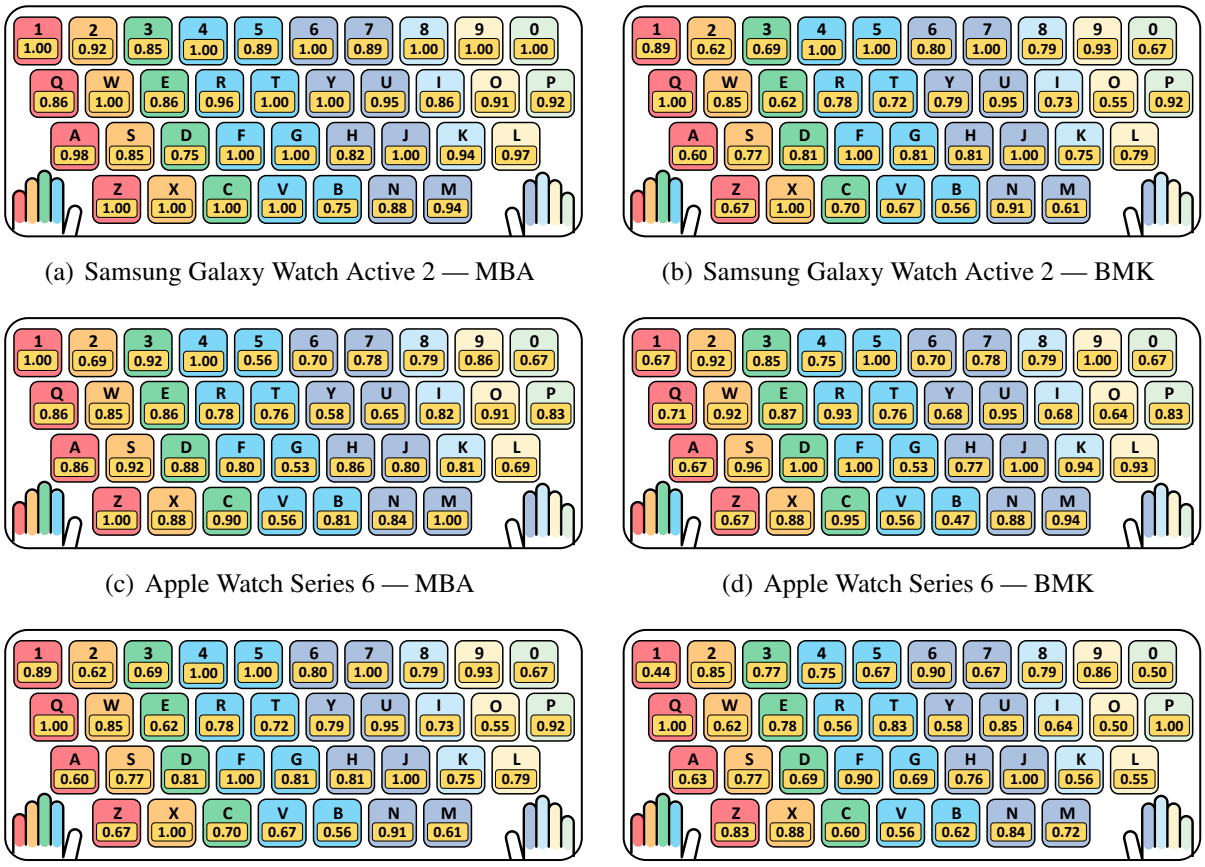
Cross-entropy: The cross-entropy is a measure of the difference between two probability distributions for a given random variable. To evaluate the performance of our classifier, we used the cross-entropy of the predicted probability distribution, f, relative to the actual distribution, p. The cross-entropy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

$$H(p, f) = -\sum_{i} p(x_i) \log(f(x_i) + \epsilon) \tag{4.4}$$

The ϵ value in (4.4) 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.

Character-wise TPR: Figure 4.7 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



(a) Samsung Galaxy Watch Active 2 — MBA

(b) Samsung Galaxy Watch Active 2 — BMK

(c) Apple Watch Series 6 — MBA

(d) Apple Watch Series 6 — BMK

(e) Samsung Galaxy s10e — MBA

(f) Samsung Galaxy s10e — BMK

Figure 4.7: 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.

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 computed.

Figure 4.7(a) shows the character-wise TPR for the MBA keyboard and Samsung Galaxy Watch Active 2. 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 4.7(a) 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

Table 4.5: 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.

	MBA	**BMK**
Most Confused	67895	104834
Never Confused	54219	80835

average of 0.86.

Figure 4.7(c) shows the character-wise TPR for the MBA keyboard and Apple Watch Series 6. The TPR values range from 0.53 to as high as 1.00. The most successful predictions are done on the keys pressed with the left little finger with an average of 0.93. The least successful predictions are recorded with the right little finger with an average of 0.75.

Figure 4.7(b) shows the character-wise TPR for the BMK keyboard recorded by the smartwatch. The results range from 0.55 to 1.00. 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.

The character-wise TPR for the BMK keyboard and Apple Watch Series 6 is shown in Figure 4.7(d). The TPR results range from 0.47 to 1.00. The highest TPR is observed with left middle and ring finger with an average of 0.92.

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-

Table 4.6: Cross entropy shows the difference between two probability distribution. **Ideal** column is the cross entropy of the actual distribution (one-hot vector) with itself ($\epsilon = -1 \times 10^{-9}$). **Frequency** column is the cross entropy between the actual distribution and the frequency distribution associated with each dataset (E-mail $\rightarrow$ English letter frequency, Random $\rightarrow$ Uniform distribution, Selected $\rightarrow$ Character distribution of RockYou leaked passwords) SIA column shows the cross entropy between the actual distribution and the distribution estimated by our method.

Keyboard (Recorder)	**Dataset**	**Ideal**	**Frequency**	SIA
MBA (Samsung Galaxy Watch Active 2)	E-mail	ϵ	39.77	0.66
	Random	ϵ	3.58	0.40
	Selected	ϵ	18.02	0.86
MBA (Apple Watch Series 6)	E-mail	ϵ	39.77	0.48
	Random	ϵ	3.58	0.98
	Selected	ϵ	18.02	0.92
MBA (Samsung Galaxy s10e)	E-mail	ϵ	39.77	0.65
	Random	ϵ	3.58	0.33
	Selected	ϵ	18.02	0.80
BMK (Samsung Galaxy Watch Active 2)	E-mail	ϵ	39.77	0.88
	Random	ϵ	3.58	1.20
	Selected	ϵ	18.02	0.96
BMK (Apple Watch Series 6)	E-mail	ϵ	39.77	0.73
	Random	ϵ	3.58	0.99
	Selected	ϵ	18.02	0.80
BMK (Samsung Galaxy s10e)	E-mail	ϵ	39.77	0.41
	Random	ϵ	3.58	1.79
	Selected	ϵ	18.02	1.10

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 4.5). 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.

Smartwatch vs. Smartphone: To compare the effect of the mobility introduced with the smart-

watches, 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 4.6). The perfect match with the actual distribution ("Ideal" column in Table 4.6) is a very small number ϵ. The **Frequency** column in Table 4.6 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 4.7 shows the TPR, TPR with one-hop, NLD, and NLD with one-hop for each dataset,

Table 4.7: String-wise evaluation results of Key Identification for all device-keyboard-dataset combinations. The predictions obtained from Key Identification stage are compared with the ground truth strings.

Keyboard (Recorder)	**Dataset**	**TPR**	**TPR** (1-hop)	**NLD**	**NLD** (1-hop)
MBA (Samsung Galaxy Watch Active 2)	E-mail	0.918	0.988	0.065	0.009
	Random	0.972	0.992	0.025	0.007
	Selected	0.878	0.939	0.110	0.055
MBA (Apple Watch Series 6)	E-mail	0.750	0.889	0.201	0.089
	Random	0.886	0.968	0.107	0.029
	Selected	0.773	0.905	0.205	0.090
MBA (Samsung Galaxy s10e)	E-mail	0.901	0.971	0.079	0.023
	Random	0.980	1.000	0.018	0.000
	Selected	0.933	0.966	0.060	0.030
BMK (Samsung Galaxy Watch Active 2)	E-mail	0.796	0.918	0.164	0.065
	Random	0.812	0.921	0.177	0.073
	Selected	0.679	0.834	0.290	0.150
BMK (Apple Watch Series 6)	E-mail	0.750	0.906	0.201	0.075
	Random	0.878	0.965	0.114	0.033
	Selected	0.817	0.928	0.165	0.065
BMK (Samsung Galaxy s10e)	E-mail	0.901	0.976	0.798	0.018
	Random	0.777	0.929	0.210	0.066
	Selected	0.442	0.712	0.505	0.260

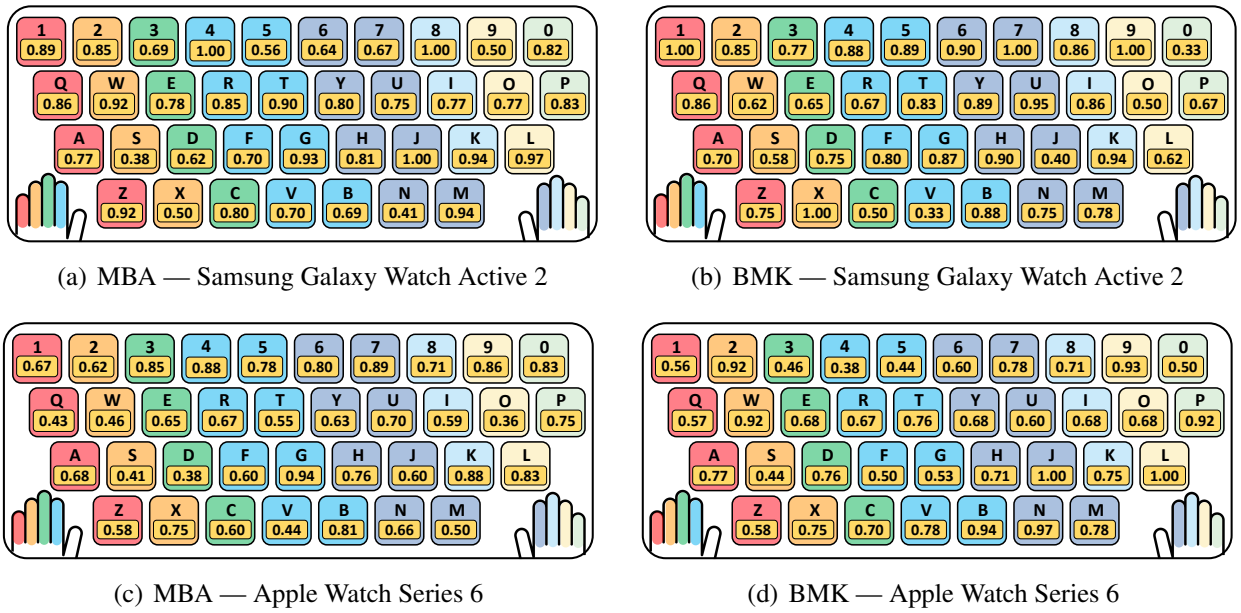
device, and keyboard model. The choice of one-hop is not arbitrary. During our experiments, we observed that some of the mispredicted letters are within the one-hop distance with the actual key (on the keyboard layout). This is reasonable since the acoustic emanations coming from keys within close proximity 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 7.8%, where that of the BMK is 20.1%. 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.

Table 4.8: 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.

Keyboard (Recorder)	**Method**	**TPR**	**TPR** (1-hop)	**NLD**	**NLD** (1-hop)
MBA (Samsung Galaxy Watch Active 2)	None	0.918	0.988	0.065	0.009
	SSC	**0.970**	0.988	0.023	0.009
	MT	0.953	0.994	0.037	0.004
	NW	0.866	0.936	0.112	0.053
MBA (Apple Watch Series 6)	None	0.750	0.889	0.201	0.089
	SSC	**0.936**	0.982	0.051	0.014
	MT	0.831	0.924	0.136	0.061
	NW	0.924	0.971	0.061	0.023
MBA (Samsung Galaxy s10e)	None	0.901	0.971	0.079	0.023
	SSC	0.959	0.988	0.033	0.009
	MT	**0.970**	0.994	0.023	0.004
	NW	0.912	0.965	0.070	0.028
BMK (Samsung Galaxy Watch Active 2)	None	0.796	0.918	0.164	0.065
	SSC	0.877	0.930	0.023	0.009
	MT	**0.970**	0.988	0.023	0.009
	NW	0.872	0.953	0.103	0.037
BMK (Apple Watch Series 6)	None	0.750	0.906	0.201	0.075
	SSC	0.895	0.970	0.084	0.023
	MT	0.936	0.976	0.084	0.023
	NW	**0.970**	0.976	0.023	0.018
BMK (Samsung Galaxy s10e)	None	0.901	0.976	0.798	0.018
	SSC	0.965	0.971	0.028	0.023
	MT	**0.970**	0.988	0.028	0.023
	NW	0.918	0.982	0.065	0.014

Word Correction. We used the three methods for word correction. *Word Correction* stage is only applicable for the E-mail dataset since other datasets do not include actual English words. Table 4.8 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 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.



(a) MBA — Samsung Galaxy Watch Active 2

(b) BMK — Samsung Galaxy Watch Active 2

(c) MBA — Apple Watch Series 6

(d) BMK — Apple Watch Series 6

Figure 4.8: The character-wise evaluation results for the practical attack.

Practical Attack. In this section, 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 cancelled, 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 User Profiling section are utilized.

Table 4.9: The cross-entropy changes for the practical attack.

Keyboard (Recorder)	**Dataset**	**Ideal**	**Frequency**	SIA
MBA (Samsung Galaxy Watch Active 2)	E-mail	ϵ	39.77	1.19
	Random	ϵ	3.58	1.29
	Selected	ϵ	18.02	1.09
MBA (Apple Watch Series 6)	E-mail	ϵ	39.77	2.15
	Random	ϵ	3.58	2.25
	Selected	ϵ	18.02	2.55
BMK (Samsung Galaxy Watch Active 2)	E-mail	ϵ	39.77	1.29
	Random	ϵ	3.58	1.12
	Selected	ϵ	18.02	1.18
BMK (Apple Watch Series 6)	E-mail	ϵ	39.77	1.91
	Random	ϵ	3.58	1.91
	Selected	ϵ	18.02	2.06

Keystroke Detection. The performance of the *Keystroke Detection* for the practical instance of SIA was almost identical with the results provided earlier in Table 4.4— we achieved about 0.98 in terms of average TPR. Therefore, we did not include another table for the *Keystroke Detection* evaluations.

Key Identification & Word Correction. Figure 4.8 demonstrates the character-wise TPR for the practical attack. When we compare the results in Figure 4.7 and Figure 4.8, we observe a slight decrease in the overall TPRs for the BMK ($0.798 \rightarrow 0.764$) and a relatively large decrease for the MBA ($0.937 \rightarrow 0.776$). The difference between the unique typing styles of the subjects leads to a decrease in the detection performance. Table 4.9 shows the cross-entropy decreases through the practical attack. Although the decrease is not as much as in Table 4.6, 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 4.10 shows the string-wise evaluations for each dataset—after *Key Identification*. Table 4.11 shows the string-wise evaluation results for each word correction method on the E-mail dataset.

Table 4.10: String-wise evaluation results of Key Identification for all keyboard-dataset combinations. The predictions obtained from Key Identification stage of the practical attack are compared with the ground truth strings.

Keyboard (Recorder)	**Dataset**	**TPR**	**TPR** (1-hop)	**NLD**	**NLD** (1-hop)
MBA (Samsung Galaxy Watch Active 2)	E-mail	0.761	0.930	0.192	0.056
	Random	0.769	0.921	0.217	0.073
	Selected	0.762	0.933	0.215	0.060
MBA (Apple Watch Series 6)	E-mail	0.686	0.767	0.253	0.187
	Random	0.675	0.777	0.306	0.210
	Selected	0.607	0.657	0.355	0.310
BMK (Samsung Galaxy Watch Active 2)	E-mail	0.680	0.877	0.258	0.098
	Random	0.792	0.949	0.195	0.047
	Selected	0.758	0.912	0.210	0.070
BMK (Apple Watch Series 6)	E-mail	0.697	0.831	0.244	0.136
	Random	0.758	0.859	0.225	0.129
	Selected	0.696	0.784	0.275	0.195

When comparing Table 4.10 and Table 4.11 with Table 4.7 and Table 4.8 respectively, 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.

Countermeasures

Having shown that SIA can successfully infer what a victim user is typing on a physical keyboard, we outline some possible countermeasures for the attack scenarios SIA covered.

Dynamic Access Control for Smartwatches. The main assumption for Scenario 1 is having an infected smartwatch that records the acoustics. To prevent a malicious software in smartwatches

Table 4.11: String-wise evaluation results of Word Correction on E-mail dataset for all keyboard-method combinations for practical attack. The output of Word Correction stage is compared with the ground truth strings.

Keyboard (Recorder)	**Method**	**TPR**	**TPR** (1-hop)	**NLD**	**NLD** (1-hop)
MBA (Samsung Galaxy Watch Active 2)	None	0.761	0.930	0.192	0.056
	SSC	0.848	0.959	0.122	0.032
	MT	0.819	0.953	0.145	0.037
	NW	0.837	0.947	0.131	0.042
MBA (Apple Watch Series 6)	None	0.686	0.767	0.253	0.187
	SSC	0.848	0.895	0.122	0.084
	MT	0.901	0.930	0.079	0.056
	NW	0.884	0.919	0.093	0.065
BMK (Samsung Galaxy Watch Active 2)	None	0.680	0.877	0.258	0.098
	SSC	0.790	0.918	0.169	0.066
	MT	0.709	0.883	0.234	0.093
	NW	0.734	0.878	0.214	0.098
BMK (Apple Watch Series 6)	None	0.697	0.831	0.244	0.136
	SSC	0.796	0.866	0.164	0.107
	MT	0.784	0.854	0.173	0.117
	NW	0.918	0.936	0.066	0.051

to constantly recording, smartwatches can support dynamic access control. Dynamic access control can check which application accesses which peripheral and notify the user for the suspicious activities. Although this might not be applicable for current wearables due to battery constraints, the decision mechanism can be offloaded to the smartphone that the smartwatch is connected.

Dynamic Internet Connection Control. Another assumption of Scenario 1 is that smartwatch sends the recordings to the adversary via the Internet connection. The Internet connectivity of the smartwatch applications can be monitored and in case of a suspicious activity, such as establishing a connection with an uncertified server, the user can be notified.

Keys Emanating Homomorphic Acoustics. A more generic defense mechanism for the users would be preferring keyboards emanating similar acoustics for each key. There has been some

studies proposing such designs in the literature [31]. Having a uniform noise for all keys would invalidate the inference attacks modelling the acoustic emanations.

Creating Synthetic Noise. Another way to prevent the keylogging attacks facilitated by acoustic emanations is concealing the actual acoustic emanations. Abhishek *et al.* [30] proposed emitting certain background sounds through a speaker. They showed that, these sounds mask the actual keystroke emanations and reduces the effect of keylogging attacks. However, this method decreased the usability and probably would not be applicable for Scenario 2, as the victim user will turn off the background noise during a phone call. Additionally, they considered white noise as a candidate for background noise, but as we are eliminating white noise, it would not be a candidate countermeasure for SIA.

Instead, we propose different kind of white noise that introduces some sense of randomization. Such randomized nature of the background noise would strictly reduce the effectiveness of noise cancelling and inherently affects the performance of the key identification.

Raising Awareness for the Users. Raising the awareness would be the most effective countermeasure for such attacks. Smartwatches can detect the typing activity and send a notification that warns the user to avoid phone calls engaged through smartwatch. Such notification can also list the applications utilizing the microphones to further draw attention.

Summary of the Completed Work

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.

CHAPTER 5: EXPLOITING GEOMETRIC PROJECTIONS FOR PRIVACY INFERENCE THROUGH MAPPINGS

Despite limitless benefits that AR/VR devices offer to consumers, new security and privacy risks emerge from the fundamentally new interaction methods. As AR/VR devices get closer to end-users, the community has put efforts into identifying and remedy potential risks. While HMD devices are considered more secure because only the user can see the contents [79], recent work [69, 72, 80, 91, 106, 126] showed they bring risks in other respects.

The interaction methods with immersive platforms differ from the conventional methods on computers and other smart devices by shifting towards more natural methods to offer a more convincing perception [14, 57, 127–129, 131]. These methods include interaction through voice commands, hand gestures, head movements, IMU rings, and remote controllers. For text entry, Microsoft HoloLens uses head movements and hand gestures, Microsoft Hololens 2 uses air-tapping, and Magic Leap 1 uses a remote controller to write on a virtual keyboard with a conventional layout shown only to users.

Such novel input methods draw apart from their alternatives by not exposing the input flat. However, recent work [69, 80] shows that these methods are still vulnerable to inference attacks. Sun et al. [112] present a method for recognizing in-air tapping on low latency AR/VR devices using the on-device sensors. As a use case, they evaluated *DolphinBoard*, an in-air tapping text input method, through a user study and found that the typing speed doubles when compared to the Microsoft HoloLens system keyboard, and the participants show a strongly positive attitude towards DolphinBoard. Recently, after Microsoft released HoloLens 2, Microsoft started to support air-typing through hand trackers embedded in the AR HMDs. Such developments in academia and industry demonstrate that more natural interactions with the AR environment are up-and-coming.

This study presents and evaluates a keylogging inference attack on in-air tapping input entry meth-

This work has been published in IEEE Conference on Virtual Reality + 3D User Interfaces (IEEE VR) 2022.

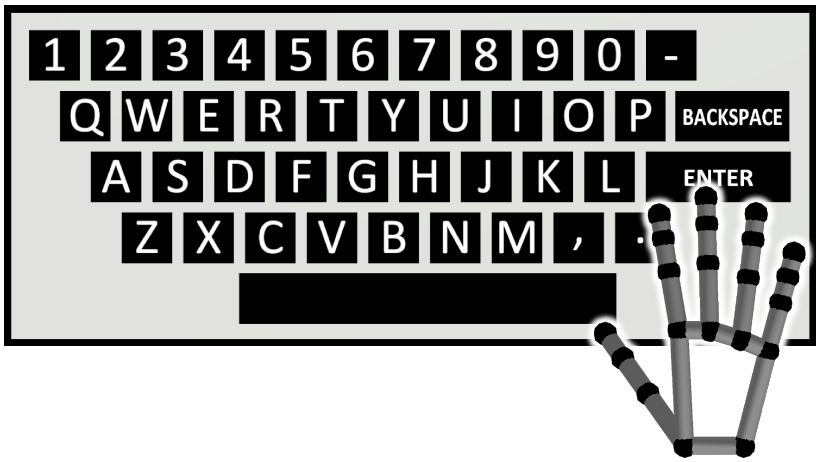


Figure 5.1: First person view of the AR keyboard and hand meshes operated by the user from the Magic Leap 1.

ods by exploiting the geometric projections of the hands. Our attack leverages the observation that hands follow specific patterns when tapping keys to achieve keystroke detection. By localizing the keystrokes in time and space and interpreting the task as a temporal localization problem, we rigorously explore possible keyboard reconstructions from the key tap points and develop heuristics to achieve best to worst ordering of the reconstructions. Our method avoids strong assumptions on the positioning and size of the keyboard to account for AR/VR users' freedom in configuring the placement and the scale of virtual objects, which is an essential feature of AR/VR applications.

As we do not have direct access to in-air tapping keyboards in the wild, we developed our own keyboard–AiRType for this work and performed the proposed inference attack on AiRType.

The research community envisions that the AR/VR technology will invade many new domains and will be used by a large number of users ubiquitously, which makes understanding its potential vulnerabilities both timely and critical.

System and Threat Models

System Model. "Virtual keyboard" is broadly used to cover different keyboard types, including

on-screen keyboards driven with a mouse or by tapping on the touchscreen. Our study develops a virtual keyboard for AR/VR HMDs and uses it by hand gestures, mimicking how keys on a physical keyboard are tapped. We refer to our keyboard design as *AR keyboard*. Although the keyboard design and our study also apply to VR, we mainly pursue AR environments, specifically AR HMDs, which is essential in outlining our threat models while covering the VR context.

AiRType – AR Keyboard Design: In our keyboard design, the user attaches the AR keyboard to some arbitrary location in the virtual environment, and the AR HMD tracks the user's hands by rendering models of the hands in the virtual environment. The AR HMD detects the keystrokes by checking if a fingertip collides with a key in space. Figure 5.1 shows a first-person view of AiRType and the right-hand mesh from Magic Leap 1. The keyboard size is configured considering the visibility of all keys on the keyboard within the view range of Magic Leap 1, and the size of hand meshes is set to match the user's hand size. The keyboard provides audio-visual feedback to compensate for the lack of tactile feedback in the keyboard. When a key is tapped, a *click* sound is played, the key moves forward as if pushed, the key color turns into green for a short period, then the key returns to its original position gradually, imitating a conventional physical keyboard. The implementation and design details of AiRType is elaborated in AiRType: AR Keyboard Design section.

Key Tap. A keystroke is defined as "a single depression of a key on a keyboard" [17], which happens at a particular time instance. We define a **key tap** as a tapping gesture covering the action of a fingertip reaching a key, pressing the key causing a keystroke, and springing back to release the key. Unlike keystroke, a key tap takes time and can be defined using the start and end times. Different from Leap Motion's definition [18], which is similar to air tap gesture on Microsoft HoloLens [15], the finger undertakes the selection and targeting. Thus, we could label a key tap action with the start, keystroke moment, and finish. Two key taps are temporally disjointed by design, and key tap actions are represented as non-overlapping time windows.

(a) **Scenario 1:** The adversary plants a hand tracker device and records the victim's hand movements.

(b) **Scenario 2:** The adversary sits near the victim and records his hand movements using AR HMD.

(c) **Scenario 3:** The adversary infects the victim's AR HMD and remotely obtains the tracking data.

Figure 5.2: Illustrations of the attack scenarios with varying adversarial capabilities.

Session. We define a **session** as the process of writing a text using the AR keyboard where it is fixed in a user-selected position and scale. We assume the keyboard will have a fixed spatial configuration during a session, although it may change between sessions.

Tracking. Hand tracking sensors provide spatial information (i.e., positions, angles) regarding hands and fingers. For users to type on the AR keyboard and for the adversary to carry out the attack, hand tracking data is essential. For users, hand tracking capabilities are built-in in AR HMD, which is more practical in [14, 112]. For the adversary, the hand tracking data comes from one of a few possible mediums. Each of those mediums provides the following information: the tip positions of all fingers, the pointing directions of all fingertips, the position of the palm center, and the palm normal. We refer to this information as the **low-level hand tracking data** and simulate cases where such data is needed using the Leap Motion Controller. We expect our approach to work with minor modifications even when the low-level hand tracking data is not available precisely in the same format but with the spatial information of the hands.

Threat Model. Our threat model considers a victim using the AR keyboard in AR HMD, where the adversary's goal is to infer what the victim types using the hand traces of the victim. To achieve this goal, the adversary gains access to the victim's hand traces through one of a few possible mediums, each of which is covered in an attack scenario. Other than differences in the method

used for collecting the hand tracking data, the three scenarios, depicted in Figure 5.2, share the same procedure from the adversary's perspective.

Scenario 1. In this scenario, the adversary plants a hand tracker device near a victim in a public space, e.g., a coffee shop, and records his hand movements as he types in the AR keyboard. An example hand tracker device that the adversary might use is the Leap Motion Controller [25], which is small (8x3x1.1cm) and can be easily hidden for a stealthy attack. Although the off-the-shelf version of the controller needs to be tethered with a cable to a PC, wireless tethering systems are available [13] and could be utilized to reduce the cable connection burden the attack stealthier.

Scenario 2. In this scenario, the adversary wears an AR HMD device and sits near the victim, possibly in a public space, to record the hand movements of the victim using the hand tracking feature of the adversary's AR glasses. The attack assumes a ubiquitous use of AR HMD, making it challenging to identify the adversary since no unusual hardware is used. For this attack to succeed, the adversary needs to be close to the victim, where the attack is much easier to carry out in context, e.g., two acquaintances in the same space, where one of them might be curious what the other is typing.

Scenario 3. In this scenario, the adversary infects the AR glasses of the victim with malware, enabling the adversary to seamlessly read the data from the hand tracking sensor of the victim's AR glasses through the API of the AR glasses. One advantage of this scenario is that there is no assumption of where the victim is located or the adversary's proximity to the victim. As such, the attack could be conducted remotely, at any time, from anywhere, without needing to deploy any additional device near the victim. We should note that obtaining the sensor readings is a more straightforward attack than reading the processed outputs of the keyboard application. Malware can obtain the sensor readings using the device API stealthily; however, reading the keyboard's processed output while running a separate task requires more sophisticated malware. For example, assuming the gyroscope readings are available to the

attacker does not mean that the attacker can directly read the keyboard's output.

We should note that the adversary does not know the position of the AR keyboard in the victim's augmented reality environment since the users can change the position however they would like. Therefore, for all adversarial scenarios, the sensor readings cannot enable the adversary to directly reach the keyboard output by simply forwarding the sensor readings to the keyboard application. In all scenarios, calibration between the coordinate system of the victim's virtual environment and the coordinate system of the virtual environment model of the adversary is needed. This requirement is one of the main challenges we overcome in this study.

AiRType: AR Keyboard Design

AiRType is an in-air tapping keyboard that supports ten-finger typing experience. AiRType is specifically designed for augmented and virtual reality HMDs, such as Magic Leap 1, Microsoft Hololens, etc. AiRType renders the keyboard plane and two hand models as virtual objects in the augmented environment. The targeting and selection of the keys in AiRType are done through the hand models that mirror the user's hands via hand tracking feature of AR HMDs.

In designing AiRType, we utilized the conventional US keyboard as our default layout. Figure 5.3(a) illustrates the default ratio of the keyboard size with the field-of-view of the AR HMD. Although the position and the scale of the keyboard plane might change, we set the default depth of the keyboard as one meter. In the default setup, we also set the size of the keyboard to 40 cm x 16.8 cm, and the size of the keys alphanumerical keys to 2.5 cm x 2.5 cm.

Augmented reality environments allow users to freely change the size of the virtual objects. As such, we designed our keyboard to be mobile. Even though the actual size of the keyboard does not change in this configuration, users can push the keyboard and hand models further to make them smaller in the augmented environment. Figure 5.3(b) depicts an example of this use case. The user here increases the keyboard depth with respect to the AR HMD to make it look smaller and pin it

(a) **Default Setup.** Increasing the field-of-view.

(b) **Flexible Setup.** Maximizing the field-of-view.

(c) **Overlapping Setup.** More immersive user experience.

Figure 5.3: Illustrations of possible positioning options of AiRType in the augmented reality environment.

to a position where it does not block any significant object in the real-world, e.g., the empty space of the table.

Another configuration might be directly overlapping the hand models with the hands of the user for more immersive experience. In this setting, the fingertips of the user directly touch the keys in the augmented environment. However, as illustrated in Figure 5.3(c), this requires us to design a large keyboard that occludes most of the real-world view. Irrespective of the decrease in the field-of-view, our design does not restrict the user's freedom of scaling, as users are free to scale the AR objects in the virtual environment, and making the keyboard too small or too large has a detrimental effect on the usability. A smaller keyboard makes it difficult to pick the correct key, while a larger keyboard increases the distance the hands have to travel to touch the keys. Moreover, with a too large keyboard, the keyboard may not fit into the AR HMD frame.

AiRType Implementation Details. To deploy AiRType for Magic Leap 1, we developed and built a Unity3D application for Lumin OS, which is the custom operating system of Magic Leap 1. The application consists of the keyboard, hand models, and an invisible controller.

Keyboard. The keyboard object consists of 42 cubes, each of which is an embedded collision interface. The collision interface defines the actions triggered by the collision of fingertips with the keys. To compensate for the lack of tactile feedback in the AR environment, AiRType provides an audiovisual feedback. When a collision is detected on a cube, i.e., a user presses a key, the cube

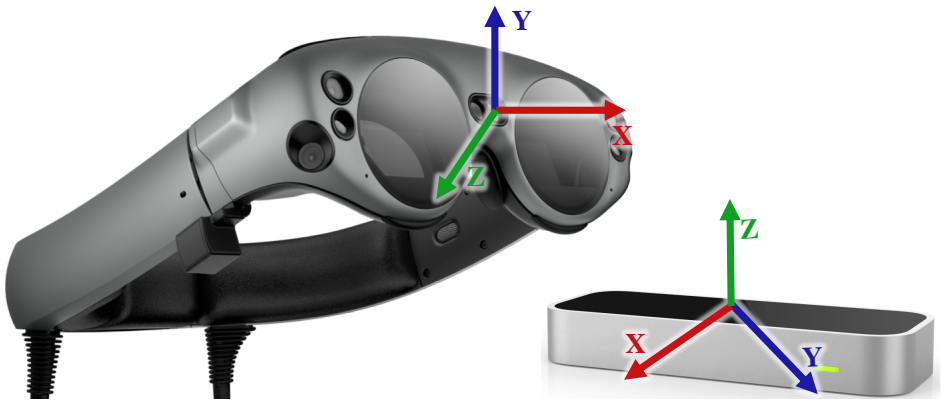


Figure 5.4: The position and orientation of coordinate systems of Magic Leap 1 (left) and Leap Motion Controller (right) is different.

travels in, becomes green, play a "click" sound, put the associated character into the text box, and returns back to its original position, sequentially.

Hand Models. Hand models are designed to be driven with the (x, y, z) coordinates or certain points on hand, such as the fingertips, palm center, finger joints, etc., which come from the hand tracking sensor. In the first version of the AiRType for Magic Leap 1, we utilized the hand tracking information coming from the built-in hand tracker sensor. However, unlike Hololens 2, the current hand tracking sensor of Magic Leap 1 does not provide a steady and sensitive hand tracking information. To be able to successfully mirror the user hands in the AR environment, we needed more robust hand tracking data. Therefore, and to facilitate a fair comparison and evaluation, instead of using the data coming form the built-in hand trackers, we used the data obtained from the Leap Motion Controller (LMC), a small sensor (7.6 cm x 3.0 cm x 1.2 cm) that is specially designed to capture the hand features, i.e., positions of the fingertips, palm center, and joints with respect to its own coordinate system.

Since the coordinate systems of Magic Leap 1 and LMC are out of sync (Figure 5.4), integrating LMC required a calibration step. In this step, we obtained the transformation between the coordinate systems using the reference points that are common in both spaces–fingertip positions. Although Magic Leap 1 cannot provide a steady data stream, it can correctly detect the fingertip

positions with respect to its own coordinate system. In the calibration step, we fetch the fingertip positions from both devices and calculate a transformation matrix that transforms the LMC coordinate system to the Magic Leap 1's. To compute this transformation matrix, we feed the point clouds we obtained from both spaces to an Iterative Closest Point (ICP) algorithm. After the calibration step is performed, the data stream coming from LMC is transformed to the Magic Leap 1's coordinate system and the hand models are rendered on the transformed positions to mirror the user hands in the AR environment.

Due to the calibration step, the placement of LMC is flexible, where users can attach it to the glasses, put it on a table, or wear it on their body for a more convenient use. We note that it is expected to have an upgrade on the hand tracking feature of Magic Leap 1, which will enable us to remove the LMC from the setup altogether.

Controller. The controller object is invisible, and is responsible for setting the keyboard position and scale. When users want to customize the keyboard appearance, they interact with the controller object where the position of the keyboard is controlled by three sliders associated with the three dimensions. Users arrange the sliders to put the keyboard to the desired position in the AR environment.

Usability Study. To motivate that in-air tapping keyboards are likely to replace current system keyboard designs in commercial AR HMDs, we performed a comparative usability analysis. In this section, we review the procedures and the results of our comparative usability analysis. The goal of this user-based evaluation is to measure the usability of AiRType and compare its performance with the performance of the baseline keyboard. For this evaluation, we use the system keyboard of Magic Leap 1 as the baseline keyboard. As illustrated in Figure 5.5, the system keyboard utilizes the remote controller for the targeting and selection of the keys. Users move the cursor through the ray coming out of the remote to target letters, then pull the trigger button to select the keys. In the system keyboard design, the background becomes blurry when the keyboard opens, which occludes the real-world environment.

Figure 5.5: The system keyboard of Magic Leap 1, i.e., baseline keyboard. The targeting is done through moving the cursor via the ray coming out of the remote controller. The selection is done by pulling the trigger button at the back of the remote.

Figure 5.10 shows the experimental setup for AiRType. The size of the keyboard is kept to the default size and the Leap Motion Controller is placed on the table to capture the hand movements. The computer screen mirrors the content of the AR environment rendered by the HMD for visual evaluation purpose only.

We conduct the user study based on the standard ISO 9241-11 [62] model of usability (1998) which is defined as "the extent to which a software can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use". To comply with ethical guidelines in conducting this work, the user study has been approved by our Institutional Review Board.

Usability Test Setup. We carried out the experiments with two groups of 5 novice participants (with a total of 10 participants). No participant had any experience with any AR HMD before their engagement in this study, and not training was provided to any of the users they participate in the experiments. To conduct this study, we formed the groups randomly and assigned the AiRType to the first group and the baseline keyboard to the second group, respectively. Moreover, the groups

are kept exclusive to avoid any bias through sample leakage. To measure the outcomes of our experiment against the baseline, the users from both groups typed the same target word sequences using different input entry methods. The Institutional Review Board (IRB) at our institution approved this user study.

ISO usability model metrics involves effectiveness, efficiency, and satisfaction. These metrics are defined in the following:

Effectiveness: This metric measures the proportion of erroneous key taps. In our study, we calculate the effectiveness by dividing the number of backspace presses by the target string length, then scale up by 100. The unit of this metric is percentage (%).

Efficiency: This metric measures the task completion time by the user. In this study, we calculate the efficiency by dividing the length of the target string by the time the user spends to complete the task in seconds. The unit for this metric is denoted as character-per-second (cps).

Satisfaction: This metric depicts the System Usability Scale (SUS) score of the design. We measure the satisfaction aspects of the input entry methods by a standardized SUS questionnaire score [41]. The score range of 61-70 corresponds to an average score, whereas a score of $>$80 is considered satisfactory [118].

Procedure. The following are the ordered steps of the procedure we followed in order to conduct the usability test:

1. The investigator calibrates the keyboard for the experiment session; the calibration is redone for each experiment session.

2. The investigator explains the research study, its steps, and ultimate outcomes to the participant.

Table 5.1: The average ISO usability test results obtained from 5 participants per group, 10 participants in total.

	Effectiveness	Efficiency	Satisfaction
AiRType	5.833%	0.624 cps	81.5
Baseline	8.024%	0.604 cps	74.5

3. The investigator informs the participant about two objectives to be achieved: (1) typing a sequence of words using the assigned text entry method, and (2) filling the SUS questionnaire regarding the usability of the assigned text entry method.

4. The investigator demonstrates to the participant how to wear the AR HMD and how to use the assigned text entry method to achieve the first objective.

5. The participant wears the AR HMD and starts typing the target word sequences on the assigned text entry method.

6. Upon completing the first objective, the participant fills the SUS questionnaire to achieve the second objective of the study.

AiRType Usability Results and Discussion. Table 5.1 shows the usability measurement of both keyboard designs using the effectiveness, efficiency, and satisfaction. For the effectiveness, we observe that AiRType is more effective than the baseline, since it decreased the error rate by 27%. Moreover, we observed that users tend to hit the trigger button by mistake while moving the remote to target the actual key with the baseline keyboard. This led to more erroneous key selections, thus increase in the error rate.

We observe that, however, the difference in the efficiency of the keyboards is negligible and conclude that users type with the same speed using both keyboards. However, we also observe that since the participants were inexperienced with the AR environment (see the setup), they were moving their hands or the remote controller slowly to correctly target and select the keys. Therefore,

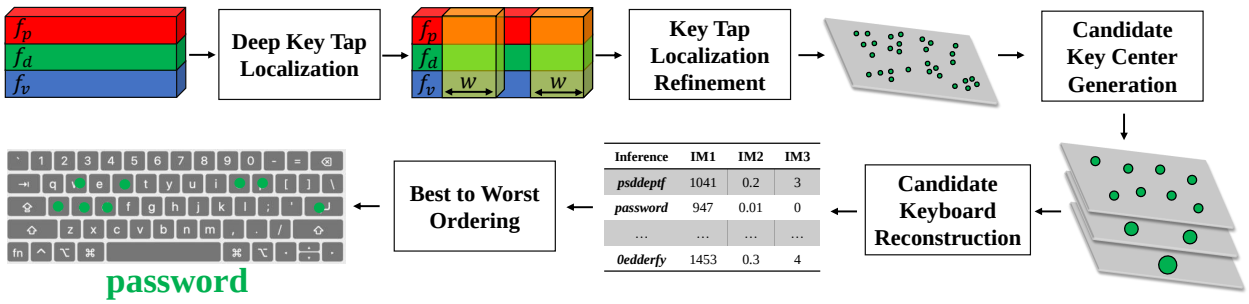


Figure 5.6: The five-step pipeline for the attack, which takes hand tracking data as input and outputs inferences ordered from best to worst. The first two steps are for keystroke detection stage while the subsequent three are for key identification stage. The attack starts with the pre-processing of the data coming from the hand tracker device, and localization of the key tap windows into time slices in the input signal at *Deep Key Tap Localization*. Then, the key taps are refined through a set of processes done with the help of an estimation of the keyboard plane at *Key Tap Localization Refinement*, outputting 2D key tap points. At the next step, *Candidate Key Center Generation* deduces the set of candidate key centroids through clustering of the key tap points. *Candidate Keyboard Reconstruction* composes a set of candidate keyboard reconstructions through computing reconstruction transformation using the candidate key centers. Finally, *Best to Worst Ordering* evaluates, filters and orders the candidate keyboard reconstructions to build a best to worst ordering of the reconstructions through a set of inference measurements.

we anticipate the cps measurements to improve significantly when the participants gain experience with the AR environment and AR HMDs.

Finally, we observe 9.4% increase in the SUS score with AiRType keyboard. The score of the baseline is slightly above the average satisfaction (> 70), while the score of the AiRType is satisfactory (> 80). Considering the metrics in combination, we conclude that AiRType offers a more usable option than the baseline for text entry in augmented reality.

Approach Overview

Having explained the details of AiRType design, in this section, we present the overview of the attack pipeline. To maintain the generalizability of the attack on different AR keyboard designs, we use the term ”AR keyboard” instead of ”AiRType”.

Our attack pipeline consists of five steps as illustrated in Figure 5.6 with the low-level hand tracking

data as input and a list of text inferences ordered from best to worst as output.

In our pipeline, we localize the key tap windows in the time domain signal obtained from the hand tracker in the *deep key tap localization* step. Then, we refine the key tap windows to get the corresponding positions in 2D in the *key tap localization refinement* step. We then obtain a set of positions to use as the key center candidates in the *candidate key center generation* step, derive keyboard reconstructions from the candidate key centers in the *candidate keyboards reconstruction* step. Finally, we obtain our final list of inferences ordered from best to worst through a set of inference measurements in the *best to worst ordering* of keyboard reconstructions step.

In the literature, keylogging inference attacks are defined as a two-stage process: keystroke detection and key identification [96]. The first two steps of our pipeline are for keystroke detection; the subsequent three steps are for key identification.

Deep Key Tap Localization. To initiate our attack, we start by detecting the key taps utilizing the following.

Observation 1. *The hands follow a certain pattern while writing on the AR keyboard, which makes key tap gestures distinguishable from other hand gestures.*

Based on Observation 1, we utilize a Convolutional Neural Network (CNN) to localize key taps in specific time windows where key tap-specific patterns of hand features occur. We follow a similar approach to those used for temporal localization problems in this context. In our modeling, the information available to the adversary is the time-domain low-level hand tracking signal of a user obtained from a hand tracker device, which changes as the sensor is placed differently. We pre-process the low-level data to resolve this dependency and ensure the attack repeatability with the same setup in different spatial configurations of the sensor.

After pre-processing, we utilize a multi-head CNN as a binary classifier to establish the presence of keystrokes. We do an exhaustive search on the overall recording by inputting successive time windows of the pre-processed data into the CNN. Through non-max suppression and energy thresh-

olding, we obtain a set of windows classified to include the key tap actions. After localization, the touchpoints to the keys, called as **key tap points**, can be estimated where the tip positions of the finger at the middle of the key tap windows.

Key Tap Localization Refinement. Since we obtain the deep key tap localizations using estimation methods, the key tap points are imprecise. We further refine the key tap localization based on a geometric observation on the keyboard to improve the results. Due to the planar structure of the keyboard, we observe the following:

Observation 2. *The key tap points lay on a plane in 3D.*

Based on Observation 2, we estimate a keyboard plane that minimizes the sum of the squared distances to the key tap points and check each key tap window against the plane by whether the fingertip crosses the plane or not. For a key tap point, we use it to improve the precision of the estimated key tap if there is an intersection. Otherwise, we eliminate the key tap point to suppress false positive samples. After refining the key tap points in 3D, we obtain 2D key tap points through dimension reduction. This procedure is particularly essential in the candidate keyboard reconstruction step as it decreases the number of correspondent points required to compute the similarity transform. To achieve this, we transform the points into a space constructed using the normal vector of the plane. Then, we omit the component with no change among all the key tap points.

Candidate Key Center Generation. We deduce candidates for the key centers from the key tap points since it is an essential input parameter for exploring the possible keyboard reconstructions. We set forth the following observation concerning the relationship between the key centers in the AR keyboard and the key tap points:

Observation 3. *When each key center in the AR keyboard is treated as the cluster centroid of that key, each key tap point belongs to the cluster of the tapped key.*

Observation 3 is only valid for alpha-numeric keys due to the special keys' physical layouts. Based on Observation 3, we hypothesize that proper clustering of the key tap points eventuates partitioning them into clusters of unique keys. For proper clustering, the number of clusters should be the same as the number of unique keys used in the session. Since not all unique keys are necessarily used during a session, the attacker is unaware of the number of clusters required, which is a key challenge in this step. We apply a weighted agglomerative hierarchical clustering algorithm on the key tap points and obtain a set of clusterings with a varying number of clusters to address this problem.

Candidate Keyboard Reconstruction. In this step, we establish the possible keyboard reconstructions with the help of our hypothesis built upon Observation 3 using the candidate key centers from the previous step. To achieve this task, we account for the possibility where each candidate key center corresponds to any key on the keyboard. We obtain countless inferences with different center – key pair assignments as an output of this exhaustive process.

The coordinate systems of the hand tracker and the AR keyboard model change at each session due to the hand tracker's positioning and the positioning and scaling of the AR keyboard in the virtual environment. Thus, there exists a similarity transform T (i.e., translation, rotation, uniform scaling) between two coordinate systems, changing for each session. Once T is found, the key tap points are projected onto the space of the keyboard model, i.e., the keyboard is reconstructed. As a result, the key tap points can be mapped to their corresponding keys on the AR keyboard.

The computation of the 2D similarity transform T—needed to reconstruct the keyboard—requires two corresponding points from each coordinate systems. For every possible tuple of center – key assignments, we solve for T and acquire the inferences, forming the set of candidate keyboard reconstructions.

Best to Worst Ordering of Keyboard Reconstructions. The previous step produces a set of keyboard reconstructions in the order of hundreds of thousands, many of which are entirely inaccurate due to incorrect center – key pairings or selection of centers. Although the search space

is significantly reduced compared to having random guesses, it is still infeasible for the attacker to consider all reconstructions. We observe that accurate reconstructions typically have specific characteristics, which draw them apart from others. In this step, we benefit from such characteristics to order the keyboard reconstructions from best to worst, drastically decreasing the number of reconstructions to consider.

Technical Details and Methods

In this section, we elaborate our approach and technical contributions in each step of the pipeline.

Deep Key Tap Localization. We pre-process the low-level hand tracking data and localize the presence of key taps into time windows using a multi-head CNN.

Pre-processing. With pre-processing, we obtain three features from the low-level hand tracking data as follows: 1) the tip position of each finger w.r.t. the palm center of the hand (f_p), 2) the pointing direction of each finger w.r.t. to the normal vector of the palm (f_d), and 3) the velocity of each fingertip computed using f_p (f_v). We call them position, direction, and velocity sub-signals, respectively.

We denote a time-domain hand features signal as $X = \{x_t\}_{t=1}^{T}$ where $x_t = [f_p\ f_d\ f_v]^T$ is the t-th frame in X. Each signal X is associated with a set of temporal key tap action annotations $\Psi = \{\psi_m\}_{m=1}^{M}$ where $\psi_m$is the time instance when the keystroke of m-th key tap occurred. A segment s is a slice from the signal, denoted by its starting and ending time, $s(t_{start}, t_{end})$ which includes the consecutive frames between x_{start} and x_{end}.

Network Architecture. As shown in Figure 5.7, the key tap localization network is a multi-head CNN that operates on the pre-processed signal. The convolutional layers after each head capture the spatial and temporal features of each finger. Two fully connected layers follow to extract high-level spatio-temporal features of the hand from finger features through position, direction, and velocity. Then, the output from each head is concatenated, which is forwarded to a final fully-

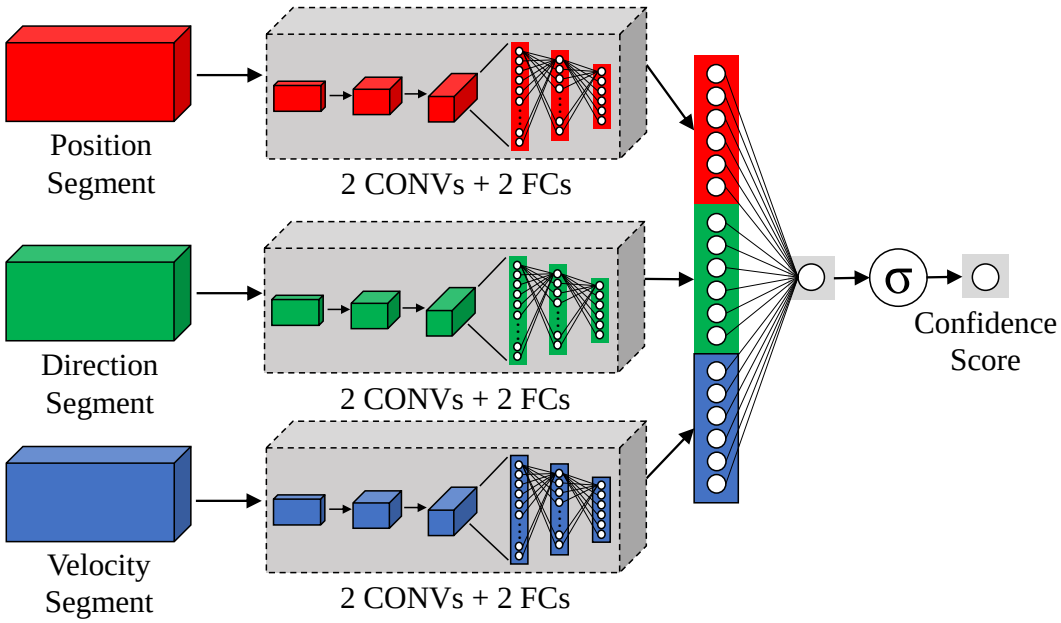


Figure 5.7: The multi-head CNN architecture used for key tap localization. The input is a segment from the pre-processed low-level hand tracking data, while the output is the confidence score for the segment belonging to a key tap.

connected layer followed by a sigmoid activation function, which outputs the confidence score. The confidence score is used to perform binary classification on the input segment to differentiate key tap segments from the background.

We interpret each key tap as a 30-frame action, 375 ms, given that the data from the hand tracker device is collected with a sampling rate of 80 fps. The number of frames is decided by the 5-fold cross-validation in our evaluation. Each head of the CNN takes the corresponding sub-segment with the shape of height (5), width (30), and depth (3), depicting five fingers, 30 frames, and three spatial dimensions.

Prediction. For prediction, we slide a fixed-length (30) temporal window on the input to generate candidate segments and input them to the CNN to obtain the corresponding confidence scores P_c. Then, we remove redundant detections by applying non-max suppression to ensure that the predictions are disjoint and only the most confident predictions are kept among temporally intersecting ones. To obtain the final predictions, we eliminate the segments with a confidence score P_c less

than the classification threshold $\tau_c = 0.5$ with the rationale explained in the evaluations. Finally, each segment is associated with a key tap point by fetching the position of the index fingertip at the midpoint of the window.

Key Tap Localization Refinement. After key tap localization, we estimate a keyboard plane then refine the key tap points described in the following.

Keyboard plane estimation. A plane in 3D is defined by a normal vector $\mathbf{v} = [v_x\, v_y\, v_z]$ perpendicular to the plane, a point $\mathbf{p} = (x, y, z)$ on the plane, and a scalar d. The plane equation can also be represented in scalar format as $v_x x + v_y y + v_z z + d = 0$. To find a fitting plane to the estimated 3D key tap points, we use a regression model that minimizes the linear least square error between the points and the plane. For that, a linear algebra trick is used to solve the equation of four unknowns, v_x, v_y, v_z, and d, where the solution space is three dimensional, i.e., a plane. The trick is by assigning -1 to v_z and rearranging the equation such that $v_x x_i + v_y y_i + d = z_i$. The error function then becomes $((v_x x_i + v_y y_i + d) - z_i)^2$ (ideally 0, where the point (x_i, y_i, z_i) lays on the plane). Thus, the problem is reduced to an optimization where the error is calculated as $E(v_x, v_y, d) = \sum_{i=0}^{m} [(v_x x_i + v_y y_i + d) - z_i]^2$, where the coefficients of the best-fitted plane, namely v_x, v_y, and d, are found by solving the optimization problem defined as $\underset{v_x, v_y, d}{\arg\min} \sum_{i=0}^{m} [(v_x x_i + v_y y_i + d) - z_i]^2$.

Reducing the False Positives. We follow the trace of the fingertip for each key tap window and check if it crosses the estimated plane. Otherwise, we eliminate it since the finger does not touch the keyboard plane, thus any key on the keyboard.

Refining the Key Tap Points. To increase the spatial precision of the key tap points, we use the intersection point where the fingertip crosses the estimated keyboard plane instead of using the midpoint of the key tap window as before as illustrated in Figure 5.8.

Dimension Reduction. Since all the key tap points lay on a plane in 3D, they can also be represented as 2D points without loss of information. To achieve this task, we first create an orthonormal

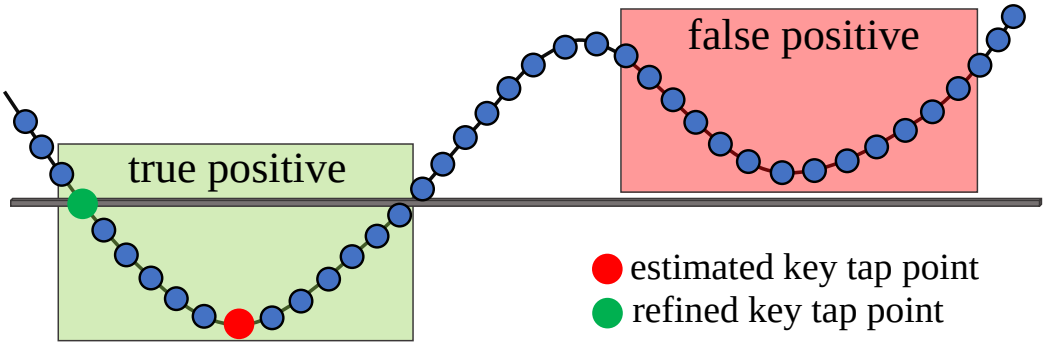


Figure 5.8: Refining through the estimated keyboard plane and the trace of the index fingertip. The dots and the curve connecting the dots represents the trace of the index fingertip. The green window is considered as true positive since the window encapsulates a trace crosses the estimated plane. The associated key tap point is refined as the green circle instead of the red circle. The red window is considered as false positive and eliminated since the encapsulated trace does not cross the plane.

basis containing the normal vector of the keyboard plane using the Gram-Schmidt Orthogonalization. Then, we change the basis for the key tap points to the newly found basis. Finally, we obtain the corresponding 2D key tap points K by omitting the component that has no change among all the key tap points, which corresponds to the component in the direction of the normal vector.

Candidate Key Center Generation. This section explains how the candidate key centers are deduced from the set of refined key tap points. Based on Observation 3, the key tap points play an essential role in defining the key centers since each key tap point stands as a sample contained in the cluster of the corresponding key in the AR keyboard. A reverse procedure to recover the key centers from the key tap points is **clustering**, yet the uncertainty in the number of clusters introduces a challenge. We generate cluster groups with a varying number of clusters through agglomerative hierarchical clustering to account for the usage of a varying number of unique keys in different sessions.

We denote a cluster group as $C = \{\mathbf{c}_m\}_{m=1}^{M}$ where $\mathbf{c}_m$ is the 2D vector representing the centroid of the m-th cluster in the group C. The output is a set of cluster groups, G, each with a different cluster

Algorithm 1: Construction of center groups

1: G: Cluster groups with varying number of clusters
2: C: Current set of weighted centroids
3: K: Key tap points
4: **procedure** BUILDCENTERGROUPS(K)
5: **initialize:**
6: $G \leftarrow \emptyset$
7: $C \leftarrow \{(\mathbf{v}, w) \in \mathbb{R}^2 \times \mathbb{R} \mid \mathbf{v} \in K, w = 1\}$
8: **while** size(C) > 2
9: $\mathbf{c}_1, \mathbf{c}_2 \leftarrow$ MINIMUMDISTANCEPAIR(C)
10: $\mathbf{c}_{merged} \leftarrow$ MERGECENTERS($\mathbf{c}_1, \mathbf{c}_2$)
11: $C \leftarrow (C \cup \{\mathbf{c}_{merged}\}) \setminus \{\mathbf{c}_1, \mathbf{c}_2\}$
12: $G \leftarrow G \cup \{C\}$
13: **return** G
14: **procedure** MINIMUMDISTANCEPAIR(C)
15: **return** $(\mathbf{c}_1, \mathbf{c}_2 \mid \mathbf{c}_1, \mathbf{c}_2 \in C,$ EUCDIST$(\mathbf{c}_1, \mathbf{c}_2)$ is min$)$
16: **procedure** MERGECENTERS($\mathbf{c}_1, \mathbf{c}_2$)
17: $(\mathbf{v}_1, w_1), (\mathbf{v}_2, w_2) \leftarrow \mathbf{c}_1, \mathbf{c}_2$
18: $\mathbf{v}_{merged} \leftarrow$ WEIGHTEDVECTORAVG($\mathbf{c}_1, \mathbf{c}_2$)
19: $w_{merged} \leftarrow w_1 + w_2$
20: **return** $(\mathbf{v}_{merged}, w_{merged})$

count. The set of cluster groups is denoted as $G = \{C_n\}_{n=1}^{N}$, where C_n is the n-th clustering.

Algorithm 1 shows the procedure to compose the cluster groups G. Initially, all key tap points are treated as singleton clusters with uniform weights as in line 1.7. Then, the cluster group C is iteratively updated (line 1.11) by merging the closest clusters considering the weights. Each update to the cluster group C results in adding a cluster group to G (line 1.12) with a different cluster count. The Euclidean distance is used to find the closest cluster pair. Merging is done through weighted vector average to ensure candidate clusters are created considering an equal effect of each key tap point in the cluster.

The number of unique keys used in a session is upper-bounded by the number of keys in the AR keyboard. Therefore, we eliminate the cluster groups containing more clusters than the number of keys, which is 42 for the AR keyboard. This elimination implies that the number of reconstructions to explore in the following steps is also upper-bounded. Thus, the search space is not affected if

Algorithm 2: The algorithm to compute similarity transformation T using two pairs of corresponding points in 2D. It can be used to map any point $\mathbf{a}$ in space A to its correspondent $\mathbf{b}$ in space B.

procedure COMPUTETRANSFORMATION($\mathbf{a}_1, \mathbf{a}_2, \mathbf{b}_1, \mathbf{b}_2$)

$\mathbf{t} \leftarrow \mathbf{b}_1 - \mathbf{a}_1$ ▷ translation vector

$s \leftarrow \frac{|\overrightarrow{\mathbf{a}_1\mathbf{a}_2}|}{|\overrightarrow{\mathbf{b}_1\mathbf{b}_2}|}$ ▷ uniform scaling scalar

$\alpha \leftarrow \cos^{-1} \frac{\overrightarrow{\mathbf{a}_1\mathbf{a}_2} \cdot \overrightarrow{\mathbf{b}_1\mathbf{b}_2}}{|\overrightarrow{\mathbf{a}_1\mathbf{a}_2}| \cdot |\overrightarrow{\mathbf{b}_1\mathbf{b}_2}|}$ ▷ rotation angle

return FORMMATRIXT($\mathbf{t}, s, \alpha$) ▷ see [108]

the number of key tap points recorded in a session grows above the keys.

Candidate Keyboard Reconstruction. We establish the candidate keyboard reconstructions using a set of candidate key centers. In this section, we first discuss the challenges of keyboard reconstruction then outline our approach to address those challenges. We show how we take advantage of the candidate key centroids to obtain a broad set of keyboard reconstructions.

Due to the differences in origins and orientations of the coordinate systems of the hand tracker and the AR keyboard model, the same geometric constructs (e.g., points, vectors) in a shared environment (i.e., real-world) are expressed differently by those two parties. Moreover, the user's flexibility on positioning and scaling the AR keyboard differently in the virtual environment introduces yet another discrepancy with the addition of a difference in scale. Considering these series of linear transformations between the coordinate systems, we can speak of a similarity transform $T : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ which maps any given geometric construct in hand tracker's coordinate system to that of the AR keyboard model. Using two corresponding point pairs from each space, T could be computed as shown in Algorithm 2. We refer to T( ) as the similarity transformation function that applies T to its input. We denote the uniform scaling component of T as T_s.

Having T, the key tap points K could be transformed into the space of the AR keyboard model as $K' = \text{T}(K)$. Then, each key tap position k' in K' could be associated with a key by checking which key's area k' falls into. This results in recovering the keys tapped. However, the adversary

lacks two correspondent points from both spaces, failing to directly obtain a single, accurate reconstruction. To address this challenge, we use the clues we have regarding the key centers from the previous step. As there is no prior knowledge on which cluster corresponds to which key, we account for the possibility of each cluster c belonging to any unique key j for each cluster group C in G. We call a pairing of cluster center $\mathbf{c}$ with a key $\mathbf{j}$ as **center – key pair**, and denote it as $r = (\mathbf{c}, \mathbf{j})$. Then, we obtain candidate keyboard reconstructions by computing T using every possible tuple of center – key pairs. We summarize the reconstruction of candidates in Algorithm 3.

Best to Worst Ordering of Reconstructions. The results from the previous step include reconstructions with accurate results, only possible when the input cluster centroids accurately approximate the key centroids and the center – key pairing is accurate. However, the reconstructions are predominantly composed of inaccurate results due to contrary cases, as we show in our evaluation. Although the search space is favorable, especially when compared to a random guess, this space is still infeasible to explore by the attacker for most cases. We notice that the inaccurate reconstructions exhibit specific characteristics differing from the accurate ones. This section discusses how we exploit such observations to eliminate invalid reconstructions and order the remaining from best to worst. To achieve this task, we consider three inference metrics: scaling factor ($\mathbf{IM_1}$), outlier ratio ($\mathbf{IM_2}$), and the difference between the number of clusters and unique keys ($\mathbf{IM_3}$) as explained in the following.

$\mathbf{IM_1}$: Scaling Factor. In general, users have the flexibility of scaling the size of the AR objects in the virtual environment, which is why we consider a variable scaling factor T_s. However, making the keyboard too small or too large negatively affects usability. A smaller keyboard makes it difficult to pick the correct key, resulting in unintentional taps on the neighbors of the target keys. In comparison, a larger keyboard increases the distance the hands have to travel to the keys at the sides of the keyboard, affecting usability. Moreover, with a too large keyboard, the keyboard may not fit into the frame of the AR HMD. As such, we define upper and lower limits for the scaling factor T_s to eliminate the reconstructions that result in estimations with keyboard sizes that are

Algorithm 3: Reconstruction of candidate keyboards

G: Cluster groups with varying number of clusters
K: Key tap points
J: 2D centroids of the keys in the AR keyboard model
L: The candidate keyboard reconstructions

procedure COMPUTECANDIDRECONS(G)
  initialize: $L \leftarrow \emptyset$
  for each C in G
    for each permutations of centers $\mathbf{c}_1, \mathbf{c}_2$ from C
      for each combinations of centroids $\mathbf{j}_1, \mathbf{j}_2$ from J
        $T \leftarrow$ COMPUTETRANSFORMATION($\mathbf{c}_1, \mathbf{c}_2, \mathbf{j}_1, \mathbf{j}_2$)
        $K' \leftarrow \mathrm{T}(K)$
        $L \leftarrow$ INFERKEYS(K') $\cup\, L$
  return L

unlikely to be used. Figure 5.9 shows keyboards configured with borderline sizes (0.5 and 2.0) as visual reasoning for the chosen limits, which shows the infeasibility of such dimensions.

$\mathbf{IM_2}$: Outlier Ratio. All key tap points k' in K' should optimally fall into the area of some key in the AR keyboard. However, outlier key tap points are observed in improper reconstructions. To measure the existence of such anomaly, we define the outlier ratio as $\mathbf{IM_2}$ and compute it as the ratio of the number of outliers to all key tap points. Although the optimal value for $\mathbf{IM_2}$ is zero, we expect a small number of outliers because of the false positive samples in the key tap points detected at the outside of the AR keyboard.

$\mathrm{IM_3}$: Difference Between the Number of Clusters and Unique Keys. Our hypothesis built on Observation 3 suggests that clustering key tap points with the proper cluster count choice divides the key tap points into clusters of unique keys. For a reconstruction l in L built using the center – key pairs from some cluster group C in G, we expect the number of clusters in C to be optimally equal to the number of unique keys found in l. For contrary cases, we measure the anomaly using $\mathbf{IM_3}$, which is computed as the difference between the number of clusters and unique keys. The optimal value for $\mathbf{IM_3}$ is zero, yet the exceptional cases may result in values not equal but close to zero for accurate inferences. These include cases where the number of key tap points is not

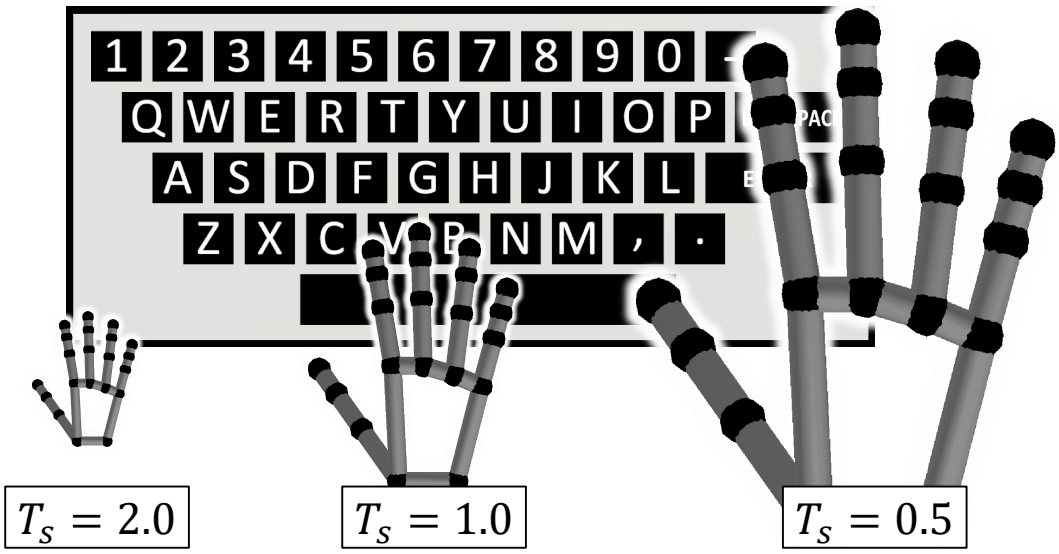


Figure 5.9: The demonstration of the maximum and the minimum scaling factors ($\mathbf{IM_1}$) used to filter the candidate keyboard reconstructions. For concise visualization, the effect of the scaling factor T_s is shown by inversely scaling the hand model rather than scaling the keyboard itself. In the virtual environment, however, the keyboard model gets bigger as the scaling factor T_s grows while the size of the hand model is fixed.

sufficient to approximate the centers of the keys well or when the key tap points for some keys happen to accumulate around some point that is not close enough to the center of the key.

After obtaining $\mathbf{IM_1}$, $\mathbf{IM_2}$ and $\mathbf{IM_3}$ measurements, we first eliminate the invalid reconstructions with respect to $\mathbf{IM_1}$ by enforcing a scaling factor limit. We continue processing the reconstructions using $\mathbf{IM_2}$ and $\mathbf{IM_3}$. However, we do not directly enforce the optimal values for these parameters due to the exceptional cases and accurate reconstructions, as we discussed earlier. To capture and estimate the joint role of $\mathbf{IM_2}$ and $\mathbf{IM_3}$ on the reconstruction's accuracy, we utilize a linear regression model, with the values of $\mathbf{IM_2}$ and $\mathbf{IM_3}$ as an input, where the output is the expected accuracy.

The accuracy is measured in terms of the normalized Levenshtein similarity between the estimated and the ground truth strings, which helps measure the performance of both the keystroke detection and key identification stages simultaneously [96]. To compute it, first, the Levenshtein distance [75] between two strings is calculated as the minimum number of insertions, deletions,

and substitutions required to transform one string into another. Then, the distance is normalized to be in the range $[0, 1]$ by dividing the distance by the length of the longest input string. Finally, the normalized distance measurement is inverted to normalized Levenshtein similarity by subtracting the resulting distance from 1.

Upon filtering using $\mathbf{IM_1}$, we feed the measurements of $\mathbf{IM_2}$ and $\mathbf{IM_3}$ for each remaining candidate reconstructions into the regression model. Then, we sort the reconstructions in a decreasing order w.r.t. the predicted accuracy. These sorted reconstructions constitute the final output of our pipeline, best to worst ordering of the keyboard reconstructions. Note that once the correct keyboard reconstruction is found, it is straightforward to trace any input on the keyboard, including the special characters and keys.

Evaluation

To evaluate the performance of our approach, we conducted various experiments on the individual steps of the pipeline and end-to-end. The experiments involved five participants, including one female and four males with ages ranging from 23 to 37. The Institutional Review Board (IRB) at our institution approved the data collection process.

We first describe our experimental setup. Then, assessing the performance of our approach in keystroke detection, we continue with the evaluation of the key identification stage. Next, we evaluate the performance of the end-to-end pipeline. Finally, we discuss the possible countermeasures. With the evaluations of the individual steps, we intend to: i) reason about the validity of our assumptions which led to construct each step, ii) explore how each step contributes to the overall pipeline through certain types of information gain, and iii) explore the limitations of each step and discuss their effects on the overall approach.

Experimental Setup. Since all of our threat models utilize the same type of low-level hand tracking data, we simulated and evaluated all threat models with the same experimental setup irrespective of the data collection medium. For our experiments, Magic Leap 1 (ML1) is used as the

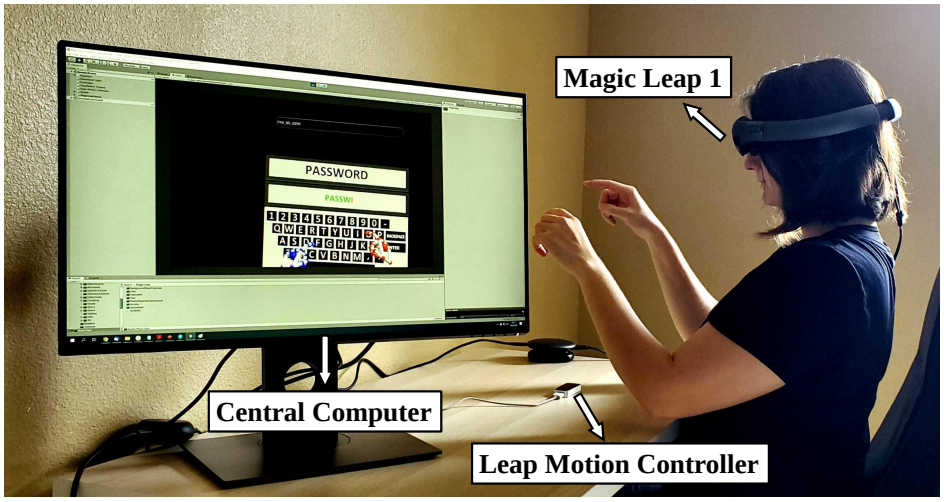


Figure 5.10: The setup used to carry out the experiments.

AR HMD, and Leap Motion Controller (LMC) is used for hand tracking shown in Figure 5.10. LMC can track hands within 60 cm in a 120×150° field of view, and ML1 detects hands within 80 cm distance. Both devices are connected to a single computer used to drive the experiments. ML1 is run in Zero Iteration (ZI) mode [24] to ensure the data flow between two devices and control the overall experiment on the computer. A single Unity3d application is run on the computer to handle the required computations for the experiments. This application also composes and provides the frames to ML1.

The position information of the index fingertips is essential to utilize the AR keyboard as it is used to detect the keystrokes at the user side. The current state of ML1's hand tracking is insufficient to use the AR keyboard since the fingertip positions do not remain stable as the hands move. Although the literature [112] shows that the required hand tracking can be achieved using the available sensors, we instead use LMC to simplify the model. Given that these devices and their software are rapidly evolving, we expect to see steady hand tracking functionality as default in ML1.

A single LMC is used to provide data to both the user and the attacker side in our setup. Since ML1 and LMC do not share a common coordinate system, we performed the following procedure

at the beginning of each session. First, we obtained a set of corresponding points (the fingertip positions) by holding the hands still in a position where both devices can stably detect the fingertip positions. Then, we estimated the transformation through a Singular Value Decomposition (SVD)-based estimation algorithm [21, 22]. Finally, we applied the computed transformation to all data coming from LMC to be used at ML1.

We measure the accuracy of a text inference in terms of the normalized Levenshtein similarity. We use the pinpoint accuracy, h-hop, and top-k accuracy [111]. For h-hop accuracy, h-hop neighbors, which are the keys that can be reached by taking h hops from the source key, are considered. It is computed as the normalized Levenshtein similarity where two keys k_1 and k_2 are said to match if k_1 is in the h-hop neighborhood of k_2, or vice versa. The pinpoint accuracy is a special case of h-hop accuracy, where h is zero. The accuracy in top-k results corresponds to the maximum accuracy achieved within the first k results populated in the output inferences list. The h-hop metric helps show how some portion of the inferences we obtain still causes serious privacy leaks even though they do not yield high pinpoint accuracy. Top-k is essential to evaluate our method since we output a list of inferences instead of one.

Keystroke Detection. In this section, we evaluate the first two steps of the pipeline: deep key tap localization and key tap localization refinement. First, we collected data from two distinct users while writing 54 random pangrams (108 pangrams in total; hereafter, the dataset is called **P108**), mainly used throughout the paper to simulate the data recorded by the adversary for training. The data is then split into train-validation (96 pangrams) and test (12 pangrams) splits.

Deep Key Tap Localization. We train the CNN as a binary classifier to differentiate a key tap action from the background. We denote the training data as $S_{train} = \{(s_n, l_n)\}_{n=1}^{N}$, where label $l_n \in \{0, 1\}$. The positive (1) and negative (0) labels correspond to key tap action and background. We construct S_{train} by splitting the data into a set of disjoint segments of key tap or background data. For that, we sample the key tap segments from the input data and assign a positive label to each of them. Similarly, we sample disjoint background segments from the remaining background

data and assign a negative label to each of them.

In our experimental evaluation, we use Adam optimizer [66] with a learning rate of 0.001, L2 regularization with a penalty of 0.001, and the binary cross-entropy loss function, which is determined by 5-fold cross-validation on the training-validation split. Since the data is imbalanced with a larger number of background segments, we apply a manual re-scaling to the loss of each element with a weight inversely proportional to its class frequency.

After training, we perform predictions on the test split. The prediction procedure takes the set of input segments $S_{\text{test}} = \{s_n\}_{n=1}^{N}$ as input and outputs the key tap segments with their confidence scores: $P = \{(s_m, \alpha_m)\}_{m=1}^{M}$.

Given each ground truth key tap segment gt_l in $GT = \{gt_l\}_{l=1}^{L}$, and upon prediction, we associate each key tap segment s_m in P with the ground truth segment gt_l which is closest to s_m in time.

Also, we keep track of the intersection-over-union (IoU) of each segment association as a measure of the overlap ratio between the predicted and the ground truth key tap segments in time. For two segments, $A(t_1, t_2)$ and $B(t_3, t_4)$, where t_1 and t_3 are the start time, t_2 and t_4 are finish time, and $t_3 > t_1$, IoU between A and B is defined as $min(0, t_2 - t_3)/(t_4 - t_1)$, which takes values in $[0, 1]$. IoU values are maximal when the segments entirely overlap. As a result, we obtain a set of associations of segments and their IoU values: $A = \{(s_m, gt_l, \text{IoU})\}$. We then eliminate the associations with IoU $< \tau_{\text{IoU}}$ from A, where τ_{IoU} is the IoU threshold. This elimination helps to evaluate the model over different IoU thresholds by changing τ_{IoU} to reason about the precision of the model in temporal localization. Following the standard evaluation methodology in temporal localization problems, we interpret the segments in A as true positives, $P \setminus A$ as false positives, $GT \setminus A$ as false negatives, and the remaining segments in S_{test} as true negatives.

We perform prediction on the test split with different τ_{IoU} values for performance evaluation. The Precision-Recall (PR) curves for left and right hand and varying τ_{IoU} values are shown in Figure 5.11.

To obtain the final predictions used in the subsequent steps, we establish a classification threshold

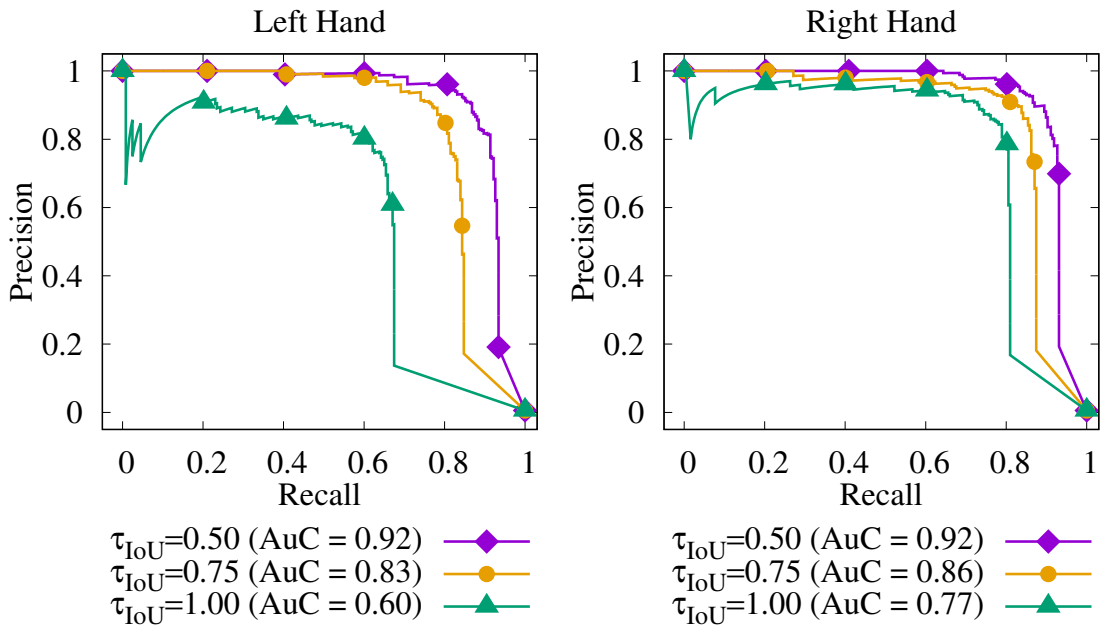


Figure 5.11: The precision-recall curve of the multi-head CNN in *Deep Key Tap Localization*. Different networks are trained for each hand, and both of them perform similarly well.

of $\tau_c = 0.5$. For the true positive vs. true negative trade-off controlled by τ_c , the chosen τ_c value is inclined to favor a greater number of true positive samples while keeping the false positives less than the true positives. The former fact comes in handy in avoiding false negatives. The latter ensures that the following step of the pipeline, *Key Tap Localization Refinement*, can assume that the predictions are dominantly composed of true positives. Eventually, in the next step, we eliminate a significant number of false positives and obtain an overall keystroke detection result that is not possible just by tweaking τ_c .

Key Tap Localization Refinement. To assess the performance of this step, we performed *Key Tap Localization Refinement* on the output obtained from the previous step. As shown in Figure 5.12, the refining procedure significantly decreased the number of false positives (by 64.1% and 86.2%, respectively) while slightly affecting the true positives (decreased by 3.4% and 13.1%, respectively). Thus, the results showed that utilizing the implications of Observation 2 improved the performance of keystroke detection.

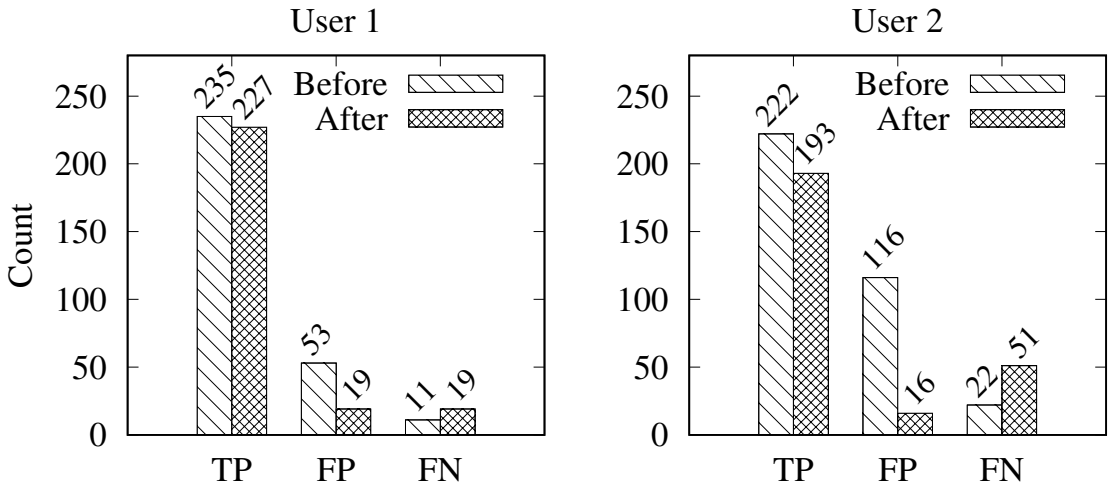


Figure 5.12: The effect of the *Key Tap Localization Refinement* on the number of true positive (TP), false positive (FP), and false negative (FN) samples for different users.

The slight decrease in true positive segments can be negligible while performing the inference attack on a long text with a context, since the missing letters are deductible while there is a context. However, the refining of the points may not be as beneficial for random sequences such as passwords, since true positives are vital to define search space for password cracking.

Key Identification. This section evaluates the last three steps of the pipeline constituting the key identification stage. To this end, we collected data from three users (other than P108's users) while they were writing a random e-mail selected from the Enron e-mail dataset [68]. For length uniformity, we use the first 250 characters for each recording. We denote this data as **E-mail** data. Additionally, we collected three sets, **R5**, **R10**, and **R15**, while the users were writing random character sequences with varying lengths of 5, 10, and 15. We generate the sequences by considering the letter frequencies in the English language, and we collected 15 different sequences for each length. The reported results are the average of the 15 sequences. We aim to simulate the long-text scenarios, e.g., writing an e-mail, with the **E-mail** dataset while using **R5**, **R10**, and **R15** for simulating the entry of short-text scenarios, e.g., writing login credentials. In this section, we use User-1's data and directly use the ground truth key tap points. We report the evaluation of the

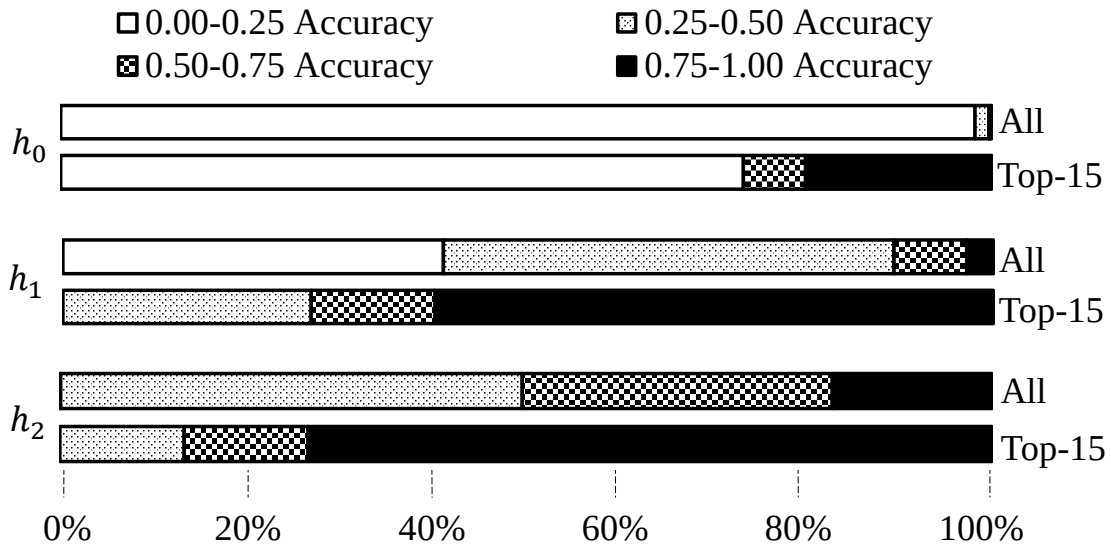


Figure 5.13: The accuracy distribution of the E-mail keyboard reconstructions before and after best to worst ordering. "All" corresponds to reconstructions after $\mathbf{IM_1}$ filtering. Top-15 ordering and all reconstructions after $\mathbf{IM_1}$ filtering categorized by their h_0, h_1, and h_2 accuracy quarters.

end-to-end pipeline with the data from all users under End-to-End Pipeline below.

We first perform *Candidate Key Center Generation*, then *Candidate Keyboard Reconstructions*. This results in 889200 keyboard reconstructions for the e-mail data, 9750, 52000, and 126750 keyboard reconstructions for each sequence in **R5**, **R10**, and **R15**, respectively. The difference in the number of keyboard reconstructions is due to the number of key taps, and it is upper-bounded by 889200 reconstructions due to the upper bound on the number of centers. Among those, we are able to achieve a maximum pinpoint (h_0) accuracy of $\mathbf{97}\%$ for **E-mail**, $\mathbf{99}\%$, $\mathbf{100}\%$, and $\mathbf{99}\%$ for **R5**, **R10**, and **R15** sets, respectively, and all reach $\mathbf{100}\%$ of h_1 and h_2 accuracy. Although not all reconstructions have the same accuracy, the results show that the key identification successfully includes accurate reconstructions. The evaluation results of **R5**, **R10**, and **R15** also demonstrate that our method works with a limited number of samples, as low as 5, to create the correct keyboard reconstruction.

Upon obtaining the reconstructions, we continue with the best-to-worst ordering by first filtering the reconstructions based on the minimum and the maximum scaling factors ($\mathbf{IM_1}$) and consider

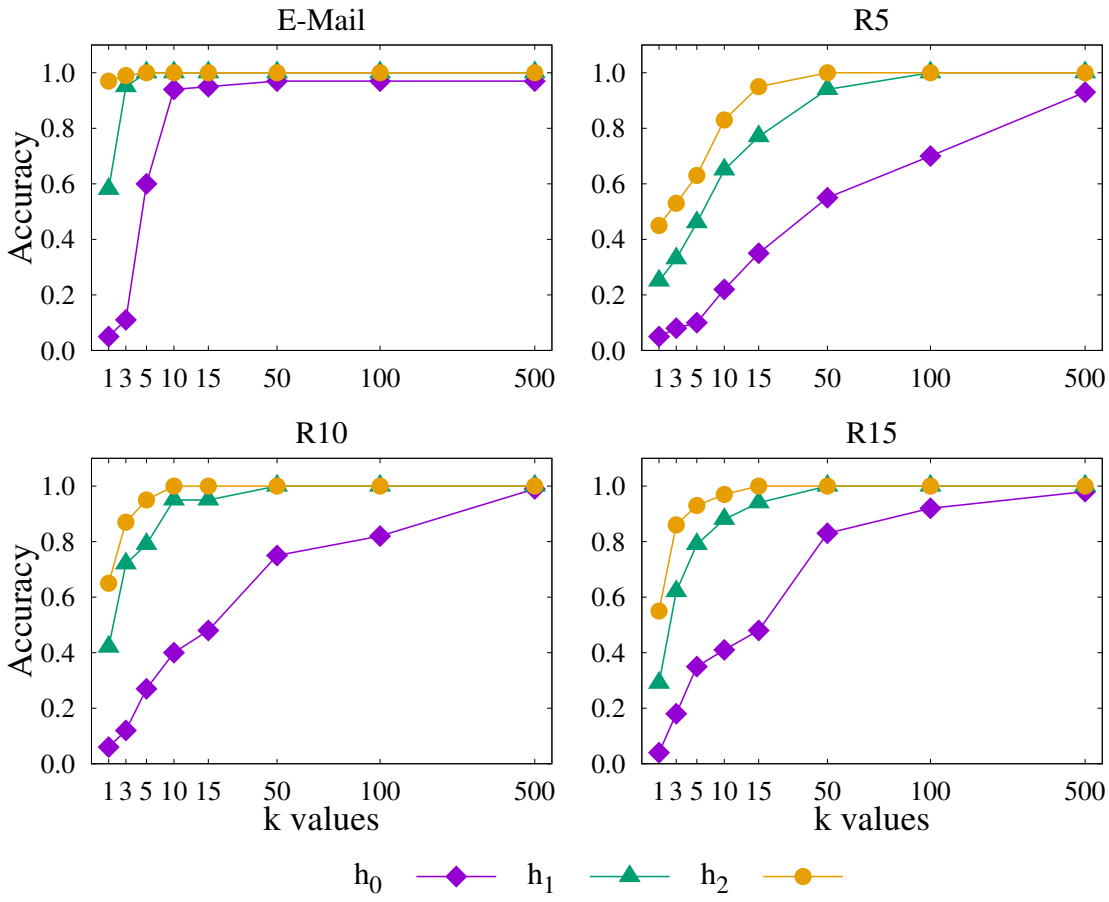


Figure 5.14: The top-k h_0, h_1, and h_2 accuracy for varying k considerations for each data set obtained by utilizing the key identification steps on the ground truth key tap points.

the pinpoint (h_0) accuracy in evaluating its effect. The filter extracts over 40% of the reconstructions with accuracy ranging between 0% - 25% and keeps all reconstructions with accuracy higher than 50%. We then continue with the ordering through $\mathbf{IM_2}$ and $\mathbf{IM_3}$. We fit a linear regression model using the **P108** data, and obtain the best-to-worst ordering for the reconstructions belonging to **E-mail**, **R5**, **R10**, and **R15**. Figure 5.13 shows the distribution of all (after $\mathbf{IM_1}$ filtering) and the top-15 reconstructions for **E-mail** data on different accuracy quarters.

The results highlight that ordering the reconstructions indeed uplifts the better reconstructions to the top of the list and significantly increases the probability of finding an accurate reconstruction even when randomly chosen from the first 15 elements of the list. Figure 5.14 demonstrates the top-

Table 5.2: The maximum h_0, h_1, and h_2 accuracy achieved for each data set across varying top-k orderings and brute-force attack.

		Top-1			Top-3			Top-15			Top-50			Top-100			Top-500			Random		
	User	h_0	h_1	h_2	h_0	h_1	h_2	h_0	h_1	h_2	h_0	h_1	h_2	h_0	h_1	h_2	h_0	h_1	h_2	h_0	h_1	h_2
E-mail	**1**	0.15	0.38	0.53	0.19	0.42	0.54	0.61	0.65	0.66	0.61	0.65	0.66	0.62	0.65	0.66	0.63	0.65	0.66	0.06	0.16	0.30
	2	0.15	0.40	0.66	0.17	0.40	0.66	0.65	0.73	0.77	0.66	0.74	0.78	0.67	0.74	0.78	0.68	0.74	0.78	0.06	0.18	0.34
	3	0.12	0.40	0.51	0.17	0.40	0.56	0.21	0.49	0.58	0.40	0.55	0.59	0.40	0.55	0.59	0.40	0.55	0.59	0.10	0.17	0.33
R5	**1**	0.04	0.31	0.56	0.09	0.43	0.66	0.29	0.71	0.82	0.49	0.80	0.84	0.65	0.84	0.84	0.76	0.84	0.84	0.06	0.20	0.46
	2	0.08	0.19	0.35	0.17	0.37	0.53	0.23	0.46	0.54	0.31	0.50	0.54	0.38	0.53	0.54	0.50	0.53	0.54	0.08	0.21	0.46
	3	0.03	0.26	0.44	0.13	0.38	0.50	0.25	0.50	0.63	0.30	0.52	0.64	0.41	0.55	0.64	0.46	0.60	0.65	0.11	0.24	0.46
R10	**1**	0.10	0.39	0.60	0.23	0.67	0.81	0.46	0.85	0.89	0.59	0.89	0.89	0.71	0.89	0.89	0.87	0.89	0.89	0.09	0.23	0.45
	2	0.06	0.27	0.46	0.14	0.47	0.63	0.29	0.61	0.71	0.37	0.68	0.73	0.44	0.69	0.73	0.53	0.72	0.73	0.06	0.23	0.45
	3	0.07	0.27	0.44	0.11	0.40	0.54	0.24	0.52	0.62	0.31	0.57	0.64	0.37	0.60	0.64	0.43	0.61	0.65	0.07	0.23	0.46
R15	**1**	0.06	0.31	0.51	0.13	0.47	0.64	0.25	0.58	0.71	0.40	0.69	0.73	0.46	0.70	0.73	0.55	0.70	0.74	0.05	0.16	0.34
	2	0.09	0.34	0.54	0.17	0.44	0.63	0.26	0.60	0.67	0.39	0.66	0.67	0.44	0.66	0.67	0.54	0.67	0.67	0.07	0.19	0.35
	3	0.07	0.38	0.57	0.20	0.56	0.73	0.33	0.67	0.78	0.45	0.75	0.80	0.54	0.76	0.80	0.62	0.76	0.81	0.07	0.18	0.35

k accuracy for varying k using h_0, h_1, and h_2 metrics. Over 95% h_2 accurate reconstructions are listed within top-15 of the ordered list. The reconstructions with over 92% h_0 accuracy are listed within the top-500. As the number of key tap points scattered over the AR keyboard increases, the shifted reconstructions are more easily detected with a higher $\mathbf{IM_2}$. For example, two centers representing the keys A and D are matched with the AR keyboard model's keys Q and E, causing the keys to be recovered as the keys above them, causing a *shift*. Hence, a higher h_0 accuracy is reached in the earlier top-k considerations when the number of key tap points is larger. Yet, having h_0 inaccurate results does not necessarily amount to entirely irrelevant results. Obtaining much higher h_1 and h_2 for such low h_0 values shows that the low pinpoint accuracy is compensated with higher h_1 and h_2 accuracy due to the existence of shifted reconstructions.

End-to-End Pipeline. In this section, we report the performance of the end-to-end pipeline for **E-mail**, **R5**, **R10**, and **R15** data sets. Using the **P108** data, we first train the two models; the CNN model for deep key tap localization and the linear regression for best-to-worst ordering. Next, we feed the **E-mail**, **R5**, **R10**, and **R15** data sets to the end-to-end pipeline, and obtain a set of reconstructions for each data set, ordered from best to worst.At all steps, we use the same hyper-parameters determined in the evaluation of the individual steps.

Table 5.2 shows the h_0, h_1, and h_2 accuracy obtained for each data set and each user considering the top-k with select values of k and the average accuracy obtained by randomly guessing each

Algorithm 4: The Monte Carlo experiment.

```
gt: Ground truth data
procedure MonteCarloExperiment(gt, length)
    initialize:
        random_ps ← {A, B, C, ..., Y, Z}^length
        accs ← [ ]
        hop ← 0, 1, 2
    for each s in random_ps
        accs.append( hhop_accuracy(gt, s, hop) )
     return average(accs)
```

character, i.e., brute-force attack.

Since the search space is vast for the brute-force attack and the accuracy is measured in a composite manner—using string difference, h-hop, and top-k—we obtain an approximation of the accuracy for brute-force attacks using a Monte-Carlo method [94]. We achieve this by generating 100000 random string samples for each text and averaging the accuracy to obtain the approximation. The sample size is chosen by monitoring the average, which converged at our choice point. The procedure for this computation can be found in Algorithm 4.

Overall, the results show that our model can successfully infer the text with a high accuracy. For edge cases where the pinpoint (h_0) accuracy is close to that of a random guess, the h_1 and h_2 accuracy shows that the inaccurate key estimations of our method appear close to the actual key, decreasing the search space for the adversary. For example, even though the pinpoint (h_0) accuracy of the first guess (Top-1) of our model for **R15** data of User-3 is the same as that of random guess, i.e., 0.07, our method performs much better in terms of h_1 (our method: 0.38 vs. random guess: 0.18) and h_2 (our method: 0.57 vs. random guess: 0.35) accuracy.

The maximum pinpoint (h_0) accuracy achieved over all reconstructions for **E-mail**, **R5**, **R10**, and **R15** is 68%, 76%, 87%, and 62%, respectively. As Table 5.2 shows, these reconstructions are also populated within the top-500 results. Comparing the results with those obtained using ground truth key tap points, we find clear evidence of the propagation of errors in keystroke detection to the end results, preventing it from achieving 100% accuracy.

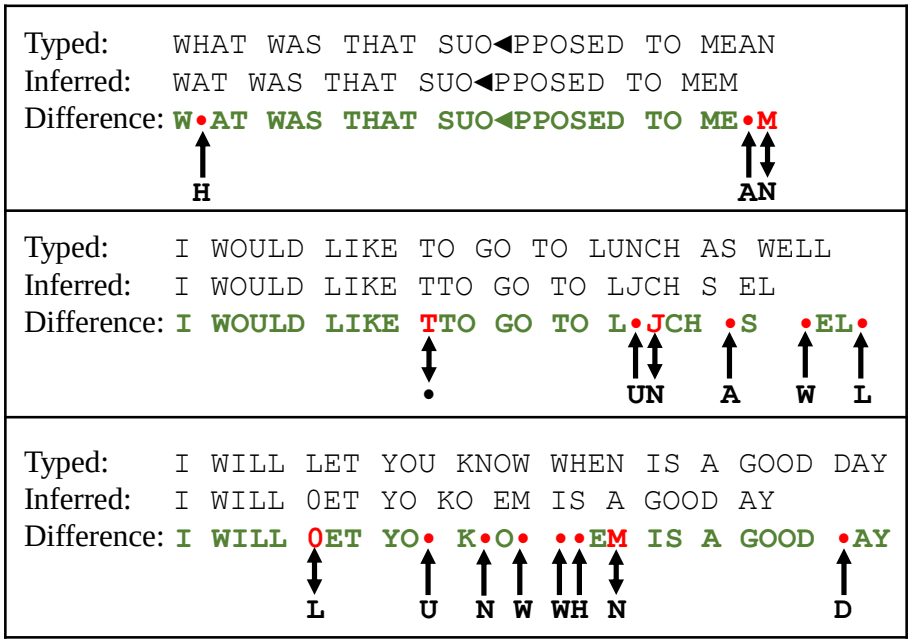


Figure 5.15: The inference found in the ordered list which is obtained utilizing the end-to-end pipeline on the E-mail. The end-to-end pipeline inference results on the first three sentences of the E-mail, which is found at the 11^{th} place in the ordered list, which were obtained by utilizing the end-to-end pipeline. The spaces are inserted for clearer visualization. Although the experiment did not include the usage of space key, spaces are inserted in the figure for clearer visualization of the difference.

An example random password of length 5 (alphabets) has 23.5 bits of entropy. We calculate the average entropy of the inferences on **R5** dataset considering correct inferences, considering the weighted probability distribution around the key corresponding to the character. The weights are assigned to the keys with respect to their physical proximity to the actual key. The entropy of the wrong characters is calculated by considering a uniform probability distribution. With our attack, we can reduce the entropy of the passwords by 10.77 bits, showing the impact of the attack.

Figure 5.15 shows the first three sentences of User-1's e-mail data and the inference of those sentences found at 11^{th} place in the ordered reconstructions list. In these inferred sentences, there are 12 missing, 1 extra, and 3 incorrectly identified letters. The missing letters are due to the false negatives in keystroke detection and require spotting and guessing for correction. The extra letter is due to the false positives in keystroke detection and needs to be spotted to fix. We can easily

correct the incorrectly identified letters by replacing them with the close neighbor keys.

Countermeasures

This study demonstrates the privacy risks to be of high significance, where the adversary only needs to observe a user's hands, which suffices to carry out the attack and recover a substantial amount of text input. In this section, we explore the defense mechanisms to counteract this attack.

One simple but highly effective defense would be randomizing the keys at each keyboard usage or after every key tap [88]. However, this defense would affect usability, specifically for long text inputs [85]. Also, if a random layout is fixed throughout a session, useful information, including the number of unique keys, their frequency, and the length of the text, could still be recovered.

Dynamic Positioning. Another defense is invalidating the adversary's assumption that the keyboard position remains fixed throughout a session. An instance of such a defense is by altering the position of the keyboard after each keystroke, i.e., dynamic positioning. We describe the implementation and conduct a security analysis to demonstrate the efficacy of this defense.

First, we fix the keyboard position and its depth since it is only effective when the change is significant, which causes unusable settings. On the two other dimensions, the translation vector $\vec{\mathbf{t}}$ is represented with two scalar parameters: its direction angle α and length d. After each keystroke, we form and use a translation vector $\vec{\mathbf{t}}$ by choosing α and d randomly from the ranges $[0, 2\pi]$ and $[r, 2r]$, respectively, where r is the diagonal length of an alphanumerical key. The motivation for upper-bounding d by $2r$ is to ensure that the usability is not reduced by much. The user would quickly locate the key position by taking the former keyboard position as a reference. With the lower bound of r, we guarantee that the keys will always change position by at least one hop between two keystrokes. Finally, randomly choosing d within $[r, 2r]$ range, the adversary is substantially challenged by the fact that any key's position on the keyboard may correspond to any key within the 2-hop neighborhood of that key in the next displacement.

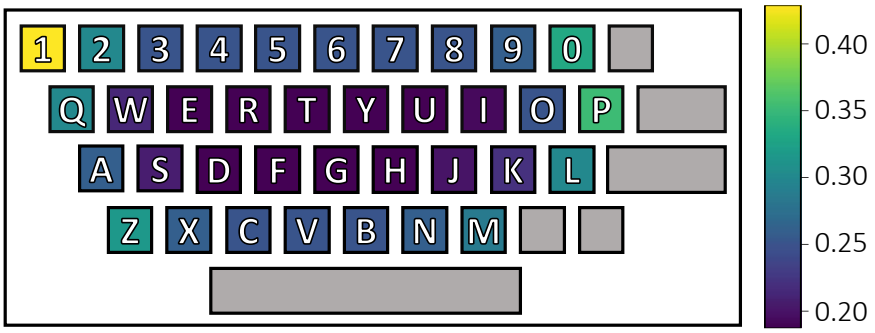


Figure 5.16: With the dynamically positioned keyboard defense, the probability for each key that the adversary's guess on the key is a true positive. Outward keys have more chance to be correctly guessed since they have fewer target positions after displacement that conflict with the other keys.

Analysis. For the security analysis, we consider a few assumptions to make the analysis tractable. Namely, we assume that: 1) the adversary achieves keystroke detection with 100% accuracy with perfect spatial localization, 2) the user always taps a key from the key's center point, 3) for each displacement, the translation vector $\vec{\mathbf{t}}$ is chosen to displace the keyboard such that the centers of the alphanumerical keys are moved to exactly one of the keys' centers within their 2-hop neighborhood. These assumptions are non-realistic and only put the adversary in a more favorable position while enabling us to sketch a lower bound on the security analysis. We emphasize that what we do next is a lower bound security analysis, and the exact security of this defense is more robust in reality, where the assumptions are relaxed.

When the initial position of the keyboard is known, to guess the key for the next keystroke, the adversary can exploit the fact that it can only be to one of the keys within the 2-hop neighborhood of the key formerly present at that location. Therefore, for each key k, the probability of successfully recovering k after a displacement is equal to the ratio of the number of positions k can appear to the number of distinct keys that can appear at these positions. We show the probabilities for each key in Figure 5.16, from which we derive the average probability as $P_{\text{next}} = 0.25$.

Since the adversary does not know the initial keyboard position, we consider the probability of successfully identifying the first key as $P_{\text{first}} = \frac{1}{36} = 0.028$, where 36 are the alphanumerical

keys. Using this analysis, the adversary's success probability to guess a text of length l is $P_l = P_{\text{first}} \times (P_{\text{next}})^{l-1} = 0.028 \times (0.25)^{l-1}$. For example, for texts of length 4 and 8, $P_{l=4} = 4.4 \times 10^{-4}$ and $P_{l=8} = 1.7 \times 10^{-6}$, which are low, showing the security of the countermeasure for practical scenarios such as inputting passwords.

Assuming a 100% keystroke detection accuracy, our method reaches at least 97% top-500 key identification accuracy. With this mechanism, we defend against the most decisive steps of our pipeline (i.e., key identification), which invalidates our attack scenario. However, similar to the randomized keyboard, we still note that the defense degrades the usability, making the attack a significant threat for users reluctant to use the defense mechanism.

We expect the dynamically-positioned keyboard to be more usable, especially compared to the randomized keyboard, since it preserves the keyboard layout and only changes the position of the keyboard one to two keys away. Moreover, this defense mechanism can easily generalize to any input method where 1) the adversary assumes a fixed keyboard position in a session, 2) a soft displaceable keyboard is used. Due to their prevalence and the many attacks targeting soft keyboards used in tablets [111], we envision them as a viable candidate for our defense. In our future work, we aim to study the usability aspects of the dynamically-positioned keyboard on AR/VR and tablet devices to validate this intuition. Moreover, we aim to study a dynamically-positioned keyboard implementation where the keyboard is randomly positioned within a frame after each keystroke, possibly providing more security by invalidating the 2-hop neighborhood clue.

Another defense is replacing the AR keyboard with other means of input or using it minimally in specific applications. For instance, Roesner et al. [104] suggest using a password manager to prevent the adversary from capturing login credentials while enabling the user to achieve authentication through simple inputs.

Summary of the Completed Work

In this work, we presented a keylogging inference attack targeting in air tapping keyboards for AR/VR HMDs by exploiting the observation that hands follow specific patterns while users are typing in the air. Substantiated by three different attack scenarios and threat models with reasonable capabilities, our attack provides up to 87% accuracy in inferring a random text of any length without requiring any special user profiling. We discuss various countermeasures to the attack, and show that they are a nontrivial task as they often conflict with usability.

CHAPTER 6: CONCLUSION AND FUTURE WORK

The rapid advancements in wearable technologies, such as fitness trackers, smartwatches, and AR/VR head mounted displays, brings lots of security and privacy concerns. In this dissertation, we explored in-depth the privacy dimensions of various wearable technologies through machine learning-enabled inference attacks and associated defenses. We namely examined three types of wearables through different features and techniques that allow for breaching the privacy of application semantics and contexts.

First, we examined wearable fitness trackers in Chapter 3. We performed a machine learning-enabled inference on the elevation profiles collected by wearable fitness trackers and showed that location privacy can be breached from the elevation profiles. Second, we examined the privacy of smartwatches in Chapter 4. We carried out a keylogging inference attack exploiting the acoustic emanations of the keyboards, which are captured by the microphone embedded in smartwatches. We showed that it is possible to recover the text typed on a keyboard through the smartwatches. Third, we examined the privacy of augmented reality head mounted displays in Chapter 5. We performed a keylogging inference attack facilitated by machine learning to recover the text typed on in-air tapping keyboards. Through the observations on geometric projections of the hand movements in air, we showed that the typed text on in-air tapping keyboards can be recovered.

The privacy examinations covered in this dissertation open up future research directions, including the following:

- In the third chapter, we recovered the location up to borough-level granularity. However, a borough can also be divided into smaller regions and those regions can be modelled through machine learning to increase the precision of the recovered location. In the future, it would be worthwhile to consider more advanced adversaries with extended mapping capabilities and how such adversaries would be capable of targeting victims to the exact location.

- In the fourth chapter, as in all keylogging attacks leveraging acoustic emanations, the as-

sumption is that the same keyboard models emanate the same acoustics. However, the effect of the production is never discussed. Therefore, another research question is naturally introduced: Do all keyboard models emanate the same acoustics? Do the changes in production date (keyboard generation) or material also change the acoustics they emanate? Exploring the impact of such manufacturing differences is an open direction that is worth exploring.

- In Chapter 4, we selected our equipment among the most popular smartwatches and keyboards on the market. However, to have a complete privacy examination, the study can be extended to cover more equipment. In this study, we also show that some keyboard models have more deterministic acoustic emanation. To this end, another research direction would be examining various keyboard characteristics such as material, surface dimension, etc. and the effects of these characteristics on the emanated acoustics.

- In the fifth chapter, we focused on a keyboard design that lays on a single plane and the position of the keyboard in the virtual environment is fixed. However, there are different AR keyboard designs, such as curved keyboard. It can be examined whether it is possible to perform the same kind of attack against such keyboards. Moreover, a word correction mechanism similar to the one in Chapter 4 can be added to this inference attack to obtain more accurate inferences.

APPENDIX A: COPYRIGHT INFORMATION

IEEE COPYRIGHT AND CONSENT FORM

To ensure uniformity of treatment among all contributors, other forms may not be substituted for this form, nor may any wording of the form be changed. This form is intended for original material submitted to the IEEE and must accompany any such material in order to be published by the IEEE. Please read the form carefully and keep a copy for your files.

Understanding the Potential Risks of Sharing Elevation Information on Fitness Applications
Ulku Meteriz, Necip Fazil Yildiran, Joongheon Kim, David Mohaisen
2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)

COPYRIGHT TRANSFER

The undersigned hereby assigns to The Institute of Electrical and Electronics Engineers, Incorporated (the "IEEE") all rights under copyright that may exist in and to: (a) the Work, including any revised or expanded derivative works submitted to the IEEE by the undersigned based on the Work; and (b) any associated written or multimedia components or other enhancements accompanying the Work.

GENERAL TERMS

1. The undersigned represents that he/she has the power and authority to make and execute this form.
2. The undersigned agrees to indemnify and hold harmless the IEEE from any damage or expense that may arise in the event of a breach of any of the warranties set forth above.
3. The undersigned agrees that publication with IEEE is subject to the policies and procedures of the IEEE PSPB Operations Manual.
4. In the event the above work is not accepted and published by the IEEE or is withdrawn by the author(s) before acceptance by the IEEE, the foregoing copyright transfer shall be null and void. In this case, IEEE will retain a copy of the manuscript for internal administrative/record-keeping purposes.
5. For jointly authored Works, all joint authors should sign, or one of the authors should sign as authorized agent for the others.
6. The author hereby warrants that the Work and Presentation (collectively, the "Materials") are original and that he/she is the author of the Materials. To the extent the Materials incorporate text passages, figures, data or other material from the works of others, the author has obtained any necessary permissions. Where necessary, the author has obtained all third party permissions and consents to grant the license above and has provided copies of such permissions and consents to IEEE

You have indicated that you DO wish to have video/audio recordings made of your conference presentation under terms and conditions set forth in "Consent and Release."

CONSENT AND RELEASE

1. In the event the author makes a presentation based upon the Work at a conference hosted or sponsored in whole or in part by the IEEE, the author, in consideration for his/her participation in the conference, hereby grants the IEEE the unlimited, worldwide, irrevocable permission to use, distribute, publish, license, exhibit, record, digitize, broadcast, reproduce and archive, in any format or medium, whether now known or hereafter developed: (a) his/her presentation and comments at the conference; (b) any written materials or multimedia files used in connection with his/her presentation; and (c) any recorded interviews of him/her (collectively, the "Presentation"). The permission granted includes the transcription and reproduction of the Presentation for inclusion in products sold or distributed by IEEE and live or recorded broadcast of the Presentation during or after the conference.
2. In connection with the permission granted in Section 1, the author hereby grants IEEE the unlimited, worldwide, irrevocable right to use his/her name, picture, likeness, voice and biographical information as part of the advertisement, distribution and sale of products incorporating the Work or Presentation, and releases IEEE from any claim based on right of privacy or publicity.

BY TYPING IN YOUR FULL NAME BELOW AND CLICKING THE SUBMIT BUTTON, YOU CERTIFY THAT SUCH ACTION CONSTITUTES YOUR ELECTRONIC SIGNATURE TO THIS FORM IN ACCORDANCE WITH UNITED STATES LAW, WHICH AUTHORIZES ELECTRONIC SIGNATURE BY AUTHENTICATED REQUEST FROM A USER OVER THE INTERNET AS A VALID SUBSTITUTE FOR A WRITTEN SIGNATURE.

Ulku Meteriz	16-04-2020
Signature	**Date (dd-mm-yyyy)**

Information for Authors

AUTHOR RESPONSIBILITIES

The IEEE distributes its technical publications throughout the world and wants to ensure that the material submitted to its publications is properly available to the readership of those publications. Authors must ensure that their Work meets the requirements as stated in section 8.2.1 of the IEEE PSPB Operations Manual, including provisions covering originality, authorship, author responsibilities and author misconduct. More information on IEEE's publishing policies may be found at http://www.ieee.org/publications_standards/publications/rights/authorrightsresponsibilities.html Authors are advised especially of IEEE PSPB Operations Manual section 8.2.1.B12: "It is the responsibility of the authors, not the IEEE, to determine whether disclosure of their material requires the prior consent of other parties and, if so, to obtain it." Authors are also advised of IEEE PSPB Operations Manual section 8.1.1B: "Statements and opinions given in work published by the IEEE are the expression of the authors."

RETAINED RIGHTS/TERMS AND CONDITIONS

- Authors/employers retain all proprietary rights in any process, procedure, or article of manufacture described in the Work.
- Authors/employers may reproduce or authorize others to reproduce the Work, material extracted verbatim from the Work, or derivative works for the author's personal use or for company use, provided that the source and the IEEE copyright notice are indicated, the copies are not used in any way that implies IEEE endorsement of a product or service of any employer, and the copies themselves are not offered for sale.
- Although authors are permitted to re-use all or portions of the Work in other works, this does not include granting third-party requests for reprinting, republishing, or other types of re-use.The IEEE Intellectual Property Rights office must handle all such third-party requests.
- Authors whose work was performed under a grant from a government funding agency are free to fulfill any deposit mandates from that funding agency.

AUTHOR ONLINE USE

- **Personal Servers**. Authors and/or their employers shall have the right to post the accepted version of IEEE-copyrighted articles on their own personal servers or the servers of their institutions or employers without permission from IEEE, provided that the posted version includes a prominently displayed IEEE copyright notice and, when published, a full citation to the original IEEE publication, including a link to the article abstract in IEEE Xplore. Authors shall not post the final, published versions of their papers.
- **Classroom or Internal Training Use.** An author is expressly permitted to post any portion of the accepted version of his/her own IEEE-copyrighted articles on the author's personal web site or the servers of the author's institution or company in connection with the author's teaching, training, or work responsibilities, provided that the appropriate copyright, credit, and reuse notices appear prominently with the posted material. Examples of permitted uses are lecture materials, course packs, e-reserves, conference presentations, or in-house training courses.
- **Electronic Preprints.** Before submitting an article to an IEEE publication, authors frequently post their manuscripts to their own web site, their employer's site, or to another server that invites constructive comment from colleagues. Upon submission of an article to IEEE, an author is required to transfer copyright in the article to IEEE, and the author must update any previously posted version of the article with a prominently displayed IEEE copyright notice. Upon publication of an article by the IEEE, the author must replace any previously posted electronic versions of the article with either (1) the full citation to the

ACM Copyright and Audio/Video Release

Title of the Work: SIA: Smartwatch-Enabled Inference Attacks on Physical Keyboards Using Acoustic Signals
Submission ID:wpes14

Author/Presenter(s): Ulku Meteriz,Necip Fazil Yildiran,David Mohaisen

Type of material:full paper

Publication and/or Conference Name: WPES '21: 20th Workshop on Privacy in the Electronic Society Proceedings

I. Copyright Transfer, Reserved Rights and Permitted Uses

* Your Copyright Transfer is conditional upon you agreeing to the terms set out below.

Copyright to the Work and to any supplemental files integral to the Work which are submitted with it for review and publication such as an extended proof, a PowerPoint outline, or appendices that may exceed a printed page limit, (including without limitation, the right to publish the Work in whole or in part in any and all forms of media, now or hereafter known) is hereby transferred to the ACM (for Government work, to the extent transferable) effective as of the date of this agreement, on the understanding that the Work has been accepted for publication by ACM.

Reserved Rights and Permitted Uses

(a) All rights and permissions the author has not granted to ACM are reserved to the Owner, including all other proprietary rights such as patent or trademark rights.

(b) Furthermore, notwithstanding the exclusive rights the Owner has granted to ACM, Owner shall have the right to do the following:

(i) Reuse any portion of the Work, without fee, in any future works written or edited by the Author, including books, lectures and presentations in any and all media.

(ii) Create a "Major Revision" which is wholly owned by the author

(iii) Post the Accepted Version of the Work on (1) the Author's home page, (2) the Owner's institutional repository, (3) any repository legally mandated by an agency funding the research on which the Work is based, and (4) any non-commercial repository or aggregation that does not duplicate ACM tables of contents, i.e., whose patterns of links do not substantially duplicate an ACM-copyrighted volume or issue. Non-commercial repositories are here understood as repositories owned by non-profit organizations that do not charge a fee for accessing deposited articles and that do not sell advertising or otherwise profit from serving articles.

(iv) Post an "Author-Izer" link enabling free downloads of the Version of Record in the ACM Digital Library on (1) the Author's home page or (2) the Owner's institutional repository;

(v) Prior to commencement of the ACM peer review process, post the version of the Work as submitted to ACM ("Submitted Version" or any earlier versions) to non-peer reviewed servers;

(vi) Make free distributions of the final published Version of Record internally to the Owner's employees, if applicable;

(vii) Make free distributions of the published Version of Record for Classroom and Personal Use;

(viii) Bundle the Work in any of Owner's software distributions; and

(ix) Use any Auxiliary Material independent from the Work. (x) If your paper is withdrawn before it is published in the ACM Digital Library, the rights revert back to the author(s).

When preparing your paper for submission using the ACM TeX templates, the rights and permissions information and the bibliographic strip must appear on the lower left hand portion of the first page.

The new ACM Consolidated TeX template Version 1.3 and above automatically creates and positions these text blocks for you based on the code snippet which is system-generated based on your rights management choice and this particular conference.

NOTE: For authors using the ACM Microsoft Word Master Article Template and Publication Workflow, The ACM Publishing System (TAPS) will add the rights statement to your papers for you. Please check with your conference contact for information regarding submitting your source file(s) for processing.

Please put the following LaTeX commands in the preamble of your document - i.e., before \begin{document}:

```
\copyrightyear{2021}
\acmYear{2021}
\setcopyright{acmcopyright}\acmConference[WPES '21]{Proceedings of the 20th Workshop on Privacy in the Electronic Society}{November 15, 2021}{Virtual Event, Republic of Korea}
\acmBooktitle{Proceedings of the 20th Workshop on Privacy in the Electronic Society (WPES '21), November 15, 2021, Virtual Event, Republic of Korea}
\acmPrice{15.00}
\acmDOI{10.1145/3463676.3485607}
\acmISBN{978-1-4503-8527-5/21/11}
```

NOTE: For authors using the ACM Microsoft Word Master Article Template and Publication Workflow, The ACM Publishing System (TAPS) will add the rights statement to your papers for you. Please check with your conference contact for information regarding submitting your source file(s) for processing.

If you are using the ACM Interim Microsoft Word template, or still using or older versions of the ACM SIGCHI template, you must copy and paste the following text block into your document as per the instructions provided with the templates you are using:

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific

permission and/or a fee. Request permissions from Permissions@acm.org.

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

NOTE: Make sure to include your article's DOI as part of the bibstrip data; DOIs will be registered and become active shortly after publication in the ACM Digital Library. Once you have your camera ready copy ready, please send your source files and PDF to your event contact for processing.

☑ A. Assent to Assignment. I hereby represent and warrant that I am the sole owner (or authorized agent of the copyright owner(s)), with the exception of third party materials detailed in section III below. I have obtained permission for any third-party material included in the Work.

☐ B. Declaration for Government Work. I am an employee of the National Government of my country/region and my Government claims rights to this work, or it is not copyrightable (Government work is classified as Public Domain in U.S. only)

II. Permission For Conference Recording and Distribution

* Your Audio/Video Release is conditional upon you agreeing to the terms set out below.

I hereby grant permission for ACM to include my name, likeness, presentation and comments in any and all forms, for the Conference and/or Publication.

I further grant permission for ACM to record and/or transcribe and reproduce my presentation as part of the ACM Digital Library, and to distribute the same for sale in complete or partial form as part of an ACM product on CD-ROM, DVD, webcast, USB device, streaming video or any other media format now or hereafter known.

I understand that my presentation will not be sold separately as a stand-alone product without my direct consent. Accordingly, I give ACM the right to use my image, voice, pronouncements, likeness, and my name, and any biographical material submitted by me, in connection with the Conference and/or Publication, whether used in excerpts or in full, for distribution described above and for any associated advertising or exhibition.

Do you agree to the above Audio/Video Release? ◉ Yes ○ No

III. Auxiliary Material

Do you have any Auxiliary Materials? ○ Yes ◉ No

IV. Third Party Materials

In the event that any materials used in my presentation or Auxiliary Materials contain the work of third-party individuals or organizations (including copyrighted music or movie excerpts or anything not owned by me), I understand that it is my responsibility to secure any necessary permissions and/or licenses for print and/or digital publication, and cite or attach them below.

◉ We/I have not used third-party material.
○ We/I have used third-party materials and have necessary permissions.

V. Artistic Images

If your paper includes images that were created for any purpose other than this paper and to which you or your employer claim copyright, you must complete Part V and be sure to include a notice of copyright with each such image in the paper.

◉ We/I do not have any artistic images.
○ We/I have any artistic images.

VI. Representations, Warranties and Covenants

The undersigned hereby represents, warrants and covenants as follows:

(a) Owner is the sole owner or authorized agent of Owner(s) of the Work;

(b) The undersigned is authorized to enter into this Agreement and grant the rights included in this license to ACM;

(c) The Work is original and does not infringe the rights of any third party; all permissions for use of third-party materials consistent in scope and duration with the rights granted to ACM have been obtained, copies of such permissions have been provided to ACM, and the Work as submitted to ACM clearly and accurately indicates the credit to the proprietors of any such third-party materials (including any applicable copyright notice), or will be revised to indicate such credit;

(d) The Work has not been published except for informal postings on non-peer reviewed servers, and Owner covenants to use best efforts to place ACM DOI pointers on any such prior postings;

(e) The Auxiliary Materials, if any, contain no malicious code, virus, trojan horse or other software routines or hardware components designed to permit unauthorized access or to disable, erase or otherwise harm any computer systems or software; and

(f) The Artistic Images, if any, are clearly and accurately noted as such (including any applicable copyright notice) in the Submitted Version.

☑ I agree to the Representations, Warranties and Covenants

Funding Agents

1. National Research Foundation of Korea award number(s): NRF-2016K1A1A2912757

DATE: **09/08/2021** sent to amohaisen@gmail.com at **18:09:41**

IEEE COPYRIGHT AND CONSENT FORM

To ensure uniformity of treatment among all contributors, other forms may not be substituted for this form, nor may any wording of the form be changed. This form is intended for original material submitted to the IEEE and must accompany any such material in order to be published by the IEEE. Please read the form carefully and keep a copy for your files.

A Keylogging Inference Attack on Air-Tapping Keyboards in Virtual Environments
\u00DClk\u00FC Meteriz-Y\u0131ld\u0131ran, Necip Faz\u0131l Y\u0131ld\u0131ran, Amro Awad, David Mohaisen
2022 IEEE on Conference Virtual Reality and 3D User Interfaces (VR)

COPYRIGHT TRANSFER

The undersigned hereby assigns to The Institute of Electrical and Electronics Engineers, Incorporated (the "IEEE") all rights under copyright that may exist in and to: (a) the Work, including any revised or expanded derivative works submitted to the IEEE by the undersigned based on the Work; and (b) any associated written or multimedia components or other enhancements accompanying the Work.

GENERAL TERMS

1. The undersigned represents that he/she has the power and authority to make and execute this form.
2. The undersigned agrees to indemnify and hold harmless the IEEE from any damage or expense that may arise in the event of a breach of any of the warranties set forth above.
3. The undersigned agrees that publication with IEEE is subject to the policies and procedures of the IEEE PSPB Operations Manual.
4. In the event the above work is not accepted and published by the IEEE or is withdrawn by the author(s) before acceptance by the IEEE, the foregoing copyright transfer shall be null and void. In this case, IEEE will retain a copy of the manuscript for internal administrative/record-keeping purposes.
5. For jointly authored Works, all joint authors should sign, or one of the authors should sign as authorized agent for the others.
6. The author hereby warrants that the Work and Presentation (collectively, the "Materials") are original and that he/she is the author of the Materials. To the extent the Materials incorporate text passages, figures, data or other material from the works of others, the author has obtained any necessary permissions. Where necessary, the author has obtained all third party permissions and consents to grant the license above and has provided copies of such permissions and consents to IEEE

You have indicated that you DO wish to have video/audio recordings made of your conference presentation under terms and conditions set forth in "Consent and Release."

CONSENT AND RELEASE

1. In the event the author makes a presentation based upon the Work at a conference hosted or sponsored in whole or in part by the IEEE, the author, in consideration for his/her participation in the conference, hereby grants the IEEE the unlimited, worldwide, irrevocable permission to use, distribute, publish, license, exhibit, record, digitize, broadcast, reproduce and archive, in any format or medium, whether now known or hereafter developed: (a) his/her presentation and comments at the conference; (b) any written materials or multimedia files used in connection with his/her presentation; and (c) any recorded interviews of him/her (collectively, the "Presentation"). The permission granted includes the transcription and reproduction of the Presentation for inclusion in products sold or distributed by IEEE and live or recorded broadcast of the Presentation during or after the conference.
2. In connection with the permission granted in Section 1, the author hereby grants IEEE the unlimited, worldwide, irrevocable right to use his/her name, picture, likeness, voice and biographical information as part of the advertisement, distribution and sale of products incorporating the Work or Presentation, and releases IEEE from any claim based on right of privacy or publicity.

BY TYPING IN YOUR FULL NAME BELOW AND CLICKING THE SUBMIT BUTTON, YOU CERTIFY THAT SUCH ACTION CONSTITUTES YOUR ELECTRONIC SIGNATURE TO THIS FORM IN ACCORDANCE WITH UNITED STATES LAW, WHICH AUTHORIZES ELECTRONIC SIGNATURE BY AUTHENTICATED REQUEST FROM A USER OVER THE INTERNET AS A VALID SUBSTITUTE FOR A WRITTEN SIGNATURE.

Ulku Meteriz-Yildiran	20-01-2022
Signature	**Date (dd-mm-yyyy)**

Information for Authors

AUTHOR RESPONSIBILITIES

The IEEE distributes its technical publications throughout the world and wants to ensure that the material submitted to its publications is properly available to the readership of those publications. Authors must ensure that their Work meets the requirements as stated in section 8.2.1 of the IEEE PSPB Operations Manual, including provisions covering originality, authorship, author responsibilities and author misconduct. More information on IEEE's publishing policies may be found at http://www.ieee.org/publications_standards/publications/rights/authorrightsresponsibilities.html Authors are advised especially of IEEE PSPB Operations Manual section 8.2.1.B12: "It is the responsibility of the authors, not the IEEE, to determine whether disclosure of their material requires the prior consent of other parties and, if so, to obtain it." Authors are also advised of IEEE PSPB Operations Manual section 8.1.1B: "Statements and opinions given in work published by the IEEE are the expression of the authors."

RETAINED RIGHTS/TERMS AND CONDITIONS

- Authors/employers retain all proprietary rights in any process, procedure, or article of manufacture described in the Work.
- Authors/employers may reproduce or authorize others to reproduce the Work, material extracted verbatim from the Work, or derivative works for the author's personal use or for company use, provided that the source and the IEEE copyright notice are indicated, the copies are not used in any way that implies IEEE endorsement of a product or service of any employer, and the copies themselves are not offered for sale.
- Although authors are permitted to re-use all or portions of the Work in other works, this does not include granting third-party requests for reprinting, republishing, or other types of re-use.The IEEE Intellectual Property Rights office must handle all such third-party requests.
- Authors whose work was performed under a grant from a government funding agency are free to fulfill any deposit mandates from that funding agency.

AUTHOR ONLINE USE

- **Personal Servers**. Authors and/or their employers shall have the right to post the accepted version of IEEE-copyrighted articles on their own personal servers or the servers of their institutions or employers without permission from IEEE, provided that the posted version includes a prominently displayed IEEE copyright notice and, when published, a full citation to the original IEEE publication, including a link to the article abstract in IEEE Xplore. Authors shall not post the final, published versions of their papers.
- **Classroom or Internal Training Use.** An author is expressly permitted to post any portion of the accepted version of his/her own IEEE-copyrighted articles on the author's personal web site or the servers of the author's institution or company in connection with the author's teaching, training, or work responsibilities, provided that the appropriate copyright, credit, and reuse notices appear prominently with the posted material. Examples of permitted uses are lecture materials, course packs, e-reserves, conference presentations, or in-house training courses.
- **Electronic Preprints.** Before submitting an article to an IEEE publication, authors frequently post their manuscripts to their own web site, their employer's site, or to another server that invites constructive comment from colleagues. Upon submission of an article to IEEE, an author is required to transfer copyright in the article to IEEE, and the author must update any previously posted version of the article with a prominently displayed IEEE copyright notice. Upon publication of an article by the IEEE, the author must replace any previously posted electronic versions of the article with either (1) the full citation to the

IEEE work with a Digital Object Identifier (DOI) or link to the article abstract in IEEE Xplore, or (2) the accepted version only (not the IEEE-published version), including the IEEE copyright notice and full citation, with a link to the final, published article in IEEE Xplore.

Questions about the submission of the form or manuscript must be sent to the publication's editor.
Please direct all questions about IEEE copyright policy to:
IEEE Intellectual Property Rights Office, copyrights@ieee.org, +1-732-562-3966

IEEE COPYRIGHT AND CONSENT FORM

To ensure uniformity of treatment among all contributors, other forms may not be substituted for this form, nor may any wording of the form be changed. This form is intended for original material submitted to the IEEE and must accompany any such material in order to be published by the IEEE. Please read the form carefully and keep a copy for your files.

AiRType: An Air-Tapping Keyboard for Augmented Reality Environments
Necip Faz\u0131l Y\u0131ld\u0131ran, \u00DClk\u00FC Meteriz-Y\u0131ld\u0131ran, David Mohaisen
2022 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)

COPYRIGHT TRANSFER

The undersigned hereby assigns to The Institute of Electrical and Electronics Engineers, Incorporated (the "IEEE") all rights under copyright that may exist in and to: (a) the Work, including any revised or expanded derivative works submitted to the IEEE by the undersigned based on the Work; and (b) any associated written or multimedia components or other enhancements accompanying the Work.

GENERAL TERMS

1. The undersigned represents that he/she has the power and authority to make and execute this form.
2. The undersigned agrees to indemnify and hold harmless the IEEE from any damage or expense that may arise in the event of a breach of any of the warranties set forth above.
3. The undersigned agrees that publication with IEEE is subject to the policies and procedures of the IEEE PSPB Operations Manual.
4. In the event the above work is not accepted and published by the IEEE or is withdrawn by the author(s) before acceptance by the IEEE, the foregoing copyright transfer shall be null and void. In this case, IEEE will retain a copy of the manuscript for internal administrative/record-keeping purposes.
5. For jointly authored Works, all joint authors should sign, or one of the authors should sign as authorized agent for the others.
6. The author hereby warrants that the Work and Presentation (collectively, the "Materials") are original and that he/she is the author of the Materials. To the extent the Materials incorporate text passages, figures, data or other material from the works of others, the author has obtained any necessary permissions. Where necessary, the author has obtained all third party permissions and consents to grant the license above and has provided copies of such permissions and consents to IEEE

You have indicated that you DO wish to have video/audio recordings made of your conference presentation under terms and conditions set forth in "Consent and Release."

CONSENT AND RELEASE

1. In the event the author makes a presentation based upon the Work at a conference hosted or sponsored in whole or in part by the IEEE, the author, in consideration for his/her participation in the conference, hereby grants the IEEE the unlimited, worldwide, irrevocable permission to use, distribute, publish, license, exhibit, record, digitize, broadcast, reproduce and archive, in any format or medium, whether now known or hereafter developed: (a) his/her presentation and comments at the conference; (b) any written materials or multimedia files used in connection with his/her presentation; and (c) any recorded interviews of him/her (collectively, the "Presentation"). The permission granted includes the transcription and reproduction of the Presentation for inclusion in products sold or distributed by IEEE and live or recorded broadcast of the Presentation during or after the conference.
2. In connection with the permission granted in Section 1, the author hereby grants IEEE the unlimited, worldwide, irrevocable right to use his/her name, picture, likeness, voice and biographical information as part of the advertisement, distribution and sale of products incorporating the Work or Presentation, and releases IEEE from any claim based on right of privacy or publicity.

BY TYPING IN YOUR FULL NAME BELOW AND CLICKING THE SUBMIT BUTTON, YOU CERTIFY THAT SUCH ACTION CONSTITUTES YOUR ELECTRONIC SIGNATURE TO THIS FORM IN ACCORDANCE WITH UNITED STATES LAW, WHICH AUTHORIZES ELECTRONIC SIGNATURE BY AUTHENTICATED REQUEST FROM A USER OVER THE INTERNET AS A VALID SUBSTITUTE FOR A WRITTEN SIGNATURE.

Ulku Meteriz-Yildiran	21-01-2022
Signature	**Date (dd-mm-yyyy)**

Information for Authors

AUTHOR RESPONSIBILITIES

The IEEE distributes its technical publications throughout the world and wants to ensure that the material submitted to its publications is properly available to the readership of those publications. Authors must ensure that their Work meets the requirements as stated in section 8.2.1 of the IEEE PSPB Operations Manual, including provisions covering originality, authorship, author responsibilities and author misconduct. More information on IEEE's publishing policies may be found at http://www.ieee.org/publications_standards/publications/rights/authorrightsresponsibilities.html Authors are advised especially of IEEE PSPB Operations Manual section 8.2.1.B12: "It is the responsibility of the authors, not the IEEE, to determine whether disclosure of their material requires the prior consent of other parties and, if so, to obtain it." Authors are also advised of IEEE PSPB Operations Manual section 8.1.1B: "Statements and opinions given in work published by the IEEE are the expression of the authors."

RETAINED RIGHTS/TERMS AND CONDITIONS

- Authors/employers retain all proprietary rights in any process, procedure, or article of manufacture described in the Work.
- Authors/employers may reproduce or authorize others to reproduce the Work, material extracted verbatim from the Work, or derivative works for the author's personal use or for company use, provided that the source and the IEEE copyright notice are indicated, the copies are not used in any way that implies IEEE endorsement of a product or service of any employer, and the copies themselves are not offered for sale.
- Although authors are permitted to re-use all or portions of the Work in other works, this does not include granting third-party requests for reprinting, republishing, or other types of re-use.The IEEE Intellectual Property Rights office must handle all such third-party requests.
- Authors whose work was performed under a grant from a government funding agency are free to fulfill any deposit mandates from that funding agency.

AUTHOR ONLINE USE

- **Personal Servers**. Authors and/or their employers shall have the right to post the accepted version of IEEE-copyrighted articles on their own personal servers or the servers of their institutions or employers without permission from IEEE, provided that the posted version includes a prominently displayed IEEE copyright notice and, when published, a full citation to the original IEEE publication, including a link to the article abstract in IEEE Xplore. Authors shall not post the final, published versions of their papers.
- **Classroom or Internal Training Use.** An author is expressly permitted to post any portion of the accepted version of his/her own IEEE-copyrighted articles on the author's personal web site or the servers of the author's institution or company in connection with the author's teaching, training, or work responsibilities, provided that the appropriate copyright, credit, and reuse notices appear prominently with the posted material. Examples of permitted uses are lecture materials, course packs, e-reserves, conference presentations, or in-house training courses.
- **Electronic Preprints.** Before submitting an article to an IEEE publication, authors frequently post their manuscripts to their own web site, their employer's site, or to another server that invites constructive comment from colleagues. Upon submission of an article to IEEE, an author is required to transfer copyright in the article to IEEE, and the author must update any previously posted version of the article with a prominently displayed IEEE copyright notice. Upon publication of an article by the IEEE, the author must replace any previously posted electronic versions of the article with either (1) the full citation to the

APPENDIX B: IRB LETTERS

UNIVERSITY OF CENTRAL FLORIDA

Institutional Review Board
FWA00000351
IRB00001138, IRB00012110
Office of Research
12201 Research Parkway
Orlando, FL 32826-3246

APPROVAL

September 15, 2021

Dear Ulku Meteriz:

On 9/15/2021, the IRB reviewed the following submission:

Type of Review:	Initial Study
Title:	SIA: Smartwatch-Enabled Inference Attacks on Physical Keyboards Using Acoustic Signals
Investigator:	Ulku Meteriz
IRB ID:	STUDY00003404
Funding:	None
Grant ID:	None
IND, IDE, or HDE:	None
Documents Reviewed:	• HRP-251 Form, Category: Faculty Research Approval; • email invite, Category: Recruitment Materials; • HRP-502 - TEMPLATE CONSENT DOCUMENT Adult.pdf, Category: Consent Form; • HRP-503-TEMPLATE-Protocol (1).docx, Category: IRB Protocol; • instructions, Category: Other; • texts_to_be_typed.pdf, Category: Other

The IRB approved the protocol from 9/15/2021.

In conducting this protocol, you are required to follow the requirements listed in the Investigator Manual (HRP-103), which can be found by navigating to the IRB Library within the IRB system. Guidance on submitting Modifications and a Continuing Review or Administrative Check-in are detailed in the manual. When you have completed your research, please submit a Study Closure request so that IRB records will be accurate.

If you have any questions, please contact the UCF IRB at 407-823-2901 or irb@ucf.edu. Please include your project title and IRB number in all correspondence with this office.

Sincerely,

Katie Kilgore
Designated Reviewer

UNIVERSITY OF CENTRAL FLORIDA

Institutional Review Board
FWA00000351
IRB00001138, IRB00012110
Office of Research
12201 Research Parkway
Orlando, FL 32826-3246

APPROVAL

September 15, 2021

Dear Ulku Meteriz:

On 9/15/2021, the IRB reviewed the following submission:

Type of Review:	Initial Study, Category 6 and 7
Title:	The Usability of Virtual Keyboards in Augmented Reality Environments and Their Vulnerability to Side Channel Input Inference Attacks
Investigator:	Ulku Meteriz
IRB ID:	STUDY00003428
Funding:	Name: Inha University
Grant ID:	
IND, IDE, or HDE:	None
Documents Reviewed:	• HRP-251- FORM - dm.pdf, Category: Faculty Research Approval; • AR Keyboard.mp4, Category: Other; • Email-Invitation.pdf, Category: Recruitment Materials; • HRP-502 - TEMPLATE CONSENT DOCUMENT Adult.pdf, Category: Consent Form; • HRP-503-TEMPLATE-Protocol (1).docx, Category: IRB Protocol; • SUS Questionnaire for AR Keyboard.pdf, Category: Survey / Questionnaire; • texts_to_be_typed.pdf, Category: Other

The IRB approved the protocol from 9/15/2021 to .

In conducting this protocol, you are required to follow the requirements listed in the Investigator Manual (HRP-103), which can be found by navigating to the IRB Library within the IRB system. Guidance on submitting Modifications and a Continuing Review or Administrative Check-in are detailed in the manual. When you have completed your research, please submit a Study Closure request so that IRB records will be accurate.

If you have any questions, please contact the UCF IRB at 407-823-2901 or irb@ucf.edu. Please include your project title and IRB number in all correspondence with this office.

Sincerely,

Katie Kilgore
Designated Reviewer

LIST OF REFERENCES

[1] Cyclist who had five bikes stolen says thieves are looking for quick times on strava to try and find high-end bikes – warns other users to check their privacy settings. https://bit.ly/3gp8xtp. Accessed: 2021-04-20.

[2] Cyclists warned to beware sharing data on ride-tracking apps. https://bit.ly/3g66fRe. Accessed: 2021-04-20.

[3] Fitness tracking app strava gives away location of secret us army bases. https://bit.ly/3vdfVNT. Accessed: 2021-04-20.

[4] Hide map. https://bit.ly/3gaqKwe. Accessed: 2020-01-11.

[5] How do i remove display map from my activity list? https://bit.ly/2TPYdDf. Accessed: 2020-01-11.

[6] Map privacy. https://bit.ly/2TgWTJu. Accessed: 2020-01-11.

[7] Organised criminals using fitness apps to track expensive bicycles and equipment. https://bit.ly/2SlR11n. Accessed: 2021-04-20.

[8] Privacy setting to hide activity map from non-followers. https://bit.ly/3cu4ET9. Accessed: 2020-01-11.

[9] Runtastic deserves more 'heat' than strava! https://bit.ly/2SqtCvD. Accessed: 2021-04-20.

[10] Strava global heatmap. https://www.strava.com/heatmap. Accessed: 2019-03-20.

[11] Strava hone app led thieves to my £12,000 bike collection. https://bit.ly/354wcKB. Accessed: 2021-04-20.

[12] U.s. soldiers are revealing sensitive and dangerous information by jogging. https://goo.gl/tiM5VU. Accessed: 2021-04-20.

[13] Nefes data kit untethers USB devices for wireless VR setups. https://www.tomshardware.com/news/nefes-data-kit-wireless-vr,34162.html, Apr. 2017. Accessed: 2020-03-12.

[14] Keyboard mixed reality. https://docs.microsoft.com/en-us/windows/mixed-reality/design/keyboard, 2019. Accessed: 2021-05-07.

[15] Microsoft mixed reality docs, gestures, air tap. https://docs.microsoft.com/en-us/windows/mixed-reality/gestures#air-tap, 2019. Accessed: 12/12/2019.

[16] Augmented reality company builds rapid deployment kits in response to covid-19, Mar 2020.

[17] Keystroke — definition of keystroke. https://www.lexico.com/en/definition/keystroke, 2020. Accessed: 2020-03-12.

[18] Leap motion docs, key tap gesture. https://developer-archive.leapmotion.com/documentation/python/api/Leap.KeyTapGesture.html, 2020. Accessed: 12/12/2019.

[19] Magic leap 1 — magic leap. https://www.magicleap.com/magic-leap-1, 2020. Accessed: 2020-03-13.

[20] Microsoft hololens — mixed reality technology for business. https://www.microsoft.com/en-us/hololens, 2020. Accessed: 2020-03-13.

[21] PCL - point cloud library. http://pointclouds.org/, 2020. Accessed: 2020-03-02.

[22] Point cloud library(PCL): pcl::registration:TransformationEstimationSVD. http://docs.pointclouds.org/1.7.0/, 2020. Accessed: 2020-03-02.

[23] Smartwatch market - growth, trends, forecasts (2020 - 2025), 2020.

[24] Zero iteration — magic leap. https://developer.magicleap.com/learn/guides/lab-zi, 2020. Accessed: 2020-03-12.

[25] Tracking — leap motion controller. https://www.ultraleap.com/product/leap-motion-controller/, 2021. Accessed: 2021-05-07.

[26] A. I. Abdelmoty and F. Alrayes. Towards understanding location privacy awareness on geo-social networks. *ISPRS International Journal of Geo-Information*, 6(4), 2017.

[27] A. Aktypi, J. Nurse, and M. Goldsmith. Unwinding ariadne's identity thread: Privacy risks with fitness trackers and online social networks. pages 1–11. Association for Computing Machinery, 2017.

[28] M. Al-Sharrah, A. Salman, and I. Ahmad. Watch your smartwatch. In *2018 International Conference on Computing Sciences and Engineering (ICCSE)*, pages 1–5, 2018.

[29] K. Ali, A. X. Liu, W. Wang, and M. Shahzad. Keystroke recognition using wifi signals. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, MobiCom '15, page 90–102, New York, NY, USA, 2015. Association for Computing Machinery.

[30] S. A. Anand and N. Saxena. A sound for a sound: Mitigating acoustic side channel attacks on password keystrokes with active sounds. In J. Grossklags and B. Preneel, editors, *Financial Cryptography and Data Security*, pages 346–364, Berlin, Heidelberg, 2017. Springer Berlin Heidelberg.

[31] D. Asonov and R. Agrawal. Keyboard acoustic emanations. In *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*, pages 3–11, May 2004.

[32] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre. Recent advances in augmented reality. *IEEE Comput. Graph. Appl.*, 21(6):34–47, Nov. 2001.

[33] R. T. Azuma. A survey of augmented reality. *Presence: Teleoper. Virtual Environ.*, 6(4):355–385, Aug. 1997.

[34] I. Baggili, J. Oduro, K. Anthony, F. Breitinger, and G. McGee. Watch what you wear: Preliminary forensic analysis of smart watches. In *2015 10th International Conference on Availability, Reliability and Security*, pages 303–311, 2015.

[35] D. Balzarotti, M. Cova, and G. Vigna. Clearshot: Eavesdropping on keyboard input from video. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 170–183, 2008.

[36] B. D. Barisani A. Sniffing keystrokes with lasers and voltmeters. In *In Proceedings of Black Hat USA*, 2009.

[37] V. Becker, L. Fessler, and G. Sörös. Gestear: Combining audio and motion sensing for gesture recognition on smartwatches. In *Proceedings of the 23rd International Symposium on Wearable Computers*, ISWC '19, page 10–19, New York, NY, USA, 2019. Association for Computing Machinery.

[38] Y. Berger, A. Wool, and A. Yeredor. Dictionary attacks using keyboard acoustic emanations. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, page 245–254, New York, NY, USA, 2006. Association for Computing Machinery.

[39] L. Breiman. Arcing classifiers. *ANNALS OF STATISTICS*, 26:801–823, 1998.

[40] L. Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, Oct. 2001.

[41] J. Brooke. *"SUS-A quick and dirty usability scale." Usability evaluation in industry*. CRC Press, June 1996. ISBN: 9780748404605.

[42] C. Cao, Z. Li, P. Zhou, and M. Li. Amateur: Augmented reality based vehicle navigation system. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 2(4), Dec. 2018.

[43] J. Carmigniani, B. Furht, M. Anisetti, P. Ceravolo, E. Damiani, and M. Ivkovic. Augmented reality technologies, systems and applications. *Multimedia Tools and Applications*, 51:341–377, 2010.

[44] B. Chen, V. Yenamandra, and K. Srinivasan. Tracking keystrokes using wireless signals. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '15, page 31–44, New York, NY, USA, 2015. Association for Computing Machinery.

[45] A. Compagno, M. Conti, D. Lain, and G. Tsudik. Don't skype & type! acoustic eavesdropping in voice-over-ip. In *Proceedings ACM on Asia Conference on Computer and Communications Security*, page 703–715, New York, NY, USA, 2017. ACM.

[46] S. Dash. Here's how indian startup blinkin helped set up acs in hospitals in wuhan during the coronavirus crisis, Mar 2020.

[47] A. Doshi, S. Y. Cheng, and M. M. Trivedi. A novel active heads-up display for driver assistance. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(1):85–93, 2009.

[48] M. Duckham and L. Kulik. A formal model of obfuscation and negotiation for location privacy. In *Proceedings of the Third International Conference on Pervasive Computing*, PERVASIVE'05, pages 152–170, Berlin, Heidelberg, 2005. Springer-Verlag.

[49] P. W. Eklund. A performance survey of public domain supervised machine learning algorithms. Technical report, 2002.

[50] S. C. for Digital Health and R. Health, 2019.

[51] G. Friedland and R. Sommer. Cybercasing the joint: On the privacy implications of geotagging. pages 1–8, 08 2010.

[52] J. Friedman. Tempest: A signal problem. *NSA Cryptologic Spectrum*, 35:76, 1972.

[53] Y. Gan, T. Wang, A. Javaheri, E. Momeni-Ortner, M. Dehghani, M. Hosseinzadeh, and R. Rawassizadeh. 11 years with wearables: Quantitative analysis of social media, academia, news agencies, and lead user community from 2009-2020 on wearable technologies. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 5(1), Mar. 2021.

[54] G. Ghinita, M. L. Damiani, C. Silvestri, and E. Bertino. Preventing velocity-based linkage attacks in location-aware applications. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '09, pages 246–255, New York, NY, USA, 2009. ACM.

[55] A. Gkoulalas-Divanis, P. Kalnis, and V. S. Verykios. Providing k-anonymity in location based services. *SIGKDD Explor. Newsl.*, 12(1):3–10, Nov. 2010.

[56] J. Gong, Z. Xu, Q. Guo, T. Seyed, X. A. Chen, X. Bi, and X.-D. Yang. *WrisText: One-Handed Text Entry on Smartwatch Using Wrist Gestures*, page 1–14. Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, New York, NY, USA, 2018.

[57] Y. Gu, C. Yu, Z. Li, Z. Li, X. Wei, and Y. Shi. Qwertyring: Text entry on physical surfaces using a ring. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 4(4), Dec. 2020.

[58] M. Hagiwara. *Real-World Natural Language Processing: Practical applications with deep learning*. Manning Publications, 2021.

[59] T. Halevi and N. Saxena. Keyboard acoustic side channel attacks: exploring realistic and security-sensitive scenarios. *International Journal of Information Security*, 14:1–14, 09 2014.

[60] W. U. Hassan, S. Hussain, and A. Bates. Analysis of privacy protections in fitness tracking social networks -or- you can run, but can you hide? In *27th USENIX Security Symposium (USENIX Security 18)*, pages 497–512, Baltimore, MD, 2018. USENIX Association.

[61] C.-W. Hsu, C.-C. Chang, and C. Lin. A practical guide to support vector classication. 2008.

[62] Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability. Standard, International Organization for Standardization, Geneva, CH, 1998.

[63] D. Kariuki. Hypergrid business, Feb 2020.

[64] K. Kim, M. Billinghurst, G. Bruder, H. B. Duh, and G. F. Welch. Revisiting trends in augmented reality research: A review of the 2nd decade of ismar (2008–2017). *IEEE Transactions on Visualization and Computer Graphics*, 24(11):2947–2962, Nov 2018.

[65] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[66] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[67] B. Klimt and Y. Yang. The enron corpus: A new dataset for email classification research. In *Proceedings of the 15th European Conference on Machine Learning*, ECML'04, page 217–226, Berlin, Heidelberg, 2004. Springer-Verlag.

[68] B. Klimt and Y. Yang. The enron corpus: A new dataset for email classification research. In *Proceedings of the 15th European Conference on Machine Learning*, ECML'04, page 217–226, Berlin, Heidelberg, 2004. Springer-Verlag.

[69] C. Kreider. The discoverability of password entry using virtual keyboards in an augmented reality wearable: an initial proof of concept. In *SAIS*, 2018.

[70] J. Krumm. Inference attacks on location tracks. In A. LaMarca, M. Langheinrich, and K. N. Truong, editors, *Pervasive Computing*, pages 127–143, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[71] G. Laput and C. Harrison. Sensing fine-grained hand activity with smartwatches. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, page 1–13, New York, NY, USA, 2019. Association for Computing Machinery.

[72] K. Lebeck, K. Ruth, T. Kohno, and F. Roesner. Towards security and privacy for multi-user augmented reality: Foundations with end users. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 392–408, May 2018.

[73] H.-M. C. Leung, C.-W. Fu, and P.-A. Heng. Twistin: Tangible authentication of smart devices via motion co-analysis with a smartwatch. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 2(2), July 2018.

[74] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10:707, February 1966.

[75] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710, February 1966. Doklady Akademii Nauk SSSR, V163 No4 845-848 1965.

[76] H. Li, A. Gupta, J. Zhang, and N. Flor. Who will use augmented reality? an integrated approach based on text analytics and field survey. *European Journal of Operational Research*, 10 2018.

[77] S. Li, A. Ashok, Y. Zhang, C. Xu, J. Lindqvist, and M. Gruteser. 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)*, pages 1–9, 2016.

[78] T.-S. Lim, W.-Y. Loh, and Y.-S. Shih. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, 40(3):203–228, Sep 2000.

[79] J.-W. Lin, P.-H. Han, J.-Y. Lee, Y.-S. Chen, T.-W. Chang, K.-W. Chen, and Y.-P. Hung. Visualizing the keyboard in virtual reality for enhancing immersive experience. In *ACM SIGGRAPH 2017 Posters*, SIGGRAPH '17, New York, NY, USA, 2017. Association for Computing Machinery.

[80] Z. Ling, Z. Li, C. Chen, J. Luo, W. Yu, and X. Fu. I know what you enter on gear vr. In *2019 IEEE Conference on Communications and Network Security (CNS)*, pages 241–249, June 2019.

[81] X. Liu, Z. Zhou, W. Diao, Z. Li, and K. Zhang. When good becomes evil: Keystroke inference with smartwatch. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, page 1273–1285, New York, NY, USA, 2015. Association for Computing Machinery.

[82] S. Loughran. Advanced deanonymization through strava. `https://bit.ly/3cNURHN`, 2019. Accessed: 2021-04-20.

[83] C. X. Lu, B. Du, P. Zhao, H. Wen, Y. Shen, A. Markham, and N. Trigoni. Deepauth: In-situ authentication for smartwatches via deeply learned behavioural biometrics. In *Proceedings of the 2018 ACM International Symposium on Wearable Computers*, ISWC '18, page 204–207, New York, NY, USA, 2018. Proceedings of the 2018 ACM International Symposium on Wearable Computers.

[84] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. L-diversity: privacy beyond k-anonymity. In *22nd International Conference on Data Engineering (ICDE'06)*, pages 24–24, April 2006.

[85] I. S. Mackenzie and S. X. Zhang. An empirical investigation of the novice experience with soft keyboards. *Behaviour & Information Technology*, 20:411–418, 2001.

[86] I. Maglogiannis, K. Karpouzis, M. Wallace, and J. Soldatos, editors. *Emerging Artificial Intelligence Applications in Computer Engineering - Real Word AI Systems with Applica-*

tions in eHealth, HCI, Information Retrieval and Pervasive Technologies, volume 160 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2007.

[87] A. Maiti, O. Armbruster, M. Jadliwala, and J. He. Smartwatch-based keystroke inference attacks and context-aware protection mechanisms. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '16, page 795–806, New York, NY, USA, 2016. Association for Computing Machinery.

[88] A. Maiti, M. Jadliwala, and C. Weber. Preventing shoulder surfing using randomized augmented reality keyboards. In *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 630–635, March 2017.

[89] D. Maloney. Augmented reality aids in the fight against covid-19, Mar 2020.

[90] P. Marquardt, A. Verma, H. Carter, and P. Traynor. (sp)iphone: Decoding vibrations from nearby keyboards using mobile phone accelerometers. In *18th ACM Conference on Computer and Communications Security*, CCS '11, page 551–562, New York, NY, USA, 2011. ACM.

[91] R. McPherson, S. Jana, and V. Shmatikov. No escape from reality: Security and privacy of augmented reality browsers. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, page 743–753, Republic and Canton of Geneva, CHE, 2015. International World Wide Web Conferences Steering Committee.

[92] Z. Medenica, A. L. Kun, T. Paek, and O. Palinko. Augmented reality vs. street views: A driving simulator study comparing two emerging navigation aids. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '11, page 265–274, New York, NY, USA, 2011. Association for Computing Machinery.

[93] U. Meteriz, N. F. Yildiran, J. Kim, and D. Mohaisen. Understanding the potential risks of sharing elevation information on fitness applications. In *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, pages 464–473, 2020.

[94] N. Metropolis and S.Ulam. The monte carlo method. *Journal of the American Statistical Association*, 44(247):335–341, 1949. PMID: 18139350.

[95] M. F. Mokbel. Privacy in location-based services: State-of-the-art and research directions. In *2007 International Conference on Mobile Data Management*, pages 228–228, May 2007.

[96] J. V. Monaco. Sok: Keylogging side channels. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 211–228, May 2018.

[97] P. Norvig, 2007.

[98] N. Odom, J. Lindmar, J. Hirt, and J. Brunty. Forensic inspection of sensitive user data and artifacts from smartwatch wearable devices. *Journal of Forensic Sciences*, 64, 06 2019.

[99] J. P Higgins. Smartphone applications for patients' health & fitness. *The American journal of medicine*, 129, 06 2015.

[100] S. Padilla, I. Mujika, G. Cuesta, and J. Goiriena. Level ground and uphill cycling ability in professional road cycling. *Medicine and science in sports and exercise*, 31 6:878–85, 1999.

[101] I. Polakis, G. Argyros, T. Petsios, S. Sivakorn, and A. D. Keromytis. Where's wally?: Precise user discovery attacks in location proximity services. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 817–828, New York, NY, USA, 2015. ACM.

[102] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019.

[103] P. A. Rauschnabel, R. Felix, and C. Hinsch. Augmented reality marketing: How mobile AR-apps can improve brands through inspiration. *Journal of Retailing and Consumer Services*, 49(C):43–53, 2019.

[104] F. Roesner, T. Kohno, and D. Molnar. Security and privacy for augmented reality systems. *Commun. ACM*, 57(4):88–96, Apr. 2014.

[105] L. Rossi, M. J. Williams, C. Stich, and M. Musolesi. Privacy and the city: User identification and location semantics in location-based social networks. In *Proceedings of the Ninth International Conference on Web and Social Media, ICWSM 2015, University of Oxford, Oxford, UK, May 26-29, 2015*, pages 387–396, 2015.

[106] K. Ruth, T. Kohno, and F. Roesner. Secure multi-user content sharing for augmented reality applications. In *Proceedings of the 28th USENIX Conference on Security Symposium*, page 141–158, 2019.

[107] M. Sabra, A. Maiti, and M. Jadliwala. Zoom on the keystrokes: Exploiting video calls for keystroke inference attacks, 2020.

[108] P. Shirley and S. Marschner. *Fundamentals of Computer Graphics*. A. K. Peters, Ltd., USA, 3rd edition, 2009.

[109] R. Shokri, G. Theodorakopoulos, J. Le Boudec, and J. Hubaux. Quantifying location privacy. In *2011 IEEE Symposium on Security and Privacy*, pages 247–262, May 2011.

[110] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[111] J. Sun, X. Jin, Y. Chen, J. Zhang, Y. Zhang, and R. Zhang. Visible: Video-assisted keystroke inference from tablet backside motion. In *NDSS*, 2016.

[112] K. Sun, W. Wang, A. X. Liu, and H. Dai. Depth aware finger tapping on virtual displays. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '18, page 283–295, New York, NY, USA, 2018. Association for Computing Machinery.

[113] L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):571–588, Oct. 2002.

[114] T. Sztyler and H. Stuckenschmidt. Online personalization of cross-subjects based activity recognition models on wearable devices. In *2017 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 180–189, 2017.

[115] D. Takahashi. Pokémon go is the fastest mobile game to hit $600 million in revenues, Oct 2016.

[116] M. Tonnis, C. Lange, and G. Klinker. Visual longitudinal and lateral driving assistance in the head-up display of cars. In *IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 91–94, 2007.

[117] Z. Tu, F. Xu, Y. Li, P. Zhang, and D. Jin. A new privacy breach: User trajectory recovery from aggregated mobility data. *IEEE/ACM Transactions on Networking*, 26(3):1446–1459, June 2018.

[118] T. Tullis and W. Albert. *Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics*. Morgan Kaufmann Series in Interactive Technologies. Morgan Kaufmann, Amsterdam, 2 edition, 2013.

[119] A. Visuri, Z. Sarsenbayeva, N. van Berkel, J. Goncalves, R. Rawassizadeh, V. Kostakos, and D. Ferreira. *Quantifying Sources and Types of Smartwatch Usage Sessions*, page 3569–3581. Association for Computing Machinery, New York, NY, USA, 2017.

[120] T. H. Vu, A. Misra, Q. Roy, K. C. T. Wei, and Y. Lee. Smartwatch-based early gesture detection 8 trajectory tracking for interactive gesture-driven applications. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 2(1), Mar. 2018.

[121] M. Vuagnoux and S. Pasini. Compromising electromagnetic emanations of wired and wireless keyboards. *USENIX Security Symposium*, 01 2009.

[122] C. Wang, X. Guo, Y. Wang, Y. Chen, and B. Liu. Friend or foe?: Your wearable devices reveal your personal pin. pages 189–200, 05 2016.

[123] H. Wang, T. T.-T. Lai, and R. Roy Choudhury. Mole: Motion leaks through smartwatch sensors. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, MobiCom '15, page 155–166, New York, NY, USA, 2015. Association for Computing Machinery.

[124] M. Wernke, P. Skvortsov, F. Dürr, and K. Rothermel. A classification of location privacy attacks and approaches. *Personal and Ubiquitous Computing*, 18(1):163–175, Jan 2014.

[125] M. Wilson. Snap is the world's most innovative company of 2020, Mar 2020.

[126] K. Wolf, K. Marky, and M. Funk. We should start thinking about privacy implications of sonic input in everyday augmented reality! 2018.

[127] Y. Yin, Q. Li, L. Xie, S. Yi, E. Novak, and S. Lu. Camk: Camera-based keystroke detection and localization for small mobile devices. *IEEE Transactions on Mobile Computing*, 17(10):2236–2251, 2018.

[128] C. Yu, Y. Gu, Z. Yang, X. Yi, H. Luo, and Y. Shi. Tap, dwell or gesture? exploring head-based text entry techniques for hmds. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, page 4479–4488, New York, NY, USA, 2017. Association for Computing Machinery.

[129] C. Yu, K. Sun, M. Zhong, X. Li, P. Zhao, and Y. Shi. One-dimensional handwriting: Inputting letters and words on smart glasses. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, page 71–82, New York, NY, USA, 2016. Association for Computing Machinery.

[130] X. Yu, Z. Zhou, M. Xu, X. You, and X. Li. Thumbup: Identification and authentication by smartwatch using simple hand gestures. In *2020 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–10, 2020.

[131] H. Zhang, Y. Yin, L. Xie, and S. Lu. Airtyping: A mid-air typing scheme based on leap motion. In *Adjunct Proceedings of the 2020 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2020 ACM International Symposium on Wearable Computers*, UbiComp-ISWC '20, page 168–171, New York, NY, USA, 2020. Association for Computing Machinery.

[132] D. Zheng, T. Hu, Q. You, H. A. Kautz, and J. Luo. Towards lifestyle understanding: Predicting home and vacation locations from user's online photo collections. In *Proceedings of the Ninth International Conference on Web and Social Media, ICWSM 2015, University of Oxford, Oxford, UK, May 26-29, 2015*, pages 553–561, 2015.

[133] L. Zhuang, F. Zhou, and J. D. Tygar. Keyboard acoustic emanations revisited. *ACM Trans. Inf. Syst. Secur.*, 13(1), Nov. 2009.