SECURITY AND PRIVACY OF CYBER AND PHYSICAL USER INTERACTIONS IN THE AGE OF WEARABLE COMPUTING

A Dissertation by

Anindya Maiti

Master of Science, Wichita State University, 2014

Bachelor of Technology, Vellore Institute of Technology, 2012

Submitted to the Department of Electrical Engineering and Computer Science
and the faculty of the Graduate School of
Wichita State University
in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

May 2018

SECURITY AND PRIVACY OF CYBER AND PHYSICAL USER INTERACTIONS IN THE AGE OF WEARABLE COMPUTING

The following faculty members have examined the final copy of this dissertation for form and content, and recommend that it be accepted in partial fulfillment of the requirement for the degree of Doctor of Phylosophy with a major in Electrical Engineering and Computer Science.

Murtuza Jadliwala, Committee Chair

Huzefa Kagdi, Committee Member

Sergio Salinas, Committee Member

Vinod Namboodiri, Committee Member

Jibo He, Committee Member

Accepted for the College of Engineering

Royce Bowden, Dean

Accepted for the Graduate School

Dennis Livesay, Dean

DEDICATION

To my brother, Ananda. And my parents, Bhaswati and Chinmay.

ACKNOWLEDGMENTS

I offer my sincere gratitude to my doctoral advisor, Dr. Murtuza Jadliwala, who has supported me throughout my doctoral studies with his patience and knowledge whilst allowing me to work in my own way.

I would like to thank the rest of my dissertation committee members: Dr. Huzefa Kagdi, Dr. Sergio Salinas, Dr. Vinod Namboodiri, and Dr. Jibo He. Thank you for your time and support with my research and graduate studies.

Graduate school experience can never be complete without labmates. I was fortunate to work alongside few of the most tremendous students in town. Thanks to Oscar for making me work more than I was willing to, especially on Sundays. It was also amazing to work with Kirsten, Chase, Ryan, Mohd, Arash, Nisha, and Raveen on various projects.

Research reported in this dissertation was partially supported by the Division of Computer and Network Systems (CNS) of the National Science Foundation (NSF) under award number 1828071 (originally 1523960).

ABSTRACT

Wearable devices are a new form of technology that is quickly gaining popularity among mobile users. These "smart" wearable devices are equipped with a variety of high-precision sensors that enable the collection of rich contextual information related to the wearer and her/his surroundings, which in turn enables a variety of novel applications. The presence of a diverse set of *zero-permission* sensors on wearable devices, however, also expose an additional attack surface which, if not adequately protected, could be potentially exploited to leak private user information. The first part of this dissertation aims to develop a comprehensive technical understanding of the privacy risks associated with inference of private user interactions with other *cyber* and *physical* systems, primarily using wrist-wearables. A detailed evaluation of novel attack frameworks validate the feasibility of inference attacks on both cyber interfaces, such as mobile keypads and computer keyboards, and on physical systems, such as combination padlocks and safes.

In order to thwart these new privacy threats, effective and usable techniques for detection and mitigation of wearable device misuse is critical and urgently needed. Consequently, the second part of this dissertation aims to protect user interactions by proposing new protection mechanisms, which take two different strategies. The proposed *design-time* protection mechanism tries to prevent inference attacks by altering the interaction interfaces, whereas the proposed *run-time* protection mechanism utilizes contextual information to dynamically regulate zero-permission sensor data when users are detected to be vulnerable to known inference attacks.

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

Chapter Page

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

Chapter Page

TABLE OF CONTENTS (continued)

Chapter Page

LIST OF FIGURES

LIST OF FIGURES (continued)

Figure Page

LIST OF FIGURES (continued)

Figure Page

LIST OF FIGURES (continued)

Figure Page

LIST OF FIGURES (continued)

Figure Page

LIST OF FIGURES (continued)

Figure Page

LIST OF TABLES

LIST OF TABLES (continued)

Table Page

LIST OF ABBREVIATIONS

LED Light Emitting Diode

GPS Global Positioning System

SH-NHHT Same Hand and Non-Holding Hand Typing

SH-HHT Same Hand and Holding Hand Typing

DH-NHHT Different Hand and Non-Holding Hand Typing

SLR Simple Linear Regression

RF Random Forest

k-NN k-Nearest Neighbors

SVM Support Vector Machine

BDT Bagged Decision Trees

RR Row Randomization

CR Column Randomization

IKR Individual Key Randomization

KSR Key Size Randomization

KLR Keypad Location Randomization

SUS System Usability Scale

NASA-TLX NASA Task Load Index

RS Row Shifting

CS Column Shifting

AR Augmented Reality

rTAD Real-Time Typing Activity Detection

MSAC Motion Sensor Access-Controller

CHAPTER 1
INTRODUCTION

Wearables such as smartwatches and fitness bands are gaining tremendous popularity among mobile users, and will continue to be a prevalent mobile technology in the future [50]. These "smart" wearable devices are equipped with a variety of high-precision sensors that can capture extremely rich and fine-grained contextual and physiological information related to the mobile user and her/his surroundings. This fine-grained data captured using on-board sensors can be used to augment knowledge about the user (wearer) and facilitate learning about her/his activities, physiological state and environment, which in turn, can enable several novel non-traditional applications on, or by means of, these devices. The presence of a diverse set of *zero-permission* sensors, however, also expose an additional attack surface which, if not adequately protected, could be potentially exploited to leak private user information. Weak or absent access control and security policies vis-á-vis some of these sensors have further compounded this problem. Similar privacy threats that abuse access privileges to smartphone sensors have already received significant attention in the research literature [20, 102, 40, 125, 86, 98, 77, 9, 81]. Due to the upcoming and evolving nature of wearable technology, compared to smartphone technology which is significantly mature, privacy issues in wearables have not received similar scrutiny. The distinctively unique design and usage of wearables also puts them at a significantly higher risk. Contrary to smartphones, wearables are always carried by users on their body and capture a continuous stream of contextual or activity-specific data,

access to which if not appropriately regulated, can be exploited to infer sensitive user information. Thus, as wearable technology matures, it will be critical to thoroughly investigate the privacy threats that become feasible due to the availability of rich and diverse sensory data from these devices. As a matter of fact, privacy issues related to wearables is predicted to be a major challenge in the coming years [58].

The first part of this dissertation aims to develop a comprehensive technical understanding of the privacy risks associated with inference of private user interactions with other *cyber* and *physical* systems, primarily using wrist-wearables. A detailed evaluation of novel attack frameworks validate the feasibility of inference attacks on both cyber interfaces, such as mobile keypads (Chapter 2) and computer keyboards (Chapter 3), and on physical systems, such as combination padlocks and safes (Chapter 4).

In order to thwart these new privacy threats, effective and usable techniques for detection and mitigation of wearable device misuse is critical and urgently needed. Consequently, the second part of this dissertation aims to protect user interactions by proposing new protection measures, which take two different strategies. The proposed *design-time* protection measures tries to prevent inference attacks by altering the interaction interfaces (Chapter 5), and *run-time* protection measures use contextual information to dynamically regulate zero-permission sensor data when users are detected to be vulnerable to known inference attacks (Chapter 6).

CHAPTER 2
ATTACKS ON CYBER INTERACTIONS: MOBILE KEYPADS

2.1 Introduction

In this chapter, we evaluate the feasibility of side-channel security vulnerabilities in smart wearables by investigating motion-based keystroke inference attacks using smartwatches. More specifically, we evaluate the feasibility and effectiveness of keystroke inference attacks on smartphone numeric touchpads by using smartwatch motion sensors as a side-channel. Numeric touchpads are typically targeted by adversaries for obtaining sensitive information such as security pins and credit card numbers. We propose multiple attacks suitable for three popular typing scenarios. In typing scenarios where key press events can be identified based on surge in motion sensor activity, we use *supervised learning* to infer the key presses. This attack comprises of first training appropriate classification models to learn the uniqueness in wrist motion caused during individual keystrokes depending on known relative location of the key on the screen, and then using the trained classifiers to infer unlabeled (or test) keystrokes. During preliminary experiments, we observed that keystroke induced motion data captured by smartwatch and smartphone sensors differ significantly. Consequently, we thoroughly assess how significantly smartwatch motion sensors elevate the threat of keystroke inference, compared to similar attacks using only smartphone motion data [20, 86, 118]. We also evaluate the case where the adversary may have gained access to motion sensors on both the smartwatch and smartphone, to see how our attack will perform when motion data from both devices

are combined. For the typing scenario where key press events cannot be identified based on the uniqueness of motion sensor activity surge (corresponding to a key press), we present a novel scheme to infer a sequence of key presses based on the *transitional movement* between individual key presses. We evaluate the proposed attacks in both controlled and realistic typing scenarios. We also briefly discuss possible protection measures against such inference attacks that employ motion sensors as side-channels.

The remainder of this chapter is structured as follows: First we discuss similar side-channel attacks in the literature in Section 2.2. Then we give an overview of the threat and adversary model in Section 2.3. In Section 2.4 we describe the classification-based keystroke inference framework, followed by its evaluation in Section 2.5. In Section 2.6 we describe the relative transitions-based keystroke inference framework, followed by its evaluation in Section 2.7. Finally, we discuss implications, limitations and potential future research directions.

2.2 Related Work

Inference of private information from various forms of side channels has been an active area of research in the community. Electromagnetic signals emanating from devices have been used to infer private data stored on Smart Cards [89], data transmitted on RS-232 cables [99] and content being played on CRT and LED monitors [112, 61]. Recently, Hayashi et al. [42] showed that it is also possible to remotely reconstruct and eavesdrop on flat panel displays on tablets via measurement of electromagnetic emanations. Similarly, optical emanations from monitors [60] or from

eyes [10] have also be used to infer information such as content being displayed or watched. Acoustic or sound signals emanating from devices such as printers have also been used to infer the content being printed on certain models of dot-matrix printers [11].

Availability of several high-precision sensors on modern mobile cyber-physical systems such as smartphones have given rise to additional side-channels [110], thus increasing the risk of private information leakage through such side-channels. Past research efforts have shown how malicious applications can misuse their access rights to these sensors in order to execute various imperceptible side-channel attacks by stealthily capturing information from the physical environment. For example, smartphone cameras can be accessed in an unauthorized fashion to infer sensitive information from user keystrokes [33, 98]. Unauthorized microphone recordings of ambient sound [94] provide a rich source of information that can be used to infer sensitive information about a person's daily life. Activities and locations can be inferred based on characteristic ambient sound patterns, e.g. walking on the streets, or eating in a restaurant [91]. Unauthorized access to GPS sensors can pose obvious risks related to loss of location privacy, such as revealing home/work locations, stalking and location-targeted advertisements [51]. Advanced learning-based techniques were also proposed for predicting users' future movements from previous tracking records of their location activities [84, 108, 36, 29]. Security and privacy risks associated with front-end sensors, such as, microphones, cameras and GPS, have been comprehensively studied because of the hazards apparent to users.

However, security risks due to sensors obscured from users (e.g., accelerometer, gyroscope, and magnetometer) have largely been overlooked until recently. After modern mobile operating systems introduced user-managed access control on front-end sensors, adversaries shifted attention to sensors which cannot be disengaged by users. It has been shown that malicious applications can track users' movements [40, 43] and activities [78, 124, 63] by using only smartphone accelerometer readings. It has also been shown that, with the help of standard signal processing and machine learning techniques, it is possible to recognize speakers and parse speech by using gyroscopes on modern mobile devices to measure acoustic signals [79].

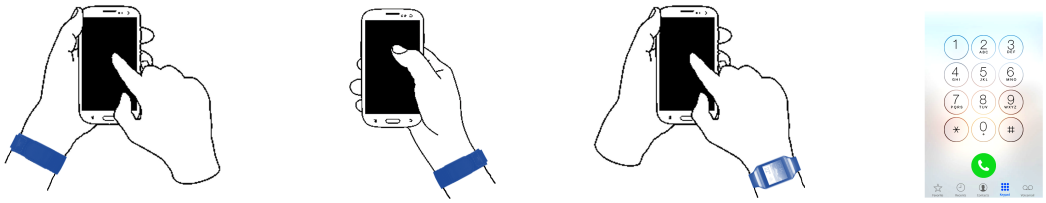
Keystroke inference attacks using side-channel information have received significant attention due to their potentially dangerous consequences. Electromagnetic emanations from external keyboards (both wired and wireless) have been used in the past to infer user keystrokes [113]. However, the requirement of extensive setup and expensive monitoring hardware prevents less sophisticated adversaries from carrying out such attacks. Keystroke inference attacks using audio or acoustic signals [8, 16, 127, 38, 126], on the other hand, have also received significant attention in the literature. Such attacks have proven to be very successful and can be carried out using modest off-the-shelf hardware (e.g., any microphone equipped device). Due to the ubiquitous nature of modern smartphones that are equipped with high-precision microphones, such attacks are much more practical than previously argued.

However, as touchscreen key press events emanate very weak acoustic signals, inference attacks using them is very difficult. Additionally, requirement of undisturbed eavesdropping is another major obstacle in using electromagnetic and acoustic em-

anations for such attacks. As a workaround to the above limitations, smartphone motion sensors have been used to recover keystroke events on the device. For instance, *TouchLogger* [20] and *TapPrints* [81] utilize change in orientation angles of the smartphone, as captured by its accelerometer, to extract appropriate features for keystroke inference. Similarly, *ACCessory* [86] also attempts to infer keystrokes using the smartphone accelerometer data by employing multiple supervised learning techniques. Alternatively, *TapLogger* [118] automates the training and logging phases and attempts to work stealthily on the smartphone. Smartphone motion sensors have also been used to detect keystroke events on other external devices/keyboards in proximity [77]. Despite these research efforts, side-channel privacy threats (especially, keystroke inference attacks) posed by wearable devices such as smartwatches have received far less attention. Recently, Wang et al. [115], Liu et al. [70], and Wang et al. [114] have explored new attacks to infer user keystrokes or key presses on external physical keyboards/keypads by using smartwatch motion sensors. In contrast to these research efforts, our work in this chapter focuses on keystroke inference on hand-held mobile keypads by employing smartwatch motion sensors.

2.3 Attack Description

In this research effort, we focus on three of the most popular typing (or tapping) scenarios in mobile hand-helds or smartphones [111]. We consider a user typing on a smartphone's numeric touchscreen keypad while wearing a smartwatch on one of his/her hand. In the first case, smartwatch and smartphone are on the same hand and the user types with the hand not holding the smartphone (see Figure 2.1(a)),



(a) (b) (c) (d)

Figure 2.1: Smartwatch and smartphone on (a) Same Hand and Non-Holding Hand Typing (SH-NHHT), (b) Same Hand and Holding Hand Typing (SH-HHT), (c) Different Hand and Non-Holding Hand Typing (DH-NHHT), (d) numeric keypad used in our experiments.

also referred by us as *SH-NHHT scenario*. In the second case, smartwatch and smartphone are again on the same hand and the user types with a finger (generally, thumb) of the smartphone holding hand (see Figure 2.1(b)), also referred by us as *SH-HHT scenario*. In the above two scenarios, the action of tapping a key on the smartphone keypad results in a unique motion of the wrist (on the smartphone holding hand) for each keystroke, which can be captured by the motion sensors (e.g., accelerometer and gyroscope) of the smartwatch and used to identify the tapped keystroke. In the third case, smartwatch and smartphone are on different hands and the user types with the hand not holding the smartphone (see Figure 2.1(c)), also referred by us as *DH-NHHT scenario*. Unlike the previous two scenarios, each key tap does not produce a unique motion signature on the wrist of the typing hand (where the smartwatch is situated), and thus it cannot be used to infer the exact keystroke in a fashion similar to the previous two cases. However, assuming that the relative position of keys on the keypad as per the standard layout shown in Figure

2.1(d) is known and remains static, we can use the relative transitional movement between taps to infer a (sub)sequence of the tapped keys.

In addition to the above three, other typing scenarios are also possible, for example, typing with both hands and holding (the phone) and typing with non watch wearing hand. However, in order to limit the scope of our study and to clearly demonstrate the keystroke inference threat posed by smartwatches, we only consider the above three typing scenarios (i.e., SH-NHHT, SH-HHT and DH-NHHT) in this work, which also happen to be very widely adopted by smartphone users. We further justify our focus on these three typing scenarios by more precisely determining the percentage of users that employ these typing methods, and as a result, are impacted by the proposed inference attacks. Based on the data available from a study concerning users' smartphone holding and usage behaviors [111], 32.83% of all users hold and use the phone as shown in Figure 2.1(b), 28.44% of users hold and use the phone as shown in Figure 2.1(a), while only 2.11% of users hold and use the phone as shown in Figure 2.1(c). In this work, we also investigate two variations of scenarios in Figures 2.1(a) and 2.1(b), where the phone is held in the other hand as shown in Figures 2.13(a) and 2.13(b) respectively. Based on [111], 16.17% of all users hold and use the phone as shown in Figure 2.13(b), while only 7.56% of users hold and use the phone as shown in Figure 2.13(a). It should be noted that [111] does not provide any data on which hand (left or right) the smartwatch is traditionally worn, so the above only represents percentages of users based on the hand in which the phone is held and the hand used to type or tap. Now even if the phone holding scenario of Figure 2.1(c) is ignored (due to its low usage percentage), our framework

can potentially impact at least 44.61% of users who wear the smartwatch on the left hand, i.e., those who type as shown in Figures 2.1(a) and 2.13(b). Similarly, at least 40.39% of users wearing the smartwatch on the right hand are impacted by our keystroke inference framework, i.e., those who type as shown in Figures 2.1(b) and 2.13(a)). Moreover, we also show in Section 2.5.9 that the performance of our framework does not vary significantly between a particular typing scenario and its variation (say, between Figure 2.1(b) and 2.13(b)). This shows that a significant percentage of users have the potential of being impacted by the proposed keystroke inference framework.

Threat Model: We assume an adversary whose goal is to infer a target's keystrokes on a generic smartphone numeric keypad (as shown in Figure 2.1(d)), based on the wrist movements perceptible by the target's smartwatch motion sensors. The adversary may gain access to the target's smartwatch by installing a malicious application on it which records the activity of the on-board accelerometer and gyroscope sensors. This step can be achieved by exploiting known software vulnerabilities or by tricking the victim into installing malicious code, e.g., using a trojan software. Based on the fact that most common smartwatch operating systems (e.g., Google's Android Wear, Apple's watchOS, etc.) do not implement access control and/or user notification for motion sensor usage, the malicious application may have unrestricted and undetected access to the on-board accelerometer and gyroscope. As a result, the compromised smartwatch can act as an eavesdropping device which the targets' themselves may place on their wrist, and unsuspectingly have it on their wrist while typing on a smartphone. The malicious application may also

maintain a covert communication channel [28] with the adversary, and periodically upload the collected wrist motion data on some adversarial server by means of this channel. The use of a covert channel by the trojan is optional. However, if a covert channel is not used, there is a possibility of this information transfer being easily detected and the trojan being inactivated. We assume that the adversary also has sufficient off-site storage and computational resources to download the raw sensor data, extract significant keystroke events, and execute standard machine learning algorithms in order to classify the keystrokes. For comparison with attacks based on smartphone data, we assume similar adversarial capabilities and actions for the smartphone.

2.4 Classification-Based Attack Framework

The linear accelerometer motion sensor found on smartwatches measures the three dimensional acceleration experienced by the device, excluding the omnipresent force of gravity. During preliminary experimentation with SH-NHHT and SH-HHT scenarios, we observed that key press events can be accurately detected using the surge in linear acceleration during a key press. Based on the observation that taps on different locations of the smartphone screen produces characteristically unique motions on the wrist, our attack framework leverages on *supervised machine learning* to directly classify the detected key presses. The attack framework (Figure 2.2) consists of a *learning phase* followed by an *attack phase*. Both phases go through similar steps of *data collection* followed by *feature extraction*, with the learning phase culminating in training (the classifiers), while the attack phase in classification (using the trained

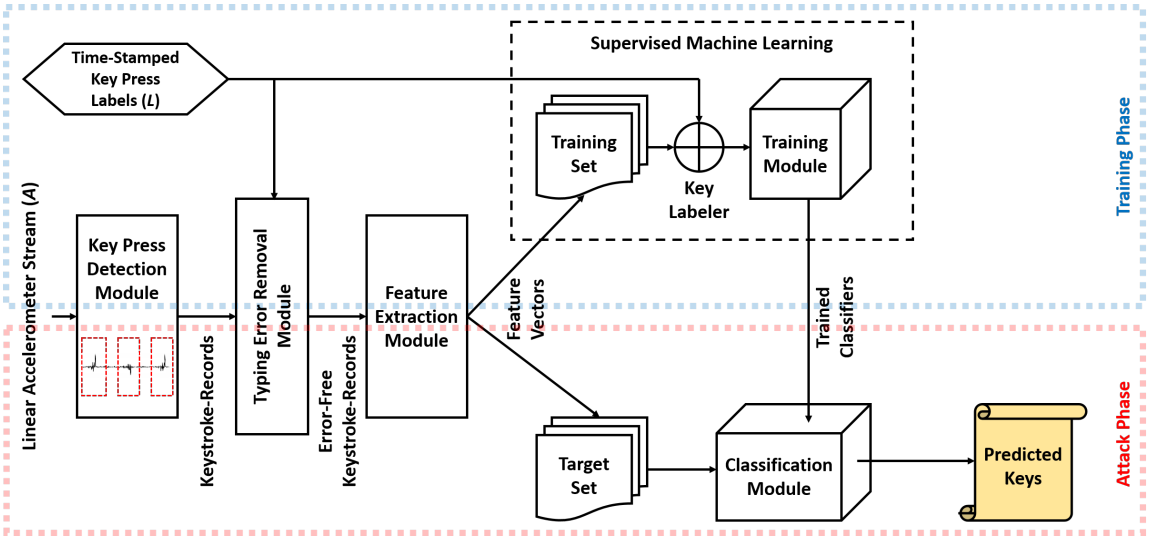


Figure 2.2: Overview of the classification-based attack framework for SH-NHHT and SH-HHT typing scenarios.

classifiers from the training phase). Next, we describe in detail each component of the proposed framework.

Data Collection and Pre-Processing: We developed an application for Android Wear that continuously samples linear accelerometer measurements on the smartwatch, and runs in the background during experiments. Details of the data collection experiments and technical specifications of the hardware used for the experiments are outlined later in Section 2.5.1. The smartwatch data collection application communicates the linear accelerometer measurements to the host smartphone (with which the watch is paired) using Andorid Wear's Wearable Data Layer API. On the smartphone, another Android application displays a keypad to the users to type sequences of numbers. In the background, the smartphone application chronologically logs all accelerometer measurements received from the smartwatch and any

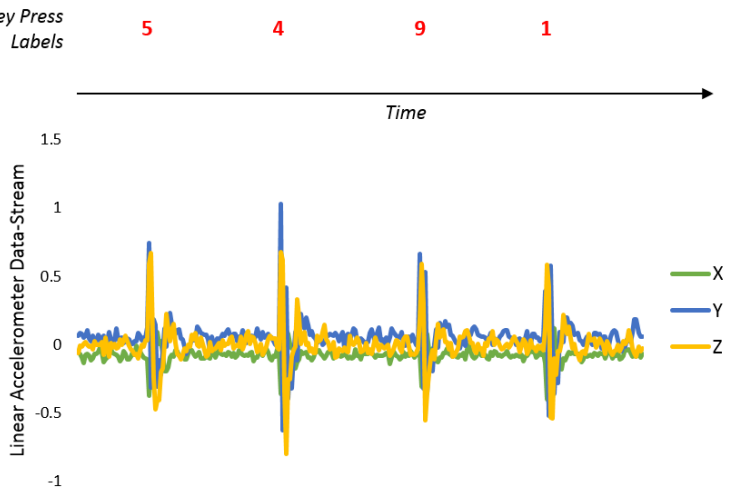


Figure 2.3: Time series of key press events in SH-NHHT, and their corresponding effect on linear accelerometer samples.

key press events registered on the displayed keypad. It logs two data-streams: (i) timestamped readings of the smartwatch's linear accelerometer (A); and (ii) timestamped key press labels (L). Both A and L are stored locally on the smartphone (which is also paired with the smartwatch) during the data collection process and retrieved later for offline evaluation. Note that in the attack phase of our experiments we use L only to verify the classification accuracy.

Due to the absence of labeled data, the attack phase requires an additional key press event detection mechanism. Figure 2.3 shows a portion of a raw linear accelerometer data-stream, spanning four key press events in the SH-NHHT scenario. As evident from the graph, each tap agitates the linear accelerometer sensor readings on the three axis, with more prominence along the Y-axis and Z-axis than X-axis. SH-HHT data also exhibits similar traits. We apply this observation to model an

algorithm for automating the process of key press event detection. Algorithm 1 sequentially examines the "energy" of each sample i in A as the sum of acceleration on the three axis (Equation 2.1). The energy value calculated in Algorithm 1 is then used in Algorithm 2 to determine keystroke events.

$$Energy[i] = \Big|\big|A[i][X]\big| + \big|A[i][Y]\big| + \big|A[i][Z]\big|\Big| \tag{2.1}$$

Algorithm 1 Key Press Detection Algorithm

```
function KeyPress_Detection(A^Target)
    KeyPresses = {∅}
    Threshold = Set_Threshold()
    for i = 1 to N (N samples in A) do
        if Energy[i] ≥ Threshold then
            ThisKeyPress = A[i − 3] to A[i + 15]
            Insert ThisKeyPress into KeyPresses
            i = i + 15
        end if
    end for
end function
```

Algorithm 2 establishes the threshold value as the average peak energy values observed in the time-stamped training set. In the attack phase, once the energy level surpasses the empirically learned threshold (from Algorithm 2), a key press event is recognized and a "keystroke-record" is saved. Each keystroke-record is intended to represent the wrist motion pattern of a key press event, and consists of few linear accelerometer readings immediately before and after the key press event is recognized. We empirically observed that the movement due to a key press subsides after approximately 350 $msecs$. Thus, a keystroke-record of eighteen samples (at 50 Hz sampling frequency) sufficiently captures all motion features related to a keystroke. Taking

into consideration some of the milder initial motion, we form each keystroke-record as follows: the sample to surpass the energy threshold is preceded by three samples and followed by fourteen samples, chronologically from A. After a key press event is recognized and the corresponding keystroke-record is saved, the key press detection algorithm resumes its search for next key press. As multiple samples during a key press may cross the energy threshold, ignoring the fourteen samples following the last keystroke-record ensures that the same key press is not recorded multiple times.

Algorithm 2 Determining Energy Threshold

function SET_THRESHOLD($A^{Training}$, $L^{Training}$)
  $Threshold = 0$
  $KeyPressTime = 0$
  for $j = 1$ to M (M key presses in L) **do**
    $KeyPressTime = L[j][time]$
    $ThisKeyEnergy =$
    $\{Energy[KeyPressTime - 1] +$
    $Energy[KeyPressTime] +$
    $Energy[KeyPressTime + 1] +$
    $Energy[KeyPressTime + 2]\}/4$
    $Threshold = Threshold + ThisKeyEnergy$
  end for
  $Threshold = Threshold/M$
return M
end function

Feature Extraction: Our proposed attack infers the numeric key that was pressed based on features of the underlying physical event of wrist motion caused during typing (or tapping) on a smartphone. The features of a keystroke-record must be able to capture as many attributes as possible about the underlying three-dimensional movement caused by a key press. A properly designed feature vector

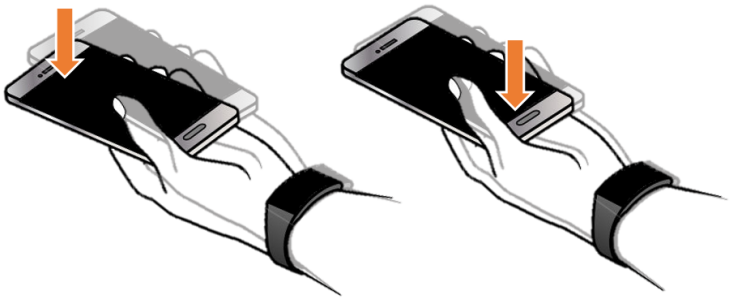


Figure 2.4: The intuition behind our classification-based attack is that taps on different locations of the smartphone screen produces characteristically unique motions on the wrist. Accordingly, taps on each number on the keypad should be identifiable based on the uniqueness in the resultant wrist motion.

should be similar with other feature vectors of the same key, simultaneously being distinguishable between feature vectors of other keys. We observed that, based on the location of a key on the screen, the degree of movement caused by a tap varies on each of the X, Y, and Z axis of the linear accelerometer (Figure 2.4). Interestingly, this movement remains fairly consistent for the same key.

In our preliminarily work [74], we used 54 basic time domain features of the accelerometer data to identify the uniqueness of each key (and the corresponding key press event), and found those features to be reasonably useful for keystroke inference. Later we expand that to a more comprehensive set of features, employing both time and frequency domain features, with a total of 155 different features in our feature vector for each key. We continue to use time domain features of individual axis such as minimum and maximum magnitudes, squared sum of magnitude data below 33 percent and above 67 percent of maximum magnitude (to measure

the duration of major and minor movements), position of maximum and minimum magnitude samples, mean, median, variance, standard deviation, skewness (measure of any asymmetry) and kurtosis (to measure any peakedness), raw accelerometer readings, and their first order numerical derivatives (to measure the rate of change of energy). We also use inter-axis time domain features to capture the correlation between movement on the three axis, such as minimum and maximum magnitudes across all three axis, Frobenius norm, Infinity norm, 1-norm, Euclidean norm, and axis with highest and lowest magnitude for each time sample. Along with the time domain features, we also capture frequency domain features by computing the Fast Fourier Transform (FFT) of individual axis readings of the keystroke-record. The frequency domain features are necessary to identify the different rebounding (or oscillatory) motion of the wrist. Note that in the learning phase, the feature vectors are also labeled, using the timestamped key press labels (L) recorded by the data collection application.

Training and Classification: We model the keystroke inference problem as a *multi-class classification* problem. Labeled feature vectors are used to train classifiers in the *learning phase*, whereas unlabeled feature vectors are mapped to the "closest" matching class by the already trained classifiers during the *attack or test phase*. To train our classifiers, we initially tested five different classification algorithms that are appropriate given the properties of our features: (i) *simple linear regression (SLR)*, (ii) *random forest (RF)*, (iii) *k-nearest neighbors (k-NN)*, (iv) *support vector machine (SVM)*, and *bagged decision trees (BDT)*. However, each of these classification techniques has its own advantages and shortcomings, leading us to adopt an *ensemble*

classification approach. Ensemble approaches have proven to be more accurate and robust than any single classification algorithm [24, 54]. We consider an extremely broad set of classification algorithms in our ensemble method, as a result of which, the errors made by constituting classifiers are highly uncorrelated. We include parametric algorithms (SLR, SVM), as well as non-parametric algorithms (k-NN, RF, BDT). Our ensemble method involves both linear (SLR, SVM) and non-linear (k-NN, RF, BDT) techniques. Moreover, RF and BDT are strong ensemble classifiers in themselves which makes our classification framework even more robust. Such a diverse set of classification algorithms increases the likelihood of improvements in classification accuracy over a single algorithm.

During the training phase, multi-class classifiers of each constituting classification algorithm are trained separately using the labeled training data. After all the classifiers have been trained using the labeled data, feature vectors of unlabeled keystroke-records are classified using these trained classifiers (in the attack phase) using an ensemble strategy. Finally, a *majority wins* ensemble strategy is used to determine the final classification result (Figure 2.5).

2.5 Evaluation of Classification-Based Attacks

In this section, we present the findings from our evaluation of the classification-based attack framework.

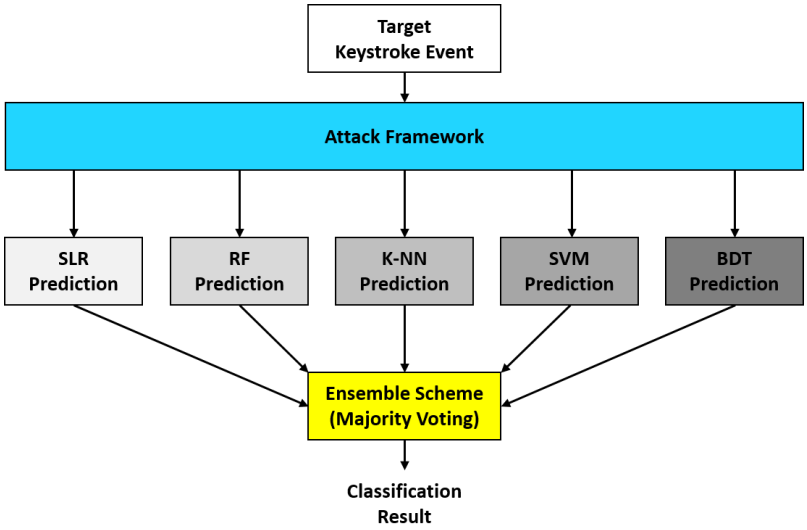


Figure 2.5: Ensemble classification scheme used in the attack phase is robust and generally more accurate than a single classification algorithm.

2.5.1 Experimental Setup

Our initial data collection experiments involve 12 participants, aged between 19-32 years. The identity of these participants are anonymized as P_1, P_2, ..., P_{12}. We employ a Samsung Gear Live smartwatch equipped with an InvenSense MP92M 9-axis Gyro + Accelerometer + Compass sensor. Smartwatch was worn on left hand for SH-NHHT (Figure 2.1(a)) and on right hand for SH-HHT (Figure 2.1(b)). Participants use the virtual numeric keypad of a Motorola XT1028 smartphone (Figure 2.1(d)) for typing. Linear accelerometer of the smartwatch was sampled at 50 Hz. We used the Weka 3.7.12 [39] libraries for both training and testing the classifiers. MATLAB R2014a was used to compute most of the time and frequency domain features.

We also evaluate the performance of our keystroke inference framework in several additional settings: (i) a more natural or uncontrolled typing scenario (Section 2.5.6), (ii) using a different smartwatch hardware (Section 2.5.7), (iii) employing an additional type of motion sensor, i.e., gyroscope (Section 2.5.4), and (iv) typing on a QWERTY or alphabetic keypad (Section 2.5.8). For these last four experimental settings, we collected additional data from different sets of participants, and in certain cases using a different smartwatch and/or smartphone hardware. The participant and data collection procedure details for these additional experimental settings appear in their respective sections.

2.5.2 Constructing and Testing the Classifiers

We construct our classifiers based on different training datasets of labeled keystroke-records generated by the participants. An audio stream of uniformly distributed random numbers between 0 to 9 guided the participants in typing. To prevent fatigue, participants were given optional breaks, during which they were allowed to set down the phone on the table and some participants even went out of the room. However, they returned to approximately the same holding position after the break. We comparatively evaluate the classification accuracy (the percentage of correct prediction divided by the total number of predictions) of our classifiers for the following three training/testing scenarios:

- ***One vs. One***: In this case, we measure the percentage of successful inferences on an individual participant, with classifiers trained from the training set of the same participant. Target set size is 100 (10 per key) and training set size is 200

(20 per key). *One vs. One* is not only a best case scenario, but also represents how the attack will perform if the adversary is able to collect target-specific training data.

- ***One vs. Rest***: In this case, we measure the percentage of successful inferences on an individual participant, with classifiers trained from the training set of the rest of the participants (not including the target participant). Target set size is 100 (10 per key) and training set size is 2200 (220 per key). *One vs. Rest* is a typical scenario where the adversary has a target, but is unable to obtain labeled training data from the target.
- ***All vs. All***: In this case, we measure the percentage of successful inferences on all participants, with classifiers trained from training set of all participants. Target set size is 1200 (120 per key) and training set size is 2400 (240 per key). *All vs. All* is helpful in understanding how our attack framework will perform if the adversary constructs a heterogeneous training data set to infer keystrokes from multiple non-specific targets.

Classification results for *One vs. One* are shown in Figure 2.6. *One vs. One* classification accuracy ranged fairly high between 94% and 77% for SH-NHHT, and between 93% and 75% for SH-HHT, with an average of 84.58% and 83.5%, respectively. However, classification accuracy drops noticeably in *One vs. Rest*. As shown in Figure 2.6, *One vs. Rest* classification accuracy ranged between 82% and 63% for SH-NHHT, and between 78% and 65% for SH-HHT, with an average of 70.08% and 71.16%, respectively. The achieved *All vs. All* classification accuracy was 88.16% and 85.83% for SH-NHHT and SH-HHT, respectively. Overall, these results validate

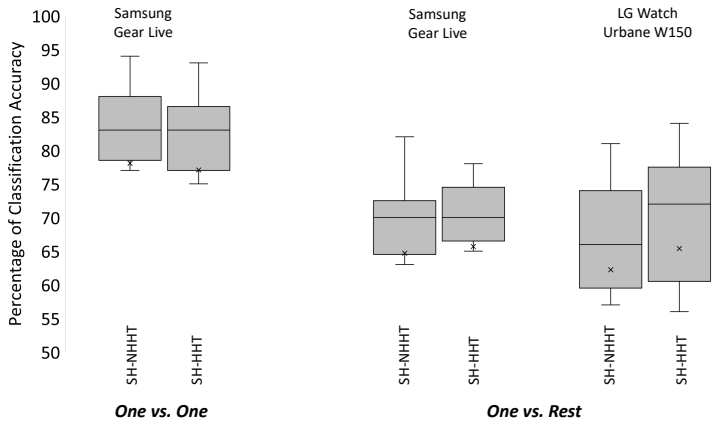


Figure 2.6: Classification accuracy for *One vs. One* and *One vs. Rest* using two different smartwatches (Samsung Gear Live and LG Watch Urbane W150).

Table 2.1: Mean computation time observed in each training/testing scenario. All measurements are in seconds.

	SLR	RF	K-NN	SVM	BDT	Total
One vs. One	191	234	98	95	65	683
One vs. Rest	166	365	398	184	167	1280
All vs. All	504	617	441	271	218	2051

our claim that smartwatch motion sensors are a feasible side-channel for inferring keystrokes on mobile touchpads.

We also recorded the mean computation time in each training/testing scenario (Table 2.1). All training and testing operations were executed on a laptop featuring a 2.7 GHz dual-core Intel i5 processor and 8 GB of working memory. Due to the use of ensemble classification technique, the total computation time is the sum of time taken by the five constituting classification algorithms. The average total computation time in *One vs. One* scenario was less then 12 minutes, about 21 minutes in *One vs. Rest*,

and about 34 minutes in *All vs. All* scenario. Moreover, the total computation time can be further reduced if the different classification algorithms are executed in parallel (with suitable hardware support). These results show that the above keystroke inference attacks can be carried out by an attacker using reasonable computation resources in a fairly short amount of time.

2.5.3 Reduced Sampling Frequency:

We also briefly investigate how our attack will perform at reduced sampling rate (25 Hz and 10 Hz), a more realistic scenario for low-cost wearables, equipped with less precise sensors. We repeat the experiments outlined in Sections 2.5.1 and 2.5.2 with smartwatch data sampled at a reduced frequency, and Figure 2.7 shows the drop in accuracy of our attacks for both the SH-NHHT and SH-HHT scenarios. For example, *One vs. One* classification accuracy in SH-NHHT dropped from 84.58% to 72% when sampling frequency was reduced to 25 Hz and to 23% when sampling frequency was reduced to 10 Hz. Similarly, the other scenarios also observed drop in classification accuracy with reduction in sampling frequency, but percentage of successful classification can be considered fairly substantial even at a sampling frequency of 25 Hz.

2.5.4 Comparison with Smartphone-Based Attacks

Previous research efforts on keystroke inference attacks by using smartphone sensor data [118, 81] (or data collected from the target's smartphone sensors) also used similar learning-based multi-class classification frameworks. This motivated us to apply our attack framework on smartphone data and compare the results with those

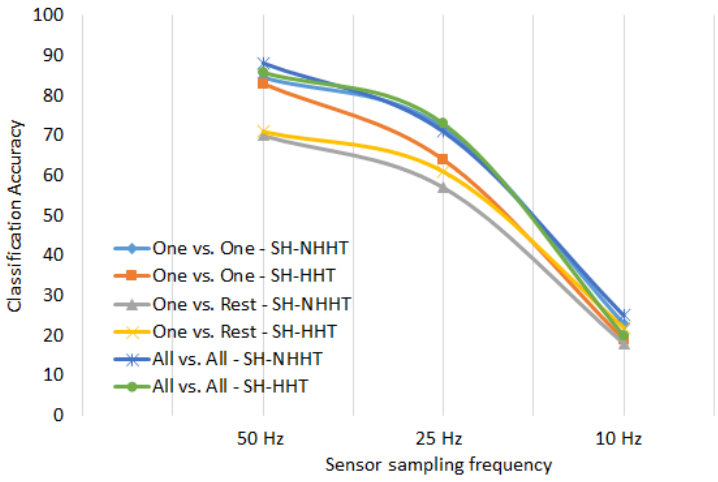


Figure 2.7: Classification accuracy dropped when sampling rate was reduced, results averaged over all 12 participants.

carried out using smartwatch data. This enables us to understand how much more or less vulnerable a motion sensor-based side-channel originating on a smartwatch makes us, as compared to known motion-based side-channels on the target users' smartphone. We conduct similar experiments (as in Sections 2.5.1 and 2.5.2) by using smartphone linear accelerometer data sampled at 50 Hz, rather than using the smartwatch data. Figures 2.8 and 2.9 shows the accuracy of our attack for SH-NHHT and SH-HHT scenarios. On comparing with previous results from Section 2.5.2, it can be observed that the keystroke inference attacks in SH-NHHT resulted in slightly better average classification accuracy when smartwatch motion data was used. Whereas in SH-HHT, classification accuracy results are mixed, and nearly equal, for both the smartwatch and smartphone data. In summary, these results demonstrate that the threat of motion-based keystroke inference may be increased

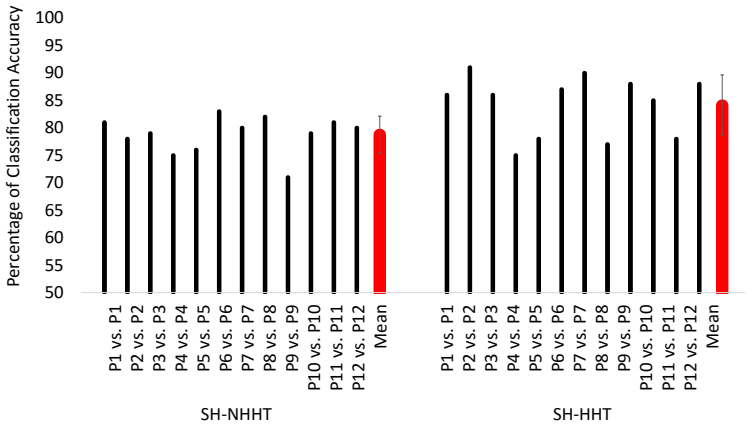


Figure 2.8: Classification accuracy for *One vs. One* using smartphone data.

in certain typing scenarios due to smartwatches. An interesting pattern of classification accuracy can be observed (see Fig. 2.10) for inference using only smartphone data in SH-HHT. We observe that the classification accuracy for certain keys (based on their location) are distinctly higher than others. Interestingly, this occurrence is not recognizable for the smartwatch dataset. This may be due to the fact that keys farther away from the thumb impels the user to bend the phone towards the thumb. As a result, significantly greater movement of the phone occurs, compared to keys that are near the thumb.

In order to conduct an exhaustive comparison between the inference threat posed by different motion sensors present on a smartwatch and smartphone, we carry out additional experiments using the gyroscope data which is another widely studied side-channel for keystroke inference [20, 81, 21]. Due to the absence of a gyroscope sensor on the Motorola XT1028, we used another smartphone for this experiment, namely a Motorola XT1096 (paired with the Samsung Gear Live). The same experiment as

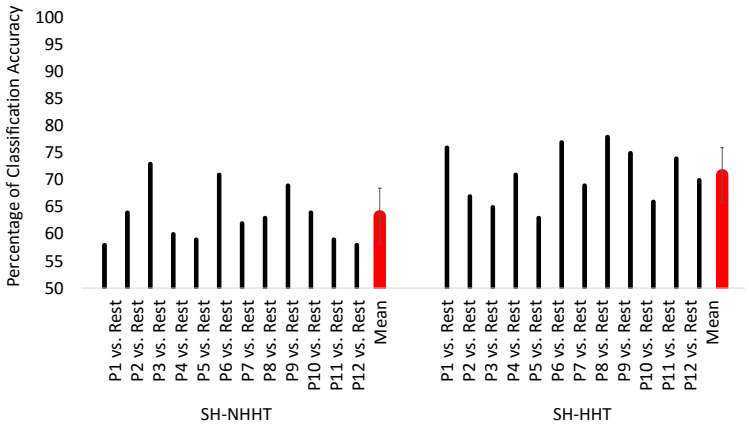


Figure 2.9: Classification accuracy for *One vs. Rest* using smartphone data.

above was repeated by 12 new participants, each typing 100 randomly dictated numbers. For this experiment, we recorded keystroke related motion data, comprising of both linear accelerometer and gyroscope measurements, from both the smartphone and the smartwatch. We derived 59 time and frequency domain features from the three-dimensional gyroscope data of both devices, such as minimum and maximum values, the mean value, variance, skewness, kurtosis, vertex angles, number of spikes, peak intervals, attenuation rate, etc. These features were selected from the literature on activity detection [7, 123] and keystroke inference [21]. Figure 2.11 shows the *One vs. Rest* classification accuracy results when solely the gyroscope features are used compared to when they are used in combination with features derived from the linear accelerometer measurements. The mean classification accuracy is marginally lower when using only the smartwatch gyroscope, compared to the smartphone gyroscope (SH-NHHT: 59.91% vs. 61.25%, SH-HHT: 59.66% vs. 64.75%). However, we can observe that after combining multiple motion sensors (linear accelerometer and gy-

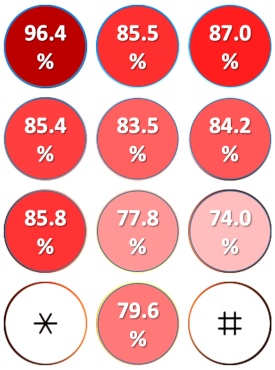


Figure 2.10: *All vs. All* classification accuracy for individual keys in SH-HHT using smartphone data, results averaged over all 12 participants.

roscope) on a device, the keystroke inference threat on the smartwatch is greater than the one on the smartphone (mean classification accuracy, SH-NHHT: 69.91% vs. 61.41% and SH-HHT: 71% vs. 66.08%).

2.5.5 Combining Smartwatch and Smartphone Data

After comparing the accuracy of keystroke inference attacks using individually both the smartwatch and smartphone motion data, we were intrigued to study the impact of combining or fusing motion sensor data from both devices in order to further reduce the number of classification errors. As most modern smartwatch operating systems and applications require the watch to be paired with a smartphone, such an attack is quite realistic. The feature vectors of *same* keystroke-records from both the devices were merged to obtain new feature vectors containing 310 features. We rebuild the classifiers with the larger feature vectors, and re-ran the previous experiments (as outlined in Sections 2.5.1 and 2.5.2). Results of these experiments (outlined in Table 2.2) show that indeed accuracy improved when the features from

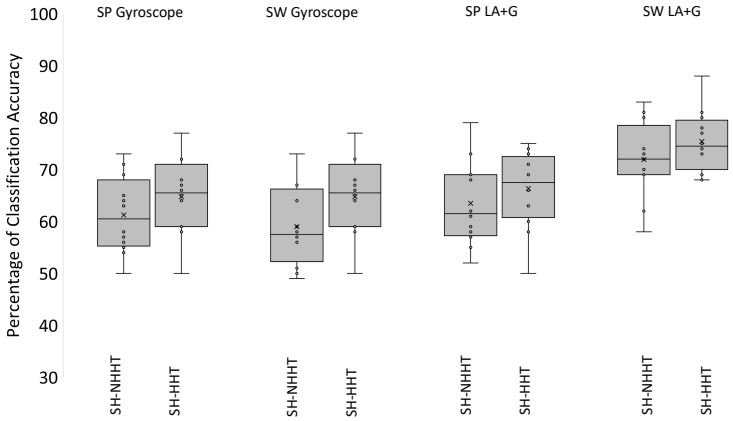


Figure 2.11: *One vs. Rest* classification accuracy using only gyroscope features, and in combination with linear accelerometer features. Results compared between smartwatch and smartphone.

both smartwatch and smartphone were combined. For example, the *One vs. One* classification accuracy in SH-NHHT was 90.66%, compared to 83.5% and 84.0% when individual smartwatch or smartphone data were used, respectively. Similar improvements can be observed in other scenarios as well. However, the improvement was relatively marginal, which can be attributed to the convergence in the learning process. Therefore, combining or fusing data from both smartwatch and smartphone may be more beneficial when the adversary has fewer training data.

2.5.6 A More Realistic Setting: Natural or Non-Controlled Typing

In all of the experiments so far, the participants were being directed (to tap) by an audio stream. Because participants have to hear the audio and then act on it, a minor delay or disturbance may be introduced in each key press. Moreover,

Table 2.2: Classification accuracy after combining features from both smartwatch and smartphone, results averaged over all 12 participants.

	SH-NHHT Combined (Smartwatch Only, Smartphone Only)	**SH-HHT Combined** (Smartwatch Only, Smartphone Only)
One vs. One	**88.91%** (84.5%, 78.7%)	**90.66%** (83.5%, 84.0%)
One vs. Rest	**71.59%** (70.0%, 63.3%)	**74.29%** (71.1%, 70.9%)
All vs. All	**88.65%** (88.1%, 85.5%)	**89.78%** (85.8%, 86.8%)

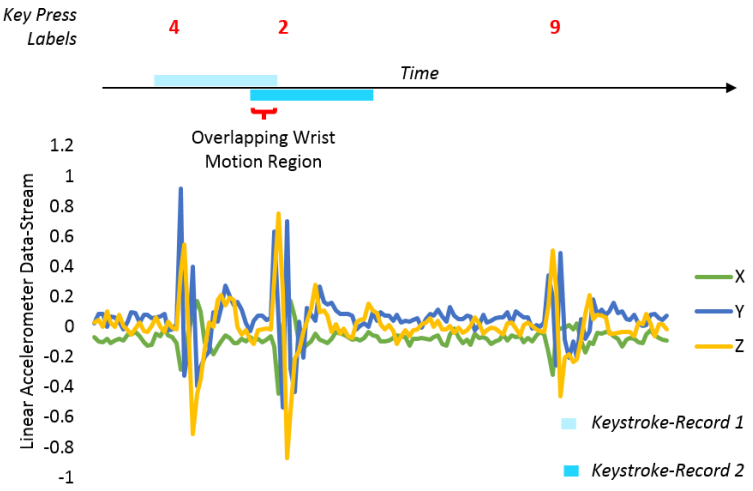


Figure 2.12: An example where rebounding motion of a key press overlapped with the next key press.

such a kind of typing or tapping does not invoke (and capture) users' natural typing behavior and speed. To evaluate a more natural typing behavior, we conduct another experiment where a new set of 10 participants were instructed to type their phone number followed by their residential zip code (a total of 15 numbers). These two pieces of information can be readily recollected by participants, thus eliminating any delay and/or disturbance while typing. This also enables us to capture more realistic typing or tapping data from users. However, for prediction we continue to use the classifiers trained earlier in the guided experiments (Section 2.5.2). The new data was processed by the same attack framework to extract keystroke-records and build feature vectors. We obtained a mean classification accuracy of 52% and 61% for SH-NHHT and SH-HHT, respectively. It was observed that the primary cause of drop in classification accuracy resulted from faster typing, where the rebounding motion of few key presses overlapped with their next key press (see Figure 2.12). Such instances were observed more often when two consecutive key presses were for number adjacent to each other on the keypad. Although the classification accuracy of naturally typed numbers is not as high as in the guided experiments, it is high enough to be a significant threat.

2.5.7 Cross Device Performance

In order to further evaluate how the proposed attack framework performs across different commercial wrist wearable or smartwatch hardware, we test our trained classifiers (from Section 2.5.2) on keystroke motion data obtained from a smartwatch of a different make and model. This simulates a situation where an adversary trains classification models using one type of smartwatch hardware and then employs those

models to infer the keystrokes of a target user who is using a completely different (possibly, unknown) smartwatch. Such a situation is much more realistic. For this set of experiments, we used a LG Urbane W150 smartwatch that has a InvenSense M651 accelerometer and gyroscope sensor and collected keystroke motion data from 12 completely new participants. Motion data corresponding to 100 keystrokes were collected from each of the new participants, and tested using the classifiers trained earlier in Section 2.5.2. The new data was collected at the same sampling frequency of 50 Hz. Results (Figure 2.6) show that while mean classification accuracy dropped slightly on the Urbane W150 (SH-NHHT: 70.08% vs. 67.41%, SH-HHT: 71.16% vs. 70.83%), the variance is significantly lower in case of the Gear Live (SH-NHHT: 26.62 vs. 56.26, SH-HHT: 17.24 vs. 77.0). Although such a trend is intuitive, it nevertheless shows that keystroke inference using the propose framework is still feasible with reasonable accuracy even in such a realistic setting.

2.5.8 Extending to QWERTY Keypads

Up until this point, our primary focus has been keystroke inference attacks on numeric mobile keypads. We now briefly investigate how our proposed attack framework performs against alphanumeric mobile keypads with the standard QWERTY layout. Intuitively, as the keys on a standard smartphone QWERTY keypad are relatively smaller and placed closer to each other (compared to keys on the numeric keypad), keystroke prediction may suffer from high confusion with neighboring keys [81]. We collected 1248 alphabet keystrokes from a completely new set of 12 participants using the LG Urbane W150 smartwatch, with equal distribution of alphabets (48 each). We then re-ran the training and attack modules in the *One vs. Rest*

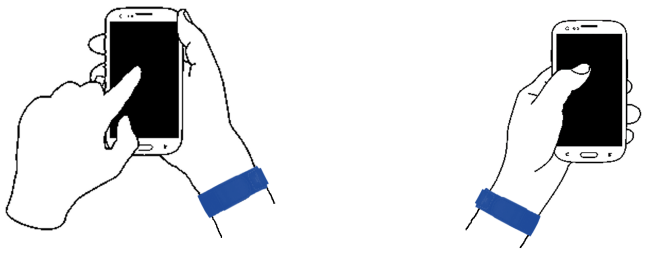
setting, with 75% data used for training and 25% data used for testing. Table 2.3 summarizes the classification accuracy of the 26 alphabets, along with two most confused keys predicted for each alphabet. As anticipated, the classification accuracy is significantly lower on the QWERTY keypad (compared to the numeric keypad), with an average accuracy of 30.44%. While the low classification accuracy of individual keys is prohibitive in carrying out effective inference attacks, it is important to note that the most confused keys are usually neighboring to the actual key. It is possible that we may be able to further improve the accuracy of these inference attacks by analyzing keyboard characteristics and/or performing a dictionary-based search [77, 71].

2.5.9 Variations of the SH-NHHT and SH-HHT Attack Scenarios

In addition to the SH-NHHT and SH-HHT scenarios presented in Figures 2.1(a) and 2.1(b), there is an additional variation for each of these scenarios, as shown in Figures 2.13(a) and 2.13(b). For SH-NHHT, the scenario 2.1(a) assumes that the smartphone and smatwatch is on the left hand (and users type with the right hand). A variation of this SH-NHHT scenario is having the smartphone and smatwatch on the right hand and typing with the left hand (2.13(a)). A similar variation (2.13(b)) can also be envisioned for the SH-HHT scenario 2.1(b). We would like to analyze whether the performance of our proposed keystroke inference framework differs significantly for these variations. The same experiment as in Section 2.5.2 was repeated by 12 new participants, each typing 100 randomly dictated numbers per variation. A two-tailed t-test on the *One vs. Rest* classification accuracies for the SH-NHHT variations in Figures 2.1(a) and 2.13(a) returned the value of $t = -0.46$,

Table 2.3: Classification accuracy of the 26 alphabets (in percent), along with two most confused keys predicted for each alphabet. Results averaged over all 12 participants.

Accuarcy	**1st Confusion**	**2nd Confusion**
a: 41.66	s: 25.00	z: 16.66
b: 25.00	v: 33.33	g: 16.66
c: 33.33	f: 25.00	v: 25.00
d: 16.66	s: 33.33	c: 25.00
e: 33.33	w: 33.33	d: 16.66
f: 16.66	d: 33.33	v: 16.66
g: 25.00	h: 58.33	b: 8.33
h: 33.33	g: 25.00	n: 25.00
i: 33.33	o: 25.00	u: 25.00
j: 16.66	h: 16.66	k: 16.66
k: 16.66	j: 41.66	m: 16.66
l: 25.00	k: 25.00	o: 16.66
m: 33.33	k: 25.00	n: 8.33
n: 25.00	h: 16.66	m: 16.66
o: 33.33	i: 16.66	l: 8.33
p: 50.00	o: 25.00	i: 8.33
q: 41.66	a: 16.66	w: 16.66
r: 25.00	e: 33.33	f: 16.66
s: 16.66	x: 25.00	z: 25.00
t: 33.33	f: 16.66	h: 16.66
u: 41.66	h: 33.33	k: 16.66
v: 25.00	c: 33.33	b: 25.00
w: 41.66	q: 25.00	e: 8.33
x: 41.66	z: 33.33	c: 8.33
y: 33.33	t: 41.66	u: 16.66
z: 33.33	a: 33.33	x: 8.33
Average Accuracy: 30.44		



(a) (b)

Figure 2.13: Variations of typing scenarios in Figures 2.1(a) and 2.1(b).

$p = 0.65$. For the SH-HHT variations 2.1(b) and 2.13(b), it returned $t = 0.61$, $p = 0.54$. Both results are not significant at $p < 0.05$, implying that our attack framework is not dependent on these variations. Therefore, an adversary can still use the same framework to carry out the inference attacks for these variations by simply retraining the classifiers.

2.6 Relative Transitions-Based Attack Framework

As discussed earlier, unlike the SH-NHHT and SH-HHT scenarios, key press events in the DH-HHT scenario (Figure 2.1(c)) cannot be uniquely and accurately detected on the smartwatch. To overcome this problem in the DH-HHT scenario, we leverage on the observation that transitional movement between each pair of keys produces characteristically unique motions on the wrist, which can be accurately captured by the smartwatch. Accordingly, for the DH-HHT scenario, the keystroke inference framework (Figure 2.14) leverages on supervised machine learning to first classify transitional movements between consecutive key presses. Then, assuming a reasonable distribution of numbers typed, *when multiple transitional directions in*

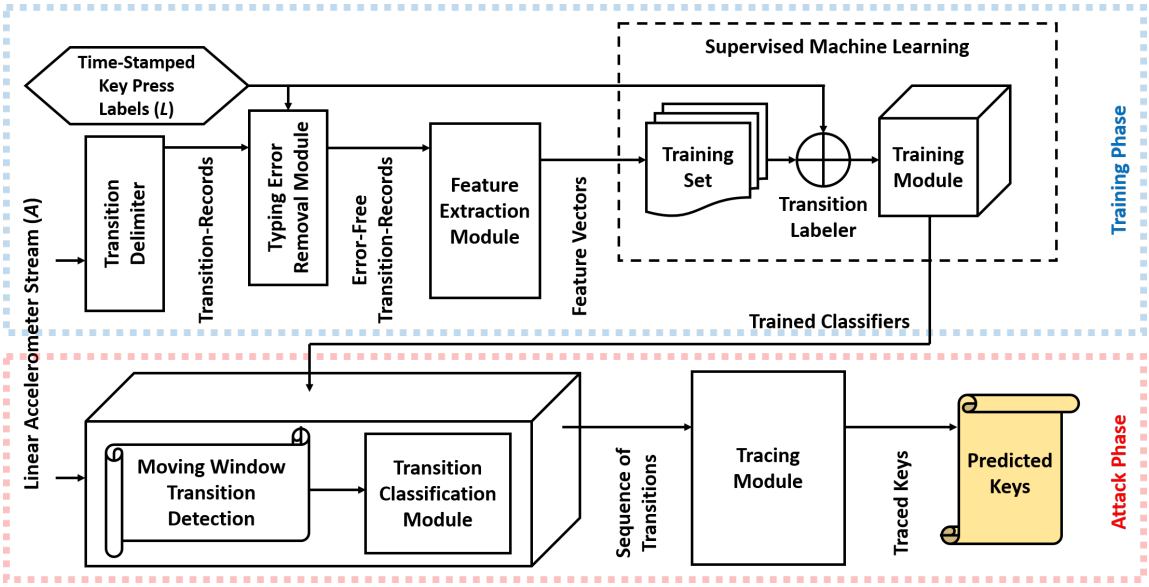


Figure 2.14: Overview of the relative transition-based attack framework for DH-NHHT typing scenario.

between a target sequence of key presses are traced on the key pad, we obtain a unique or highly reduced possibilities for the target sequence.

Data Collection and Pre-Processing: The same data collection application that was used for the SH-NHHT and SH-HHT scenarios is also used for the DH-NHHT scenario. Although the data collection process is exactly the same, the pre-processing operations are entirely different for DH-NHHT. Instead of detecting key press events, our goal here is to detect the type of wrist movement transition between every two consecutive key presses. As a result, we use the labeled stream of data to create labeled "transition-records" (Figure 2.15) and use them as the training set. To create the training set, all linear accelerometer samples between two consecutive key press events are used as the transition-record.

Transition Classification: We classify transitions based on cardinal directions. The logic behind such a classification is that transitions in the same direction results in similar wrist movement. For example, wrist movement between numbers 4 and 1 would be similar to wrist movement between 6 and 3 (North), wrist movement between numbers 4 and 7 would be similar to wrist movement between 6 and 9 (South), and so on. One classifier is trained for each possible transitional direction, as listed in Table 2.4: North (N), South (S), East (E), West (W), Northeast (NE), Northwest (NW), Southeast (SE), Southwest (SW) and Repeat (O). To achieve higher inference ability through tracing (explained later), the transition classifications must also be evenly populated. The number of possible transitions in each of the above nine categories follows a fairly even distribution, varying between 9 and 14.

As an adversary will not have access to labels L, in the attack phase we use a variable-length moving window to check and determine the occurrences of transitions. The moving window is used to traverse (in steps of one sample) the linear accelerometer data A in chronological order, and classify each window of linear accelerometer samples into one of the nine directions. The length of the window was varied from 10 samples (200 $msec$ at 50 Hz) to 100 samples (2 sec at 50 Hz), to capture the variable length intervals possible between key presses. When ten or more consecutive windows were classified to be in the same direction, the classification result was recorded and the centroid was used as the key press time to form transition-records.

Feature Extraction: Contrary to the previous direct classification-based attacks, where each key press event was denoted in a fixed time period, transition

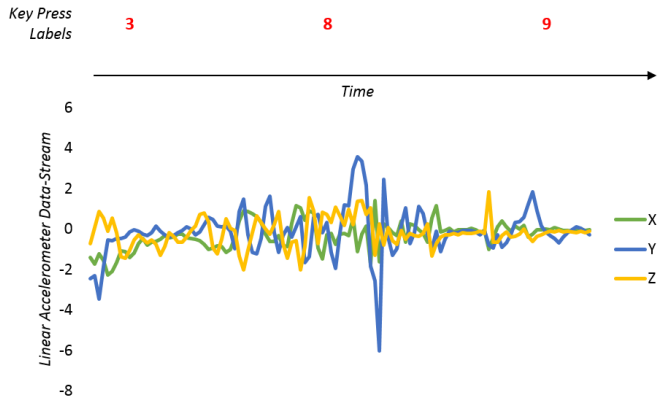


Figure 2.15: Time series of key press events in DH-NHHT, and their corresponding linear accelerometer readings. In DH-NHHT scenario, the wrist (along with the smartwatch) continues to move in between key press events. As a result, key press events cannot be identified or characterized based on spikes in energy level.

Table 2.4: Classification of all 100 possible numeric transitions.

N	4-1, 5-2, 6-3, 7-4, 8-5, 9-6, 0-8, 7-1, 8-2, 9-3, 0-5, 0-2, 0-1, 0-3
S	1-4, 2-5, 3-6, 4-7, 5-8, 6-9, 8-0, 1-7, 2-8, 3-9, 5-0, 2-0, 1-0, 3-0
E	1-2, 2-3, 4-5, 5-6, 7-8, 8-9, 1-3, 4-6, 7-9
W	2-1, 3-2, 5-4, 6-5, 8-7, 9-8, 3-1, 6-4, 9-7
NE	4-2, 5-3, 7-5, 8-6, 0-9, 4-3, 7-6, 7-2, 0-4, 8-3, 7-3
NW	5-1, 6-2, 8-4, 9-5, 0-7, 6-1, 9-4, 9-2, 0-6, 8-1, 9-1
SE	1-5, 2-6, 4-8, 5-9, 7-0, 1-6, 4-9, 1-8, 4-0, 2-9, 1-9
SW	2-4, 3-5, 5-7, 6-8, 9-0, 3-4, 6-7, 3-8, 6-0, 2-7, 3-7
O	1-1, 2-2, 3-3, 4-4, 5-5, 6-6, 7-7, 8-8, 9-9, 0-0

periods between two key presses can vary widely depending on typing habit, keypad size, key pairs, etc. As a result, many of the time domain features used in SH-NHHT and SH-HHT scenarios cannot be applied for DH-NHHT. Thus, we rely mainly on frequency domain features, such as FFT of individual axis readings of the transition-record, their mean, correlation, spectral roll-off, spectral centroid, spectral flux and power spectral density estimates, to learn and classify transitions.

Tracing and Recovery: To infer a target sequence of key presses, the proposed framework tries to "trace" the transitions between key presses on the numeric keypad. Tracing eliminates all non-fitting key-pairs (the pair of keys that may have been pressed before and after a transition) for each transition of the target sequence, where the fitness of a key-pair is determined by the preceding and following transitions. In case tracing results in a uniquely identified key-pair for a transition, the keys pressed before and after that transition can be directly inferred. In other cases where tracing results in multiple possible key-pairs for a transition, the keys pressed before and after that transition can either be inferred by multiple trials or from the other adjoining key-pairs (only if the adjoining key-pairs are uniquely identified).

After the transitions are classified, tracing of keys can be performed using one of the following strategies:

- *Forward Tracing:* The transitions are plotted on the keypad in the same order as they happened in time (Function $F\_Tracing()$ in Algorithm 3). In forward tracing, for a transition between candidate key pair (p, q), if there does not exists a pair $(*, p)$ that satisfies the directional classification of the preceding

transition, pair (p, q) is eliminated from possible key pairs for that transition. The $F\_Tracing()$ function works from left to right on the test sequence.

- *Backward Tracing:* The transitions are plotted on the keypad in the reserve order of how they actually happened in time (Function $B\_Tracing()$ in Algorithm 3). In backward tracing, for a transition between candidate key pair (p, q), if there does not exists a pair $(q, *)$ that satisfies the directional classification of the following transition, pair (p, q) is eliminated from possible key pairs for that transition. The $B\_Tracing()$ function works from right to left on the test sequence.

- *Bidirectional Tracing:* Both forward and backward tracings are applied to reduce the possibilities for the target sequence (Function $BD\_Tracing()$ in Algorithm 3).

We use bidirectional tracing in our evaluations because bidirectional tracing limits the propagation of any error that may be introduced by a transition misclassification.

2.7 Evaluation of Relative Transition Based Attack

In this section, we present the findings from our evaluation of the relative transition based attack framework.

2.7.1 Experimental Setup

The same experimental setup and participants as in Section 2.5.1 were used for DH-NHHT. The only difference was that the smartwatch was worn on the right hand by the participants, and the smartphone was held in the left hand.

Algorithm 3 Tracing Algorithms

$Transitions[N]$ (N transitions in target sequence)
$Directions[] = \{\emptyset\}$
$KeyPairs[] = \{\emptyset\}$
for $i = 1$ to N **do**
    $Directions[i] = Classify(Transitions[i])$
    $KeyPairs[i] = AllPossiblePairs(Directions[i])$
end for

function F_Tracing($KeyPairs[]$)
    for $j = 2$ to N **do**
        for each pair (p, q) in $KeyPairs[j]$ **do**
            if $\exists!$ a pair $(*, p)$ in $KeyPairs[j-1]$ **then**
                Remove (p, q) from $KeyPairs[j]$
            end if
        end for
    end for
return $KeyPairs[]$
end function

function B_Tracing($KeyPairs[]$)
    for $k = N - 1$ to 1 **do**
        for each pair (p, q) in $KeyPairs[k]$ **do**
            if $\exists!$ a pair $(q, *)$ in $KeyPairs[k+1]$ **then**
                Remove (p, q) from $KeyPairs[k]$
            end if
        end for
    end for
return $KeyPairs[]$
end function

function BD_Tracing($KeyPairs[]$)
return B_Tracing(F_Tracing($KeyPairs[]$))
end function

2.7.2 Constructing and Testing the Framework

We construct our transition classifiers based on training datasets of labeled transition-records generated by the same 12 participants who helped create the classifiers for the SH-NHHT and SH-HHT scenarios. The same audio stream of uniformly distributed random numbers between 0 to 9 guided the participants in typing 100 numbers. Out of the 1200 total numbers typed by all 12 participants, we use 960 numbers (having 948 transitions) for training and rest for testing. We test the accuracy of the transition classifiers and tracing algorithms using two 10-digit long number sequences per participant (24 total test sequences, 240 total numbers, and 216 total transitions). We calculate the accuracy of the different tracing algorithms based on the number of correctly identified key presses in the traced number sequence. *In order to infer a key, at least the preceding or following transitions should be uniquely identified.* For example, in the instance shown in Figure 2.16, the transition 9 to 2 and 2 to 0 both have other contending key-pairs (the incorrect transitions which are not removed by the tracing algorithm because they fit in the overall sequence of transitions). In such cases, it becomes impossible to determine the exact key (2 in this example) pressed in one trial. However, although the transition 2 to 0 have other contending key-pairs, the pressing of key 0 can be inferred with the help of the uniquely identified 0 to 7 transition, following the key press. In case both the preceding or following transitions are uniquely identified, the adversary can be more confident about the inference. One may also notice that the first and last number in a sequence are harder to infer, as there exists only one transition for each.

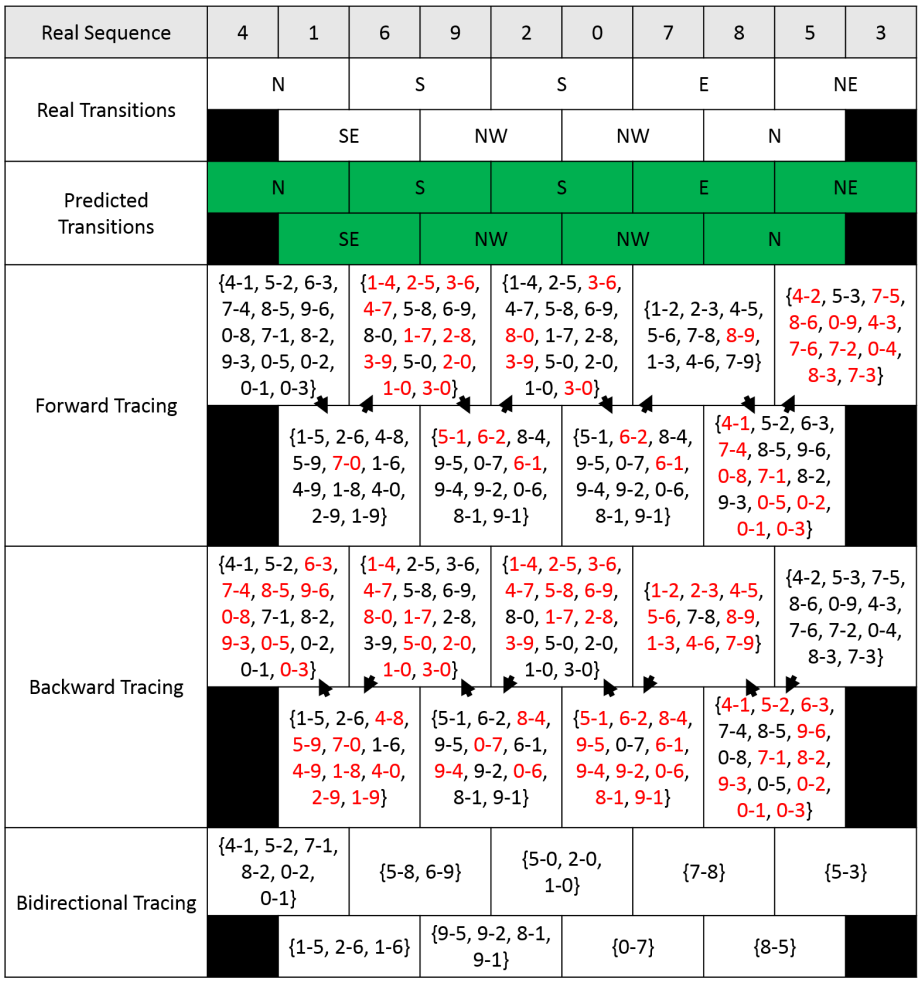


Figure 2.16: An example of how bidirectional tracing drastically reduces the possibilities of the key presses. First the forward tracing eliminates incompatible transitions (in red) in chronological order. Then the backward transition removes additional incompatible transitions in chronologically reverse order. In this example, we are able to uniquely identify the last 4 key-pairs using bidirectional tracing, which allows unambiguous inference of the last 5 key presses.

In our evaluation, the transition classifiers were able to correctly classify 191 transitions-records (88.42% accuracy), while the remaining 25 incorrect or unclassified transitions introduced error in 17 of the test sequences. We also observe that an incorrect prediction is more likely to occur immediately after a previous incorrect prediction. One of the possible explanations behind such an observation is that the transition behavior varies depending on the preceding and following transitions. In terms of inference accuracy, 85 key presses out of the 240 test numbers were unambiguously identified using the bidirectional tracing (43.75% accuracy). We relate the relatively low inference accuracy to three primary reasons: (a) incorrectly classified transitions introduce error in not one but two key presses, (b) unclassified transitions do not introduce error but there is no remedy to fill in the missing information, and (c) even a very small number of contending key-pairs makes it impossible to determine the exact key pressed.

Because most of today's information systems acknowledge natural human mistakes and allows multiple trials to validate security tokens (pin, password, card number, etc.), adversaries can easily take advantage of it to try all possible number sequences derived from the output of bidirectional tracing. Accordingly, we evaluate the inference accuracy using multiple trials (solving from left to right), up to the maximum number of trials required to correctly infer the full number sequence. For example, in the instance shown in Figure 2.16, there can be 21 possible sequences derivable from the output of bidirectional tracing (listed in Table 2.5). Results of multiple trials are presented in Figure 2.17, where we see that more ambiguous sequences require additional number of trials (in the worst case). We do not restrict

Table 2.5: The 21 possible number sequences that satisfy the bidirectional tracing obtained in Figure 2.16.

4-1-5-8-1-0-7-8-5-3	7-1-5-8-1-0-7-8-5-3	0-1-5-8-1-0-7-8-5-3
4-1-6-9-5-0-7-8-5-3	7-1-6-9-5-0-7-8-5-3	0-1-6-9-5-0-7-8-5-3
4-1-6-9-2-0-7-8-5-3	7-1-6-9-2-0-7-8-5-3	0-1-6-9-2-0-7-8-5-3
4-1-6-9-1-0-7-8-5-3	7-1-6-9-1-0-7-8-5-3	0-1-6-9-1-0-7-8-5-3
5-2-6-9-5-0-7-8-5-3	0-2-6-9-5-0-7-8-5-3	8-2-6-9-5-0-7-8-5-3
5-2-6-9-2-0-7-8-5-3	0-2-6-9-2-0-7-8-5-3	8-2-6-9-2-0-7-8-5-3
5-2-6-9-1-0-7-8-5-3	0-2-6-9-1-0-7-8-5-3	8-2-6-9-1-0-7-8-5-3

the adversary to a certain number of attempts (which would be system dependent) because the actual sequence may or may not be tried in the limited number of attempts. Instead, we evaluate the worst case scenario, where the adversary has to try all possible sequences derived from the output of bidirectional tracing. Note that we evaluate this using only the 7 bidirectionally traced sequences for which all the predicted transitions are correct.

2.7.3 Combining Smartwatch and Smartphone Data

As smartphones cannot capture transitional wrist movements of the typing hand, we cannot merge feature vectors like it was done in SH-NHHT and SH-HHT scenarios. As an alternative, we found a novel way to combine smartphone motion sensor data due to key taps, which was used in the classification-based attacks (as evaluated in Section 2.5.5), with the smartwatch transition-records obtained in DH-NHHT scenario. Based on the previous observation that classifying transition-records itself is highly accurate (88.42% accuracy), we continue using the same attack framework. However, to overcome the limitations faced in the inference process (after tracing is

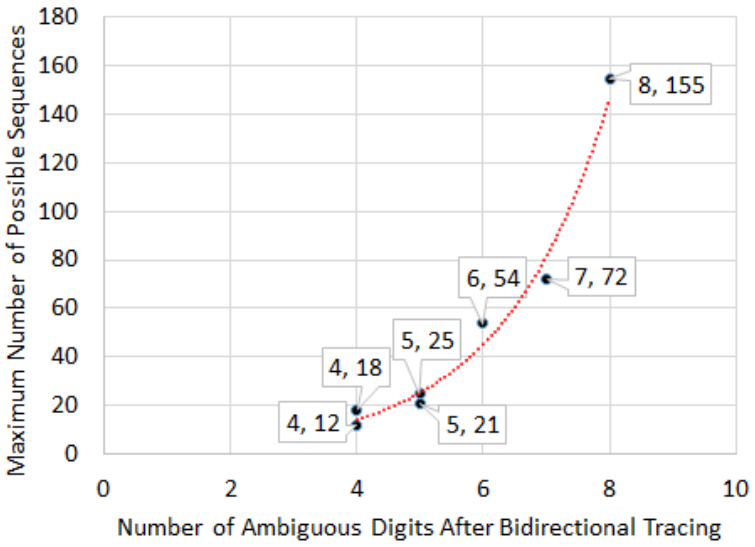


Figure 2.17: More ambiguously traced sequences require additional number of trials (in the worst case).

completed), classified smartphone keystroke-records may be used to choose from the multiple candidate sequences obtained as the output of the tracing algorithm. We evaluate this attack using linear accelerometer readings of the smartphone, which were additionally collected during the DH-NHHT experiments of previous section. Keystroke-records and feature vectors are extracted from the smartphone data as it was done for SH-NHHT and SH-HHT. Elimination of contending key-pairs and filling up of undetected transitions with the help of classified smartphone keystroke-records (combined with reasonably accurate classification of keystroke-records), resulted in 82.50% unambiguous inference of key presses. This is a substantial improvement in the inference accuracy, compared to the 43.75% accuracy obtained earlier without the help of classified smartphone keystroke-records.

2.7.4 A More Realistic Setting: Natural or Non-Controlled Typing

Similar to SH-NHHT and SH-HHT experiments, the participants were being directed by an audio stream in the above DH-NHHT experiments, which may introduce a minor delay or disturbance in each key press. As a result, we conducted a similar natural or non-controlled typing experiment in the DH-NHHT scenario, where a completely new set of 12 participants were instructed to type their phone number followed by their residential zip code (15 numbers, 14 transitions). In this setting, out of a total of 168 transitions-records, 134 were classified correctly (79.76% accuracy). The remaining incorrect or unclassified transitions introduced error in test sequences of 10 participants. Out of a total of 180 key presses, 69 were unambiguously identified using the bidirectional tracing, thus giving an accuracy of 38.33%.

2.8 Discussion

2.8.1 Limitations

Posture and Ambient Movement: In practice, wrist movement patterns may change drastically based on the target user's body posture and orientation. In other words, the key press features while sitting may differ substantially from the key press features while laying down. One main limitation of our attack framework is that it is not robust against such different body postures and orientation. In order to overcome this, an attacker must train multiple classification models using data corresponding to different user postures and orientations, and then apply the appropriate one for the victim. This, if the attacker knows what was the victim's posture while typing.

Similarly, if the target user is moving (for example: walking, running, sitting inside a car or train, etc.) while typing, keystroke events in the accelerometer/gyroscope data may get masked and our framework may not be able to correctly infer them. However, we must point out that this issue is not specific only to our attack framework, but other frameworks in the literature suffer from a similar drawback.

Power Consumption: Another limiting factor of our attack can be the power consumption rate on the smartwatch, due to the continuous recording of sensor data at a high frequency. For instance, the 300 mAh battery inside the Samsung Gear Live dropped from 100% to 69% in an hour, while recording linear accelerometer readings at 50 Hz. This limitation is less evident in case of smartphones due to their significantly higher battery capacity. To carry out a stealthy attack using the smartwatch, an attacker may have to either reduce the sensor sampling rate, or devise a mechanism to start the recording only when the potential victim is typing.

Both Hand Typing: We cover three major typing styles in this chapter, while missing the case where a user holds the smartphone and types using both hands. In this scenario, the motion captured by the smartwatch will vary depending on which thumb is used to type a key, and which hand the smartwatch is worn on. The movement captured in this typing scenario will yield very different results and requires a new inference technique.

Threats to Validity: Most of our experimental results were obtained from analysis of keystrokes typed in a relatively controlled setting, where participants were dictated on what to type. As a result, it is possible that those results may not be representative of how our attack framework may perform in more natural typing scenarios. However,

we must point out that we do investigate the efficacy of our attack framework in several natural typing scenarios (in Sections 2.5.6 and 2.7.4), and the obtained results show that our inference framework has reasonable accuracy in these scenarios as well.

2.8.2 Defenses

Defending against side-channel attacks is a much debated topic [22]. Although modern mobile and wearable operating systems offer access control on some sensors, sensors such as accelerometer and gyroscope cannot be disengaged by the user. Moreover, most mobile applications do not require explicit permissions (either at install or run time) in order to access these sensors. A straightforward defense approach is to safeguard all sensors using system or user-defined access controls. However, such a static access control will become increasing complex to manage and will not protect against applications that gain legitimate access to these sensors. Reducing the frequency at which applications can sample data from these sensors is another potential defense mechanism. A system-level monitoring mechanism that tracks the context and frequency of sensor accesses, and appropriately flag unwanted accesses requested by applications, could also serve as a useful defense tool.

2.8.3 Enhancements

Random Walk Tracing: This is a tracing algorithm we propose for use with very long number sequences typed in DH-NHHT scenario. In this tracing algorithm, a random subsequence of varying length is selected and bidirectional tracing is applied. The process is repeated several times such that every transition is covered multiple times, and each key press may end up having multiple candidate keys. Majority vot-

ing may be used to determine the final predicted keys (only if a key press has multiple candidate keys). This tracing algorithm will greatly minimize the propagation of any error that may be introduced by a transition misclassification.

2.9 Conclusion

In this chapter, we comprehensively investigated the feasibility of keystroke inference attacks on mobile numeric keypads by using smartwatch motion sensor data as an information side channel. We proposed two supervised learning-based frameworks to infer keystrokes from smartwatch motion data in three popular mobile holding and typing scenarios. We empirically evaluated the performance and efficacy of our proposed inference frameworks under various experimental settings (i.e., controlled versus natural typing), by using different types of smartwatch hardware, by using different types of motion sensors (i.e., accelerometer versus gyroscope) and by fusing motion data from multiple sources (i.e., smartphone and smartwatch). We also evaluated the performance of our attack framework on alphanumeric mobile keypads with a QWERTY layout. Results from our various experimental studies have shown that typing-induced motion data captured by smartwatch sensors can be employed as an effective side-channel to infer keystrokes on mobile keypads.

Parts of this chapter appeared in [74, 75].

CHAPTER 3
ATTACKS ON CYBER INTERACTIONS: EXTERNAL KEYBOARDS

3.1 Introduction

In this chapter, we show that unaudited access to motion sensors featured on most smartwatches can inadvertently lead to significant leakage of information relating to users and their surrounding. We demonstrate that a malicious application, with access to motion sensor readings of a smartwatch, can decode the keystrokes made on a QWERTY keyboard while wearing the smartwatch on one hand. We achieve this based on the observed relative physical position of keystrokes and direction of transition between pairs of keystrokes. We then recover the typed words by mapping the captured 'motion' of each word to pre-formed motion profiles of words in an English language dictionary. Due to the distinctive nature of perceptible sensor data on smartwatches, straightforward adaptation of earlier side-channel keystrokes attacks based on emanation of electromagnetic, acoustic or vibration pulses generated by a keystroke, is not befitting. A comprehensive empirical evaluation of our keystroke inference framework show significantly high word recovery rates.

3.2 Related Work

Emanation based side-channel inference attacks date back to the World War II era [35]. The primary types of emanations include electromagnetic signals, sounds, and vibrations. Previous studies demonstrated the use of electromagnetic emanation

to eavesdrop on contents displayed on a CRT or LCD screen [112, 61] from a distance and with opaque obstacles in between. Similar attacks using electromagnetic emanations have also been shown to work against CPU chips [5], smart cards [89], data carrying cables [99], and keyboards (wired or wireless) [113]. Optical emanation, contained in the band of electromagnetic spectrum perceptible to human eyes, present a different form of leakage for display devices. The light released from display devices may reflect off various surfaces in front of the screen, and reach an eavesdropper. Successful reconstruction of the displayed information has been demonstrated based on reflection such as from walls [60], shiny objects [12], and even from viewer's eyes [10]. While electromagnetic emanation based attacks are certainly effective, the need of specialized equipment and it's concealed placement near to the target poses difficulty. Similarly for side-channel attacks based on optical emanations, the eavesdropping equipment must be placed in line of sight of the target.

Side-channel attacks based on acoustic or sound emanations are much more feasible because of the popularity of personal devices featuring microphones. Microphones are inexpensive, and can be easily concealed because of their compact form factor. Furthermore, if a target's microphone enabled device (such as smartphones, tablets, etc.) is hijacked, it can act as a disguised eavesdropping equipment. As much as 90% of English text printed by a dot-matrix printer can be successfully recovered, by learning the acoustic emanations released by the printer [11]. The other major use of acoustic emanation has been in keystroke inference attacks, which targets to recover key presses on a nearby computer keyboard [8, 16]. Similar keystroke inference attacks can be carried out using surface vibration emanation generated during

keystrokes [77, 14]. Vibrations of nearby surfaces caused by human voice can also be recorded, and used to decode speeches [79]. While systems to record vibrations may be difficult to conceal, Marquardt et al. [77] proposed the use of a smartphone's accelerometer to record vibrations near keyboards. If an adversary is able to infect their target's smartphone with a malicious application which can record and transmit sensor data stealthily, it can serve as a very effective eavesdropping tool.

However, a critical requirement of learning based side-channel attacks using electromagnetic, acoustic, or vibration emanation, is that the target and eavesdropping equipment must not be disturbed. Change in either's position or orientation will render the training data futile, thereby making recovery of target information impossible. This also means that training must be performed in the same setting as the attack, which may not always be feasible. For example, in case of [77], if the target person puts his/her smartphone one day on the left side of the keyboard and another day on the right side, the vibrations captured by the accelerometer will be significantly different, resulting in failed recovery of typed text. Our attack setting, which uses motion data from a wearable device to infer keystrokes, is largely unaffected due to similar constraints as most people wear and use these devices in a very standard fashion (for example, smartwatches are almost always worn on the left wrist by most people). Moreover, our attack mechanism and wrist motion characterization framework is very general and can be easily extended to work in scenarios comprising of non-traditional usage of these wearable devices (for example, users wearing the watch on the right hand instead).

During the final phase of completing this work, we came across recently published works which demonstrate the ability to infer keystrokes using smartwatch. In the previous chapter [74], we used machine learning to train classifiers based on the slight differences in wrist movements observed while tapping numeric keys on a handheld smartphone keypad, depending on the location of the key on the screen. The trained classifiers are then used on test data to perform multiclass classification between the ten keys. Similar to our work, Wang et al. [115] demonstrate the feasibility of keystroke inference attack using a smartwatch, on a QWERTY keyboard. However, their attack framework is very different from ours. We also conduct a comprehensive evaluation of our attack framework and preliminary results indicate that our approach leads to better inference accuracy compared to [115]. However, [115] has a different experimental setup, due to which we are unable to make a comprehensive comparison like we do with [77] and [16].

3.3 Attack Description

In this chapter, we demonstrate the feasibility of a keystroke inference attack against a user typing on an external QWERTY keyboard by using smartwatch motion sensors. Because of limitations faced by emanation-based keystroke inference attacks, and multiple technical challenges in implementing them on a smartwatch, we pursue a slightly different approach for our attack where we focus on capturing and using keystrokes related wrist motion or movement characteristics.

We observed that the wrist movements made while typing a fixed sequence of letters on a keyboard are highly similar and consistent across multiple trials involving

a single typer. This gave us the intuition that an adversary can create a dictionary of commonly used words (words are nothing but fixed sequence of letters), along with their corresponding wrist movement patterns. During the attack, the adversary can simply match the eavesdropped wrist movement pattern to the closest matching pattern in the dictionary. Intuitively, the recovery can be highly accurate if the dictionary is carefully created and comprises of all words that the target is expected to type. However, the recovery rate also depends on how the wrist movement patterns are characterized (which we will explain in Section 3.4.1) and the granularity of the captured wrist movement data (which is generally limited by the eavesdropping sensor's maximum sampling frequency).

For carrying out the proposed inference attack, an adversary requires an eavesdropping device that is capable of continuously recording wrist movements, while avoiding detection. A modern commercial-off-the-shelf smartwatch, which is generally equipped with a range of sensors (especially motion sensors), can easily serve as such an eavesdropping device. Other forms of wrist wearable devices such as activity trackers and fitness bands, are typically also equipped with motion or inertial sensors, and can also be used as an eavesdropping device for the proposed attack. In this work, without loss of generality, let's assume that the adversary exploits the smartwatch as an eavesdropping device. However, a bigger challenge is how does an adversary gain access to the motion data captured on the smartwatch. This can be achieved by an adversary installing a malicious application that has access to the motion sensors on the target's smartwatch such that the application is able to stealthily capture and transfer the captured motion data to the adversary.

Figure 3.1: An exemplary setup where a person is typing on a QWERTY keyboard, while wearing a Samsumg Gear Live smartwatch on left hand. A similar setup is used in our experiments.

This is feasible because, even though an application's access to the some sensors (e.g., GPS and camera) is generally user-managed or restricted on most modern mobile operating systems such as Android and iOS, access to motion sensors (such as, accelerometer and gyroscope) remains highly unregulated. An adversary can easily install the malicious application on the target smartwatch by various means, for example, by gaining physical access to the device or through social engineering (e.g. masquerading as a legitimate application, pretexting, baiting, phishing etc.). The malicious application can then stealthily collect and transfer motion data by masquerading as, or piggy backing on, useful application data and network traffic. In other words, the infected smartwatch now acts as an eavesdropping device that the targets' themselves place on their wrist, and unsuspectingly have it on their wrist while typing on a keyboard, as depicted in Figure 3.1.

The malicious adversarial application on the smartwatch records the linear accelerometer data (linear accelerometer measures the acceleration experience by the device, excluding the force of gravity) and microphone data. In the proposed attack, the acoustic data recorded by the microphone is not used for keystroke inference, but rather just to identify keystroke events (as explained in detail in Section 3.4.2.2). Due to the impracticality of an on-screen keyboard on the small smartwatch screens, an adversarial smartwatch application can seek access to the microphone in order to support voice commands or dictation, which is common. Alternatively, keystroke events can also be recognized by solely using the motion sensors, as accomplished in Marquardt et al. [77]. As mentioned earlier, the recorded sensor data is then transmitted by the malicious application to the adversary directly over the Internet by masquerading as useful communication or by piggyback on communications from other applications. In an effort to save battery power (necessary for avoiding detection), the recording and communication process may be initiated remotely by the adversary or based on periodic activity tracking.

3.4 The Attack Framework

In this section, we present our model for identifying key-press events from raw motion sensor data. We then discuss our attack framework, and an experimental setup for evaluating the framework.

3.4.1 Modeling Key Press Events

With the maximum supported linear accelerometer sampling rate (∼50-70Hz) being much lower than that of smartphones (∼200-300Hz), the difficulty in recognizing individual keys is greatly increased when using a smartwatch. To overcome this shortcoming, we attempt to identify *pairs* of key presses or keystrokes by learning the relationship between them. While typing a word, there will be one key press for each character or letter in the word. Let K_i, K_j be two consecutive key press events, signifying two consecutive characters or letters of a word. We characterize the relation, $rel(K_i, K_j)$, between any two consecutive key press events K_i, K_j as follows:

- **Horizontal Position:** The location $loc(K_i)$ of each keystroke event relative to a 'central-line' dividing the keyboard into left (L) and right (R) halves. *The rationale behind this classification is that the wrist movement will be more pronounced for typing a key on the same side as the watch-wearing hand.*

- **Transitional Direction Between Consecutive Key Presses:** The direction $dir(K_i, K_j)$ represents the direction of wrist movement between consecutive key presses K_i and K_j on watch-wearing side of the keyboard. The possible directions (or values for $dir(K_i, K_j)$) are N, E, S, and W, representing geographical north, east, south, and west, movement respectively. An additional classification is O, if $K_i = K_j$. *The rationale behind this classification is that the direction of transition between a pair of keystrokes will be reflected in the wrist movement.*

With the above classification, the relationship between two consecutive key press events is defined as follows:

- When either K_i, K_j, or both, occur on the non-watch wearing side of the keyboard, $rel(K_i, K_j) = loc(K_i) \mid\mid X \mid\mid loc(K_j)$, where '$X$' implies that direction cannot be determined. The intuition behind such an assignment is that it is not possible to determine the direction of transition when at least one of the pressed key is not on the watch-wearing side of the keyboard.

- When both K_i and K_j occur on the watch-wearing side of the keyboard, $rel(K_i, K_j) = loc(K_i) \mid\mid dir(K_i, K_j) \mid\mid loc(K_j)$.

A *word-profile* for a word can then be derived by concatenating the relation between every consecutive pair of letters in the word. For example, the word "boards" can be broken down in to five pairs of keystrokes {bo, oa, ar, rd, ds}, i.e., word-profile for the word "boards" is $rel(bo).rel(oa).rel(ar).rel(rd).rel(ds)$. With the setup for a QWERTY keyboard, as shown in Figure 3.2, and the entire L/R and N/E/S/W/O classification listed in Table 3.1, the word-profile of "boards" will be:

RXR . RXL . LEL . LSL . LWL

The main idea behind our attack is that the adversary will have a pre-processed dictionary of well-known (or targeted) words and their corresponding word-profiles (formed as discussed before). These word-profiles are used in distinguishing between candidate words from the dictionary. Given the motion data, the adversary will attempt to infer word-profiles from the motion data and then use the pre-processed

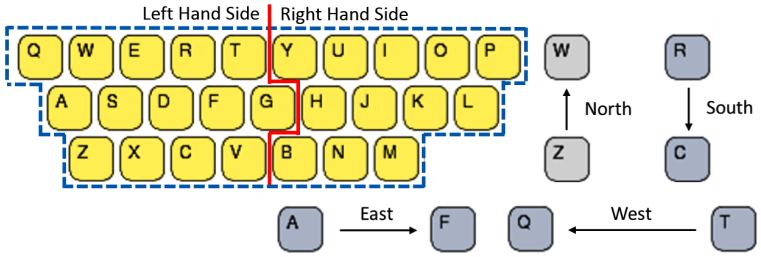


Figure 3.2: The keyboard is divided in to left (L) and right (R) halves, shown by the solid red line. Examples of N, E, S, and W classification are also shown. Each direction has 90 degrees field of view from center of the key. Keys that fall on the boundary are categorized in the direction where greater area of the key lies.

dictionary to determine the typed word by comparing the inferred word-profile to the word-profile in the dictionary. However if the dictionary is large, more than one word may have the same word-profile. Such *collisions* may result in incorrect predictions, and thus, reduce the accuracy of the inference attack by the adversary. In such cases, a frequency-based selection (as discussed in Section 5.15) could yield better word recovery results. Similarly, defining word-profiles by using additional fine-grained directional data (e.g., NE, SW, etc.) could reduce the number of collisions and improve inference accuracy, however it will also increase the attack execution time for the adversary.

3.4.2 Keystroke Inference Attack

Broadly, our proposed inference attack comprises of a *learning phase* (Figure 3.3) that is followed by the *attack phase* (Figure 3.4). However, before initiating the learning phase, the adversary must define the classification parameters. This includes deciding the keys in L and R halves, determining the hand on which the smartwatch

Table 3.1: L/R classification of individual keys and N/E/S/W/O classification of character-pairs, assuming smartwatch is worn on left hand.

L	q, w, e, r, t, a, s, d, f, g, z, x, c, v
R	y, u, i, o, p, h, j, k, l, b, n, m
N	aq, aw, sw, se, de, dr, fr, ft, gt, zq, zw, ze, za, zs, xw, xe, xr, xs, xd, ce, cr, ct, cd, cf, vr, vt, vf, vg
E	qw, qe, qr, qt, qs, qd, qf, qg, qx, qc, qv, we, wr, wt, wd, wf, wg, wc, wv, er, et, ef, eg, ev, rt, rg, ae, ar, at, as, ad, af, ag, ax, ac, av, sr, st, sd, sf, sg, sc, sv, dt, df, dg, dv, fg, zr, zt, zd, zf, zg, zx, zc, zv, xt, xf, xg, xc, xv, cg, cv
S	qa, qz, wa, ws, wz, wx, es, ed, ez, ex, ec, rd, rf, rx, rc, rv, tf, tg, tc, tv, az, sz, sx, dx, dc, fc, fv, gv
W	wq, eq, ew, ea, rq, rw, re, ra, rs, rz, tq, tw, te, tr, ta, ts, td, tz, tx, sq, sa, dq, dw, da, ds, dz, fq, fw, fe, fa, fs, fd, fz, fx, gq, gw, ge, gr, ga, gs, gd, gf, gz, gx, gc, xq, xa, xz, cq, cw, ca, cs, cz, cx, vq, vw, ve, va, vs, vd, vz, vx, vc
O	qq, ww, ee, rr, tt, aa, ss, dd, ff, gg, zz, xx, cc, vv

is worn, and accordingly form all perceptible transitions. In our experiments, we suppose that the target is wearing the smartwatch on his/her left hand and the keyboard is divided in L and R halves as shown in Figure 3.2. Accordingly, all 196 possible transitions with the watch-wearing hand are listed in Table 3.1. However, the proposed attack could easily be modified (with little effort) for the watch worn on the right hand or for other forms of L/R division of the keyboard. After these parameters are determined, the learning phase can begin.

3.4.2.1 Learning Phase

The purpose of this phase is to construct trained classification and prediction models for use during the attack phase. Training of these models comprises of the

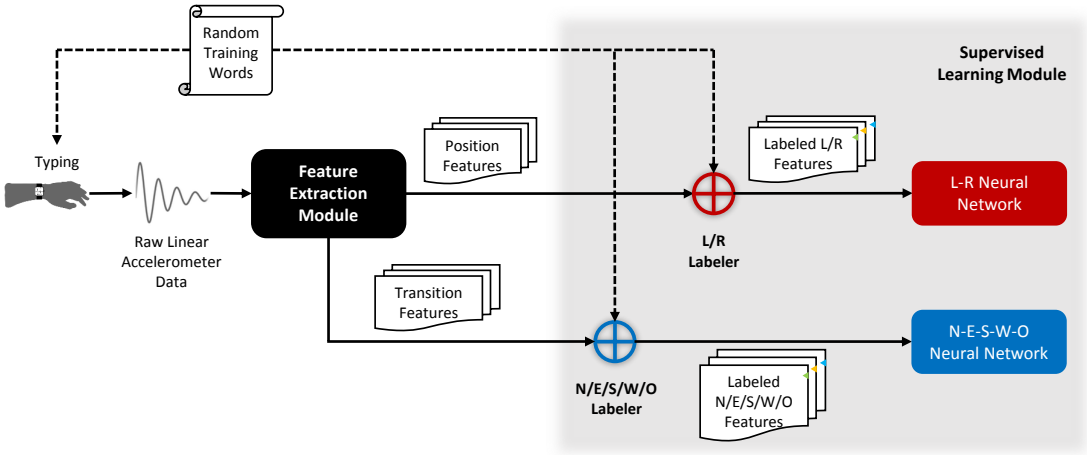


Figure 3.3: Learning Phase: A high level overview of the data processing architecture used to train the neural networks.

following four steps: (i) *data collection*, (ii) *feature extraction*, (iii) *word labeling*, and (iv) *supervised learning*. To ensure uniformity in the learning models, the training data is chosen such that it has equally distributed features. This can be achieved by using a large set of randomly generated words, uniformly covering all keys and apprehensible transitions.

Data Collection: There are two types of data recorded by our custom Android Wear attack (or data collection) application. First, is the motion data just before and immediately after a keystroke. Second, is the entire transition data between two keystrokes that occur on the watch-wearing side of the keyboard. Both types of recorded data are the *linear accelerations* experienced by the smartwatch, as sensed by it's linear accelerometer sensor. The sampled linear accelerometer readings are composed of instantaneous three dimensional linear acceleration along the X, Y, and Z axes. One of the authors (pretending to be the adversary) typed a set of

1000 random English words which uniformly covered all 26 keys and 196 transitions, without any fixed ordering or timing. Note that the number of possible transitions will be 144 if the target wears the smartwatch on the right hand and the keyboard is divided into the same L and R halves. The data collection application also clocks and tags the ground truth of the typed keys, which helps simplify the feature extraction and labeling process later, which in turn, ensures error-free training.

Feature Extraction: Feature extraction aids in dimensionality reduction by eliminating redundant measurements. For (L/R) keystrokes we compute a comprehensive set of 24 type of features such as mean, median, variance, standard deviation, skewness (measure of any asymmetry) and kurtosis (to measure any peakedness). We use multiple inter and intra-axis time domain features to capture the correlation between movement on the three axis, and frequency domain features to identify the different rebounding (or oscillatory) motion of the wrist. However, in case of (N/E/S/W/O) labeling, we observed that the transition period was varying widely based on typing speed and word composition. As a result, it is impossible to represent the entire transition in a fixed length time-domain feature vector (as used in feature vectors with (L/R) labels). As a solution, we use frequency-domain features such as Fast Fourier Transformation (FFT) of the transition data.

Labeling: Each training word is broken down into its constituent characters and character-pairs. As a result, a word of length n letters would be broken into n characters and $n-1$ character-pairs. Feature vector of each keystroke is labeled (L/R) using the ground truth characters recorded during data collection. Feature vectors of directions are labeled (N/E/S/W/O) by calculating the direction between

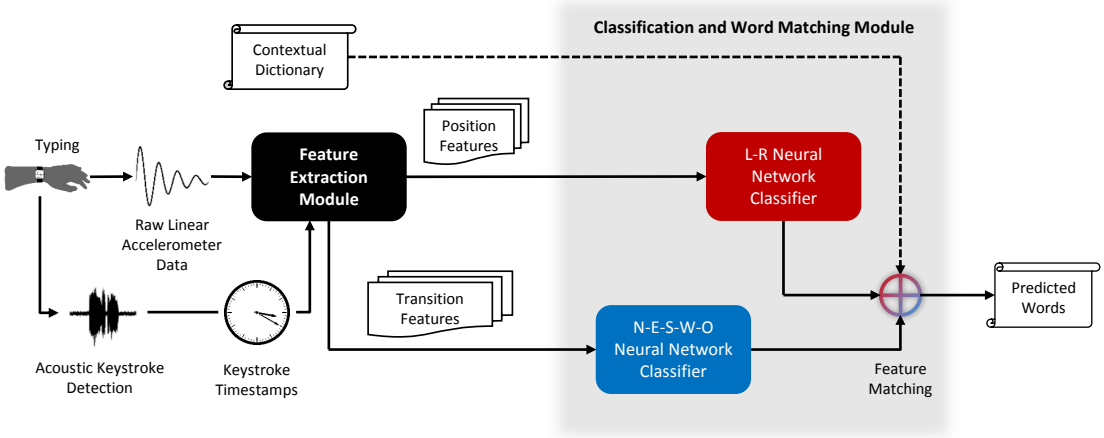


Figure 3.4: Attack Phase: A high level overview of the data processing architecture used to analyze keyboard input using the trained neural networks.

character-pairs obtained from the same ground truth. An additional processing is performed to select a set of character-pairs with even distribution of L and R and N, E, S, W and O labels. Note that the number of N, E, S, W and O labels will be approximately one-fourth of the number of L _ L, L _ R, R _ L and R _ R pairs because the direction is determined only in case of L _ L transition (or R _ R if the target wears the smartwatch on right hand). For L _ R, R _ L and R _ R character-pairs, the direction for transition cannot be determined (which is denoted by X in the word-profile), and thus, they are not used in the training phase.

Supervised Learning: We created two separate training models that will be used during the attack phase to classify keystrokes and keystroke-pairs. The two trained models are *L-R* and *N-E-S-W-O* neural networks for classifying (L/R) and (N/E/S/W/O) feature vectors, respectively. Because of the complex interactions possible between consecutive keystrokes, we train our classifiers using *neural net-*

works. Neural networks are specifically useful in discovering these complex interactions between the corresponding feature vectors, and improving the classification model based on it. Our L-R neural network uses a *back-propagation algorithm* for learning at a rate of 0.01 and with a momentum of 0.99. This neural network has 30 hidden layers and training was performed for 2000 epochs. Our N-E-S-W-O neural network also uses a back-propagation algorithm for learning at a rate of 0.001 and with a momentum of 0.99. This neural network has 100 hidden layers and the training was performed for 1000 epochs. These parameters for our neural network based classifiers were chosen heuristically. Training of these neural networks completes the learning phase.

3.4.2.2 Attack Phase

The attack phase follows a similar procedure as the learning phase, with the exception that the goal here is to recover test words using the trained neural networks-based classification models from the learning phase. In order to do so, the adversary must first create a dictionary of words, and their corresponding word-profiles, that the target is most likely of typing. The dictionary size can vary from a few words to thousands of words, depending on the target and his/her context. If the adversary is unaware of the target's context, he could also create a large dictionary of most popular or all words in the English language. The dictionary creation involves a preprocessing step to obtain equivalent word-profiles of each word in the L/R and X/N/E/S/W/O representation (as discussed before). The attack phase is then executed in sequential steps of: (i) *data collection*, (ii) *feature extraction*, (iii) *keystroke classification*, and (iv) *word matching*.

Data Collection: The same properties and operations from the data collection operation of the learning phase also applies to the data collection during the attack phase. The only exception is that the malicious Android Wear application does not have the ability to clock and tag ground truth characters. As the smartwatch can only detect motion cause by one (watch-wearing) hand, a significant challenge of the proposed inference attack is due to the inability to detect keystrokes made by the non-watch-wearing hand. However, our attack framework requires to know the number of typed characters. To solve this problem, here we assume that the adversary can employ an alternate source or sensor on the smartwatch that can detect keystroke events typed by either hand. The intention for using such an auxiliary sensor is not to classify the keystrokes using it, but to clock the time when a keystroke occurs in the stream of raw linear acceleration data. A microphone can perfectly serve this purpose. Even though a smartwatch's microphone is not effective for recovering keystrokes (due to the aforementioned reasons), it can certainly be used to detect the occurrence of a keystroke itself, made by either hand. This is where the naturally close positioning of the smartwatch near the keyboard is beneficial. Thus, the malicious application uses the microphone to detect keystroke acoustics, and in the case a keystroke event is detected from the acoustic signal, it logs a keystroke event in the linear accelerometer data stream. Space key press events are also clocked or logged because they act as word separators. Fortunately, as we empirically determined, space keys are easy to identify in an audio recording because of the key's distinctive sound and frequency of use.

Feature Extraction: During the attack phase, the same features as in the learning phase are extracted from the raw linear acceleration data recorded by the malicious application. The feature vectors are then used to create two sets of data, one for classifying L vs. R, and one for classifying N vs. E vs. S vs. W vs. O.

Keystroke Classification: The adversary initiates the classification process after extracting all feature vectors. The trained L-R neural network is used to predict the (L/R) label for each individual keystroke. Only when a L _ L key-pair is detected in the data stream by the L-R neural network, the N-E-S-W-O neural network classification is conducted to predict the transition direction label (N/E/S/W/O). Otherwise, the transition direction is labeled as X. Using the predicted labels (and the recognized space keys as described earlier), a word-profile is constructed for each word in the keystroke stream. All the constructed word-profiles are then passed as input to a word matching algorithm described next.

Word Matching: Word matching is the final step of the attack phase, where each predicted word-profile of length m is matched with all words of length $m+1$ in the preprocessed dictionary by the adversary. For each matched word in the dictionary, a *similarity score* is computed based on the number of matching labels between the predicted word-profile and the corresponding word-profile in the dictionary (see details in Algorithm 4). The dictionary word with the highest similarity score is then output as the word corresponding to the predicted word-profile. For some evaluation experiments, we also use a 'similarity list' made of dictionary words with descending order of similarity scores.

Algorithm 4 Word Matching Algorithm

```
similarityScore = 0
for all words of len(m) ∈ dic do
    for pair = 1 to m − 1 do
        for label = 1 to 3 do
            if dic.word.profile[pair][label] = predicted.profile[pair][label] then
                similarityScore++
            end if
        end for
    end for
end for
return similarityScore
```

3.4.3 Experimental Setup

In our experimental evaluation of the proposed inference attack and keystroke characterization framework, we use a setup similar to the one shown in Figure 3.1. We recruit 25 participants[1] who wear the smartwatch on their left wrist and type test words on an external QWERTY keyboard. All data recorded by the smartwatch was transferred to a remote server. Both the training and attack phases are executed on this remote server which is assumed to possess enough computational and storage resources in order to carry out these operations. The specifications of important hardware and software components used in our experiments are outlined below:

1. Smartwatch and sensor hardware: We used the *Samsung Gear Live smartwatch* running *Android Wear build 1.1.1.1944630*. The Gear Live is equipped with an *InvenSense ICS-43430* microphone and an *InvenSense MP-92M* 9-axis Gyro +

[1]Our experiments have been approved by Wichita State University's Institutional Review Board (IRB).

Accelerometer + Compass sensor. The maximum average linear accelerometer sampling rate achieved in our experiments was 50 Hz. Our data collection application can be readily used on any Android Wear smartwatch, which makes the attack framework compatible with a diverse set of smartwatches.

2. Keyboard hardware: We chose to use the *Anker A7726121* bluetooth keyboard because of its generic design. The bluetooth connectivity aided in accurate labeling of sensor data, by allowing us to aggregate recorded sensor data and corresponding typed character on the smartwatch in very close to real time.

3. Signal processing tool: Most of the features are calculated using MatLab 2015a libraries.

4. Supervised machine learning tool: We used *PyBrain v0.31* to train and test the neural networks in our framework. PyBrain is an open-source modular machine learning library for Python, supporting easy integration with underlying environment.

3.5 Evaluation

We first perform two preliminary experiments (involving only one participant) in order to evaluate (i) the base accuracy of the L-R and N-S-E-W-O classifiers by analyzing a set of test sentences from the set of *Harvard sentences* [1] and (ii) the word recovery accuracy of the proposed inference attack strategy by using a dictionary of ten Harvard sentences and attempting to recover each of the same ten sentences as test data. We choose Harvard sentences because they are phonetically-balanced. In

the two preliminary experiments we make the assumption of 'perfect typing', i.e. the participant follows our L/R separation. After the preliminary experiments, we conduct more realistic experiments involving all 25 participants, real-life sentences, larger dictionaries, and without the assumption of perfect typing.

3.5.1 Feature Accuracy

In the first experiment, we examine the base accuracy of both L-R and N-S-E-W-O classifiers in correctly distinguishing between L/R region for individual letters and N/S/E/ W/O transition between pairs of letters. We evaluate our trained classifiers using all the ten sentences in List 6 of Harvard sentences. Interestingly, without any typing errors, the L-R classifier was able to correctly identify 100% of the individual key press events as left or right. However, the N-E-S-W-O classifier had two misclassifications, resulting in 95% accuracy.

3.5.2 Basic Text Recovery

Our next experiment examines the percentage of text (in terms of words) correctly matched by the word matcher. In this preliminary experimental results, we observed that the overall percentage of words correctly matched noticeably dropped due to

Typed Text:	**The**	**show**	**was**	**a**	**flop**	**from**	**the**	**very**	**start.**
Recovered:	***	**sums**	***	*	**flop**	**from**	***	**very**	**start.**

Colliding Word-Profiles:
Show: LXR . RXR . RXL , Sums: LXR . RXR . RXL

Figure 3.5: Sentence 4 from List 6 of Harvard Sentences. The words 'show' and 'sums' have the same word-profile resulting in a collision in the dictionary.

mismatched two and three letter words in the analyzed text. The smaller number of features in these words results in several of these 'small' words having the same word-profile, thus causing more collisions during matching. We also observed that most of these words are generally articles and conjunctions (e.g. an, the, and, or), which can be easily interpolated by analyzing the language semantics of the recovered text. As a result, we opted to consider only 'long' words of four letters or more in all final percentages of recovered words in our remaining experiments. The 'short' words are instead denoted with asterisks ("*") in the recovered sentences.

In this experiment, we used the same ten sentences in List 6 of Harvard sentences, as from the first experiment. Among the 48 words of length four or more, only three were erroneous (93.75% successful recovery). Out of the three, two had incorrect N/E/S/W/O classification, while the other was due to collisions in the word-profiles. Figure 3.5 shows the sentence where the collision occurred. The problem of collision will increase with increasing size of the dictionary. However, if we also take in to account second and third ranked similar word-profiles during word matching, this problem can be moderated. Errors in word recovery (especially, due to collisions) can be further diminished by analyzing language semantics, and then selecting the word (from the multiple colliding words in the dictionary) that is semantically best fit.

3.5.3 Contextual Dictionary

This experiment evaluates how our attack performs when the adversary has some knowledge about what their targets are typing. All the 25 participants typed a paragraph of 40 words (of length four or more) that appear in a *National Public Radio*

(NPR) news article on Greece debt crisis, and this experiment simulates eavesdropping on a reporter typing the NPR news article. The dictionary is formed with words that appear in six other news articles related to Greece debt crisis, that were published a week before the target article. The dictionary is also sorted based on frequency of word appearances in the six chosen news articles, which improves our chances of successfully solving a word-profile collision. Figure 3.6 shows the percentage of words recovered per participant. As one should expect in a real-life attack, out of the 40 words in the target paragraph, only 27 were present in the contextual dictionary. Even so, our framework was able to recover as many as 21 words (for 3 participants), by matching with just the first ranked word in the sorted list of similarity scores (Figure 3.6). In other words, 21 words were uniquely identified without any ambiguity, for the 3 participants. On the lower end, only 4 words were recovered for 3 participants, but the recovery can be improved by considering words with lower rank in the sorted similarity list. The mean word recovery using only the first ranked word was 31.2% (or 46.2% if we consider only the words present in the dictionary).

3.5.4 Typing Behavior and Speed

During data collection, we observed that in many instances participants did not follow our assumed layout. Some of the participants frequently used their left hand to press a key on the right side of the keyboard, and vice versa. Upon further investigation we also found that participant who typed slower, were less likely to follow the left and right division of the keyboard. This phenomenon explains why participants who took longer to type all the 40 words saw lesser word recovery rate in Section 3.5.3 experiments. Figure 3.6 shows the time taken by the adversary (whose

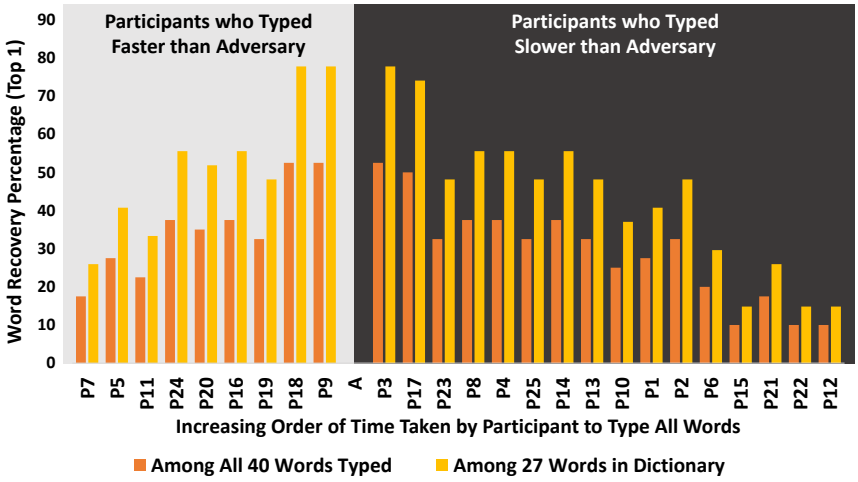


Figure 3.6: Contextual Dictionary: Percentage of words recovered per participant, presented in descending order of typing speed of the participants.

typing was used as the training data) to type the 40 words as A on the horizontal axis. Participants on the right of A typed slower than the adversary, and we can see a trend that the recovery rate drops with slower typing. Interestingly, we see a similar trend on the left of A as well, indicating that recovery rate drops with faster typing. Our speculation is that due to fast typing there may occur overlapping feature regions leading to poorly performing L/R classification, and incorrect L/R classification can significantly affect recovery of words. Combining both the trends we arrive at a conclusion that participants who typed at a similar speed as the adversary were more vulnerable to the attack. For an adversary, the take-home message from this conclusion is that the attack framework can be optimized by training it with a typing speed and style expected from the potential victim(s).

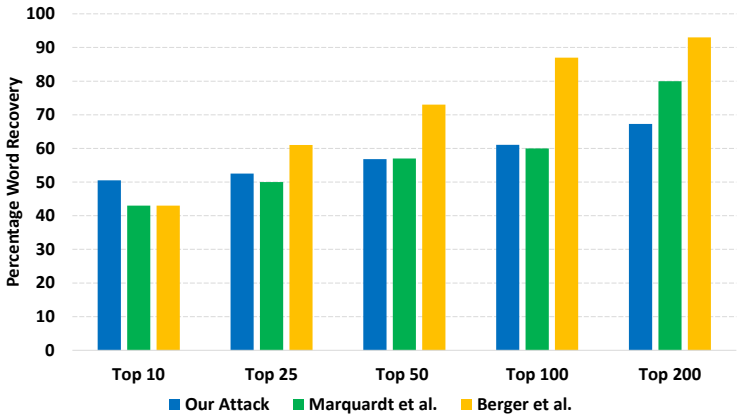


Figure 3.7: A comparison of accuracy of our attack with Marquardt et al. [77] and Berger et al. [16]. Note that in spite of not having wrist movement information available from the non-watch-wearing hand, our results are roughly comparable for a very large (60,000 words) dictionary.

3.5.5 Comparison to Previous Work

From the above experiments, we saw that relying on exact match with first ranked words may not always result in the best inference accuracy. As pointed out by earlier emanation based keystroke inference attacks [77, 16], more intelligent adversaries may be able to form target sentences with lower ranked words from the sorted similarity list. So, we re-create the experiments conducted by Marquardt et al. [77] and Berger et al. [16] in order to be able to compare our attack framework directly with theirs. We use a similar sized English dictionary of 60,000 words (of length 4 or more), sorted based on frequency of usage in English literature. We reuse 38 of the 40 words typed by participants in Section 3.5.3 experiment, while remaining 2 (first and last name of former Greek finance minister) are not contained in the 60,000 word dictionary. Figure 3.7 shows the comparison.

Our attack framework demonstrates comparable accuracies to that of Marquardt et al. and Berger et al. It was able to correctly map test words to the top 10 words in the sorted similarity list 50.5% of the time, which happen to be significantly higher than the earlier works using smartphone sensors. The word recovery steadily improves as we increase the size of selection from the sorted similarity list, but our attack trails behind the other two in case of very large selections. Note that the complexity of forming sentences with ambiguously recovered words grow exponentially with the selection size. Therefore, achieving a better recovery rate with just top 10 words is more significant than having a better recovery rate using top 500 words. It is also important to remember the distinct challenge faced by our technique where no wrist movement information is available from the non-watch-wearing hand.

We are unable to compare equitably with Wang et al. [115] because of their different experimental setup. However, using a smaller dictionary of only 5,000 words, they were able to narrow down a typed word to 24 possibilities with a 50% chance. In contrast, we use a much larger dictionary of 60,000 words, and our attack is still able to narrow down a typed word to only 25 possibilities with about 52.5% chance.

3.6 Limitations

Our proposed movement-based keystroke inference attack using smartwatches circumvents some of the limitations of emanation-based attacks, but it faces new challenges. In this section, we discuss some of them.

- **Ambient Wrist Movement:** In case the target participates in some other activity (for example, having a periodic sip of drink) in between typing, the

introduced noise can lead to incorrectly predicted words. However, since each word is treated separately, the error will not propagate.

- **Left and Right Handedness:** Although the same attack framework is applicable independent of the hand on which the smartwatch is worn, classifiers trained using data with the smartwatch worn on the left hand cannot be used to predict words typed while wearing the smartwatch on the other hand, and vice-versa.

- **Inferring Non-Dictionary Text:** Our attack performs well for dictionary words, but is incapable of recovering numeric keys and special characters. As a result, if the adversary is interested in learning data with numbers and/or special characters (such a credit card numbers, strong passwords, etc.), the presented framework and attack will not be directly applicable. However, wrist movements can still be useful in determining approximate position of keys pressed, which may significantly reduce the search space.

3.7 Conclusion

This chapter presents a novel keystroke inference attack which utilizes wrist-motion data gathered from a smartwatch as side-channel information. Wrist movements yield very different data compared to popular emanation-based keystroke inference attacks. In order to harvest the information masked in wrist movements for inferring keystrokes, we designed and validated a novel learning-based attack framework which is specifically targeted towards recovering text typed by a smartwatch

wearing user on an external QWERTY keyboard. By showing the feasibility of the proposed classification and prediction mechanisms, we validate our hypothesis that wearable devices such as smartwatches can leak sensitive personal information if access to sensors (on these devices) is not appropriately regulated.

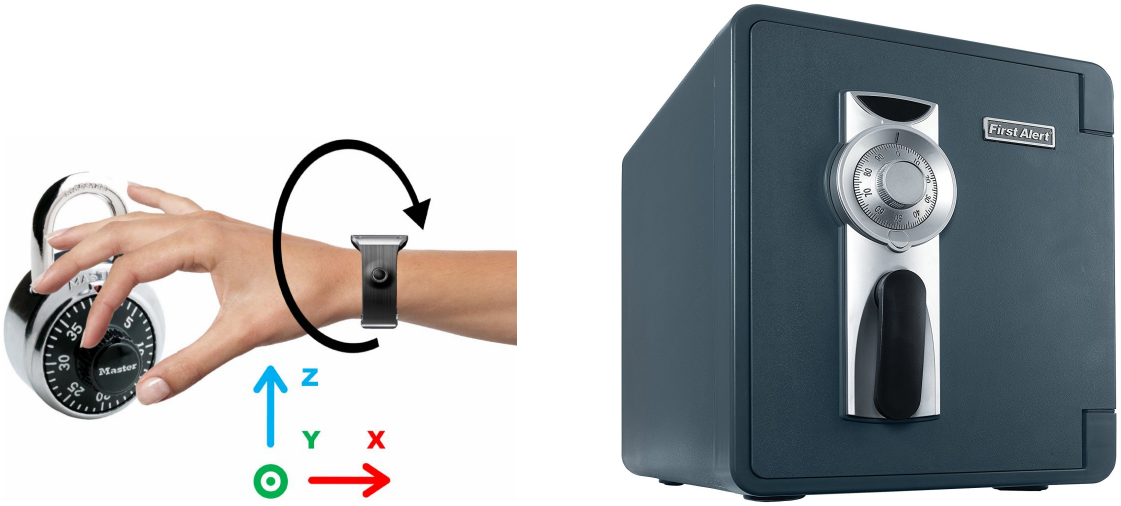
Parts of this chapter appeared in [71].

CHAPTER 4
ATTACKS ON PHYSICAL INTERACTIONS: COMBINATION PADLOCKS AND SAFES

4.1 Introduction

In the last two chapters, we have validated threats that enable an adversary to infer private inputs or interactions made by a target user on an input-interface (of some system of interest to the adversary) by taking advantage of unregulated sensor data available from the user's wrist-wearable. A majority of similar research contributions in this direction also have primarily focused on threats that attempt to infer private user inputs on interfaces of purely cyber or cyber-physical systems, for example, inference of keystrokes or taps on physical keyboards or touchscreen-based keypads [115, 70, 114, 46]. Outcomes of such threats, if successful, can significantly impact the cyber-security and cyber-presence of targeted users.

In this chapter, our focus is on a slightly different kind of threat, which is to investigate the feasibility of inferring a target user's private inputs or interactions on the interface of a purely mechanical device by harnessing the sensor data available from the user's wrist-wearable. We specifically focus on inferring inputs on mechanical devices typically used to secure physical access (on doors and lockers), for example, combination locks. Such privacy threats concerning mechanical safety devices, which may now be feasible due to the upcoming wearable device technology, has the potential of impacting the physical safety and security of users. Despite their severity, such threats have not received much attention in the literature.



(a) Master Lock 1500T (b) First Alert 2087F-BD Safe.

Figure 4.1: Target mechanical combination locks.

Our primary research goal in this chapter is to investigate the feasibility of inferring unlock combinations of commercially-available mechanical combination locks and safes (as shown in Figure 4.1) by exploiting inertial or motion sensor data from wrist-wearables such as smartwatches. To unlock such combination locks, an authorized user typically enters the secret unlocking combination or key as a sequence of counter-clockwise and clockwise rotations of the lock's circular numeric dial. Now during the unlocking process, the wrist on the unlocking hand undergoes perceptible and unique movements and rotations of its own, which is strongly correlated with the unlock combination. Our hypothesis is that, if these motions can be accurately captured and characterized, say, by targeting the gyroscope sensor on-board the unlocking hand's wrist-wearable, then it can be used to infer the lock's combination.

Our objective is to validate the above hypothesis by empirically evaluating the accuracy and effort with which such an inference attack can be executed using modern wrist-wearables. In line with this objective, we make the following technical contributions in this chapter:

1. We present a novel motion-based combination or key inference framework comprising of: (i) an activity recognition component for efficiently and accurately identifying unlocking-related data in the continuous motion data stream, (ii) a segmentation component to separate and appropriately characterize motion data corresponding to each part of the multi-part combination or key, and (iii) an attack component that maps the characterizations of the individual parts obtained from the previous steps to a (or a set of) valid combination(s) or key(s).

2. A comprehensive empirical evaluation of the proposed attack framework in order to assess its performance on: (i) a commercially available padlock and safe, (ii) using different key spaces, (iii) in a cross-device setting, (iv) in a cross-hand setting, and (iv) under real-life lock operation scenarios.

4.2 Related Work

Threats that attempt to infer private information, user-contexts or user-activities by capturing related electromagnetic, acoustic, optical and/or mechanical emanations from a target device or user and employing them as information side-channels have been well-studied in the literature [89, 60, 5, 8, 16, 12, 10, 113, 11, 42, 6, 115, 67].

With the advent of smartphones, researchers started focusing on employing the phone's on-board hardware and software sensors to investigate the feasibility of similar inference attacks [110]. One notable sensor modality that now became available as an attack vector is the smartphone's inertial or motion sensors, such as, *accelerometers* and *gyroscopes*, which are capable of capturing fine-grained linear and angular motion of the user or object on which the phone was placed. Smartphone inertial sensors have been exploited to infer keystrokes on the phone itself as well as external keyboards [14, 77, 20, 118, 86], to track user movements and locations [40, 44, 83], to infer private user activities [85] and to decode human speech [79]. Similarly, smartphone microphone and/or magnetometer have also been exploited to infer private user information [94] or trade secrets (such as 3D-printer designs) [45, 102, 32], private user activities [91] and natural handwriting [121]. Recently, aggregate power usage over a period of time available from the smartphone's power meter was used to track user movements and locations [80].

The arrival of smartwatches and fitness bands have fueled a similar line of research in the area of private user-input, activity and context inference threats that take advantage of data available from sensors on-board these commercial wrist-wearable devices. However, unlike smartphones, as smart wearables are always carried by users on their body in the same natural position, the resulting continuous nature of sensor data available from them is more vulnerable to misuse and related inference threats more likely to succeed. Smartwatch motion sensors, similar to the smartphone case, have been exploited to infer keystrokes [74, 115, 70, 71, 114], user-activities [96, 69], handwriting [117, 116] and driving behavior [57]. Recently, ambient light sensors on

these devices have also been used to infer private keystroke information [46]. Given this plethora of research results, it is clear that sensors on-board mobile and wearable devices pose a significant privacy threat. It is alarming though that common mobile and wearable device users are unaware of such threats [26].

In this chapter, we investigate the feasibility of a new kind of privacy threat, i.e., inferring unlock combinations of mechanical locks using wrist-wearable motion sensors, which has never been investigated before. Several modern smart locks offer a numeric keypad which can be compromised using known smartwatch-based keystroke inference techniques in the literature [74, 115, 70, 71, 114]. However, in this chapter we target traditional rotation-based mechanical locks which are still very popular and where existing attack techniques will not work. Blaze [18, 17] systematically examined physical and design weaknesses in both combination and pin-tumbler locks. However, our primary contribution in this work is to show how external side-channel attacks can make even a securely designed lock vulnerable.

4.3 Adversary Model

We consider the scenario of a target user who is wearing a wrist-wearable such as a smartwatch and is entering the *unlock combination* or *key* on the circular dial of a mechanical combination lock (targeted by the adversary) with the watch-wearing hand. The goal of the adversary is to infer the unlocking combination of the lock by employing the inertial or motion sensor data available from the smartwatch worn by the target user. We assume that the adversary has knowledge of the exact type (make and model) of the target combination lock and that the dial of the lock has

sufficient resistance to prevent rotation by mere movement of fingers. The adversary is able to record and obtain the inertial or motion sensor data from the target smartwatch through several different modalities. One way an adversary can achieve this is by creating a trojan app and then tricking the unsuspecting target user or victim into downloading and installing this trojan onto their wearable device. In case the adversary is a popular service provider, gaining access in such a fashion is much more straightforward as unsuspecting users may download and install the malicious app on their own volition. This malicious eavesdropping app samples the on-device sensors of interest (specifically, the *gyroscope* sensor data is used for this particular attack) and transfers the sampled sensor data to a remote server controlled by the adversary through some covert communication channel, say by hiding it within useful communications. We assume that the malicious app has the required permissions to access these sensors of interest. As the proposed attack employs the gyroscope sensor, which is a *zero-permission* sensor on popular wearable operating systems such as Android Wear and watchOS, the adversary has a relatively unobstructed attack path once the malicious app is installed on the device. We also assume that the adversary maintains a remote server with sufficient storage and computational resources to archive the eavesdropped data and to perform *offline inference computations*. The above adversary model is practically feasible and has been a standard assumption for similar lines of investigations. In addition to the above cyber resources, the adversary also has a limited amount of *physical access* to the target lock (in order to conduct the actual physical attack on the lock by trying out the inferred combination), but not long enough to manually brute-force the lock's combination.

The adversary presets/notes the position of the (lock's) dial before the target user begins the unlocking operation. However, the adversary has no visual access to the dial during the unlocking operation itself.

4.4 Background

4.4.1 Mechanical Combination Locks

After studying the technical specifications of several commercially available mechanical combination locks, we decided to focus on two specific types of locks whose internal mechanical structure and physical operation are representative and commonly found in most rotary combination locks: (i) *padlocks*, and (ii) *consumer-grade safes*. For the padlock we chose a *Master Lock 1500T* model lock (Figure 4.1(a)), while for the safe we chose a *First Alert 2087F-BD* safe (Figure 4.1(b)).

The front dial of the Master Lock 1500T is used to enter the unlock combination key and has 40 numbers on its face. As the combination key comprises of three numbers (each taking a value between 0 and 39) which must be entered sequentially, the resulting theoretical combination key space is $40^3 = 64,000$. In order to unlock the Master Lock 1500T, a user must turn the dial clockwise two full rotations and stop at the first number of the combination key on the third turn (*phase 1*), then turn it counter-clockwise past the first number of the combination key to the second number of the key (*phase 2*), and finally turn the dial clockwise to the third number of the combination key (*phase 3*). Let traversing from one number to it's sequential number (in any direction) be called a "*unit*" of traversal. Then it should be noted that, depending on the combination key being entered, in phase 1 the user traverses

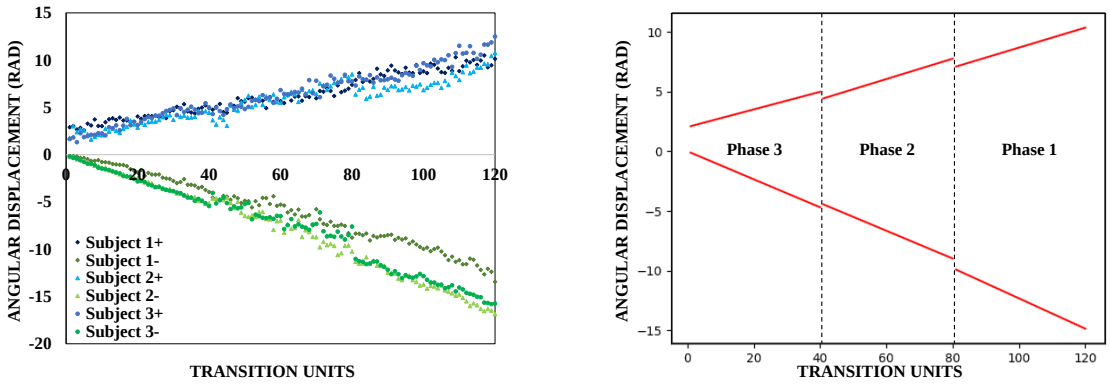
anywhere between 81 and 120 units in the clockwise direction, in phase 2 he traverses anywhere between 41 and 80 units in the counter-clockwise direction, and in phase 3 he traverses anywhere between 1 and 40 units in the clockwise direction. If this procedure is correctly followed, and if the entered combination key is correct, the indentations on the lock's cams align correctly allowing the hasp to be released and opening the lock.

The First Alert 2087F-BD safe's lock dial comprises of 100 numbers (from 0 to 99) on its face. It's combination key comprises of four numbers (each taking a value between 0 and 99) which must be entered sequentially, thus resulting in a theoretical combination key space of 100^4. In order to unlock the safe, a user must turn the dial counter-clockwise four full rotations and stop at the first number of the combination key on the fifth turn (*phase 1*), then turn it clockwise twice past the first number to the second number (*phase 2*), then turn it counter-clockwise past the second number to stop at the third number (*phase 3*), and finally turn the dial clockwise to the fourth number (*phase 4*). Depending on the combination key being entered, in phase 1 the user traverses anywhere between 401 and 500 units in the counter-clockwise direction, in phase 2 he traverses anywhere between 201 and 300 units in the clockwise direction, in phase 3 he traverses anywhere between 101 and 200 units in the counter-clockwise direction, and in phase 4 he traverses anywhere between 1 and 100 units in the clockwise direction. Similar to the Master Lock 1500T, if this procedure is correctly followed and if the entered combination key is correct, the safe opens.

4.4.2 Combination Key and Wrist Movements

Before designing an inference framework, we need to develop a clear understanding of how the activity of entering a combination key on a lock's dial impacts the wrist movement of the unlocking hand, and if it is possible to accurately and consistently characterize this movement using the motion sensor data obtained from modern wrist wearables such as smartwatches. More concretely, we would like to first understand the *relationship between* the *amount of movement of a lock's dial* and the corresponding *amount of movement of the user's wrist*. We quantify the amount of movement of a lock's dial using the parameter *transition*, which measures the *number of units traversed when inputing a particular number of the combination*. As the unlock combination key of the Master Lock 1500T padlock has three numbers (and correspondingly, the unlocking procedure has three phases), the amount of movement of the lock's dial during the unlocking process can be completely characterized by three transitions. Similarly, as the First Alert 2087F-BD safe has a four number combination, the amount of movement of the lock's dial during unlocking can be completely characterized by four transitions. We quantify the amount of movement (or rotation) of a user's wrist by computing the *angular displacement* from the observed *smartwatch gyroscope* data. As the gyroscope measures angular velocity, the corresponding angular displacements can be calculated by integrating the obtained angular velocity readings.

In order to quantify the relationship between transitions on a lock's dial and the wrist's angular displacements, we conduct some preliminary unlocking experiments on the Master Lock 1500T padlock. Specifically, we collected smartwatch gyroscope



(a) (b)

Figure 4.2: (a) – Positive (blues) and negative (greens) angular displacements, collected from three subjects; (b) Combined linear least squares fitting.

samples at a sampling rate of 200 Hz from three human subjects who unlocked the padlock wearing a Samsung Gear Live. The subjects in our preliminary experiments entered 40 different combinations on the Master Lock 1500T padlock which covered all the 120 possible transitions (40 possible transitions per number in any combination key). While entering each combination, the subjects always started from a known position (number 0)[1] and entered the combination by correctly following the unlocking procedure described in Section 4.4.1. For each subject, we plot the angular displacement (in radians), calculated by integrating the corresponding angular velocities observed on the x-axis of the smartwatch's gyroscope, for each each transition in either direction (Figure 4.2(a)).

[1]The starting point can be any number on the dial. However, the key inference function (Equations 4.1 and 4.2) must be initialized accordingly, during the inference phase.

From Figure 4.2 we first observe that, for each transition (irrespective of the direction of rotation), the angular displacement of the wrist calculated from the raw smartwatch gyroscope data is not the same as the angular displacement of the lock's dial. These inaccuracies could be attributed to the discrete nature of the gyroscope readings, which are limited by the maximum sampling rate of the gyroscope hardware. In addition to this, the cartilaginous joints between the fingers and the wrist, do not allow for a *perfect rotation* of the wrist during the unlocking operation. As a result, an adversary cannot simply use the angular displacement of the wrist calculated from the raw smartwatch gyroscope data to determine the angular displacement, and thus the corresponding transition, on the lock's dial. Our second observation is that the angular displacement of the wrist calculated from the raw smartwatch gyroscope data can be approximated as an *increasing linear function* of the transitions on the lock's dial. Although intuitive, the interesting and encouraging aspect here is that this relationship is consistent for all three subjects. Lastly, we observe that this linear relationship is reasonably *homologous* or similar across different subjects. We only used the x-axis of the gyroscope data for these plots because we observed that the x-axis remains perpendicular to the lock (Figure 4.1(a)) during the unlocking operation and provides a more accurate measure of angular displacement than the other two axes.

So, what do these observations mean to an adversary who wants to infer the combination key entered by some target user? The adversary is *unable* to accurately determine the angular displacement or transition on the lock's dial (and thus the corresponding number in the combination) directly from the corresponding angular

displacement of the wrist computed using the smartwatch's gyroscope data. However, an adversary could use the above observations to construct a *learning-based inference framework* that translates angular displacements of the wrist (computed from the smartwatch's gyroscope data) to transitions on the lock's dial, and train this framework using some representative training data. The adversary could then employ such a trained inference framework to infer the combination (entered by the target user) from the smartwatch gyroscope data. We develop such an inference framework in Sections 4.5 and 4.6. However, there are two additional challenges that we need to overcome. First, in a long sequence of time-series gyroscope data, how does the adversary *identify* data corresponding to the unlocking motion? Second, to accurately compute the angular displacement of the wrist for each phase of the unlocking procedure, the adversary needs to *divide* or *segment* the gyroscope time-series into individual phases. We address these issues by developing an unlocking activity recognition technique (Section 4.4.3), and a segmentation technique (Section 4.4.4).

4.4.3 Unlocking Activity Recognition

Before attempting to infer combinations from the target user's wrist motions, one critical challenge for the adversary is to precisely detect when the unlock event takes place. In order to overcome this challenge, we design an *offline activity recognition technique* to detect and record timestamps of unlocking operations on combination locks. Our activity recognition technique does not require any additional adversarial capabilities or resources as it employs only the gyroscope data stream (specifically, the x-axis data) which is already recorded by the adversary for the inference task.

While analyzing characteristics of the time-series gyroscope data during unlocking, we observed that the integrated angular displacement increases on both positive and negative axis in successive periods. This is because after rotating the dial (clockwise or counter-clockwise) to an extent, users release the dial, go back in reverse (counter-clockwise or clockwise, respectively), again grab the dial, and continue entering the remaining part of the combination key (clockwise or counter-clockwise, respectively). We refer to one such clockwise-counterclockwise (or vice-versa) motion during combination key entry as a *"spin"*, which is primarily related to the comfortable wrist rotation ability (or desire) of humans. Such *spin*-ing is repeated multiple times during any combination key entry, approximately every half a turn (π) and over a maximum duration of approximately 5 seconds. We can observe this phenomenon in the sample gyroscope (x-axis) time-series corresponding to a padlock unlocking operation (Figure 4.3). We utilize the above observations in the design of the following four features which will be employed by our activity recognition technique:

- **Positive Displacements ($^{+}\alpha$):** Integration of positive x-axis gyroscope samples.
- **Negative Displacements ($^{-}\alpha$):** Integration of negative x-axis gyroscope samples.
- **Summed Displacement ($^{+}\alpha + {}^{-}\alpha$):** Sum of integrated positive and negative x-axis samples.
- **Total Displacement Magnitude ($^{+}\alpha + |{}^{-}\alpha|$):** Sum of the magnitudes of integrated positive and negative x-axis samples.

In order to confirm the above observations, we computed the means and standard deviations of the above four features over all the 5 second windows (maximum duration of a *spin*) in the preliminary unlocking-related gyroscope data collected earlier (Figure 4.2(a)). We observed that the mean values of the magnitudes of ${}^{+}\alpha$ and ${}^{-}\alpha$ are approximately similar in a *spin*, the mean value of the total displacement magnitude is approximately double of both ${}^{+}\alpha$ and ${}^{-}\alpha$, and the mean value of the summed displacement is close to zero. We employ these learned mean and standard deviation values to form a *decision-tree* for detecting *spins*. During the activity recognition, the above four features are recursively computed for every 5 second window, and the decision-tree classifies a window as a *spin* if all the four features are within one standard deviation of the learned means. In the case of padlock, if 5 (minimum number of *spins* observed for the shortest padlock combination: 39-0-39) or more spins are observed within a short time window (empirically determined based on the maximum unlocking time observed in data) an unlocking activity is recognized. A similar strategy could be used to recognize unlocking operation on a safe.

4.4.4 Segmentation

Segmentation of the time-series gyroscope data representing the entire combination key input into data corresponding to individual phases or transitions (three for the padlock and four for the safe) will simplify the overall design of the inference framework. This is because the combination inference problem can then be reduced to the problem of independently inferring the combination number corresponding to each segmented transition. In order to design a reliable segmentation technique, we leverage on the observation from our earlier experiments that *humans tend to slow*

down when they approach a number in their combination key. We believe that this phenomenon is due to the cognitive processing of the human brain governing the physiological action of stopping at a particular number, which causes the subjects to slow down when approaching the intended number in their combination key or risk overshooting it (and thus having to restart the entire key entry process). We can observe this phenomenon in the time-series gyroscope data corresponding to the unlocking operation of the Master Lock 1500T padlock by one of the subjects, where we can clearly see (Figure 4.3) the sharp decreases in the angular velocity (red line) when approaching the combination key number near the end of each phase. In order to *automate* the process of segmentation, we design an algorithm to detect the relative decrease in angular velocity, and use the peaks (representing slowest movement) to segment the entire time-series. The algorithm first computes the absolute values of all samples in the gyroscope time-series data, inverts, and then amplifies the time-series by a factor of 10 (for better visualization). Then, on the resultant time-series, a Gaussian filter with a moving window [27] of 15 samples (learned empirically, at 200 Hz sampling frequency) is applied. Finally, the algorithm performs a search for top-2 global peaks in the resultant time-series, which represents approximate timestamps for the first and second number of the combination key, in chronological order. The blue (top) line in Figure 4.3 is an example of the visualized output of our segmentation algorithm, showing the detected peaks and resulting segmentation timestamps. Our algorithm also works on gyroscope data from the safe, using top-3 peaks.

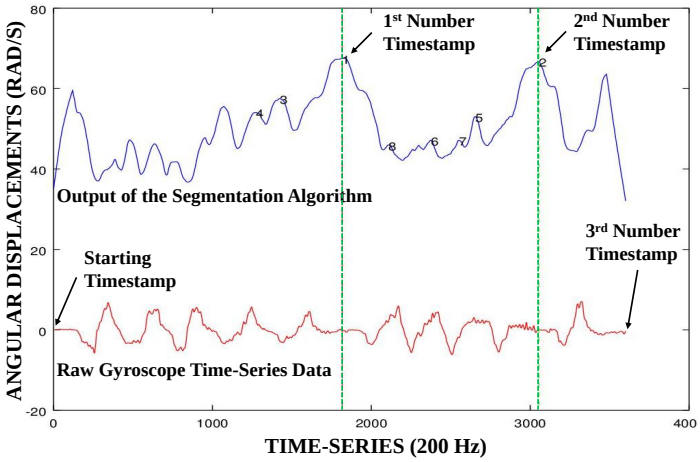


Figure 4.3: Segmentation using a Gaussian filter.

4.5 Deterministic Attack Framework

We develop two learning-based inference frameworks to infer numbers of the combination key inputted on the lock's dial from the segmented smartwatch gyroscope data. We first outline a *deterministic framework* which outputs a single inferred combination key from the segmented time-series gyroscope input.

4.5.1 Padlock Attack Model

Assuming that the starting point s on the padlock's dial is fixed/known (say, to be 0), we can define $\Phi_1 = \{81, 82.., 120\}$, $\Phi_2 = \{41, 42.., 80\}$ and $\Phi_3 = \{1, 2.., 40\}$ as the sets of *possible padlock transitions* in phase 1, phase 2 and phase 3 of the unlocking procedure, respectively. Now for a given 3-number combination key $k = \langle a, b, c \rangle$ of the Master Lock 1500T padlock, where $a, b, c \in \{0, 1.., 39\}$, let $\theta_k^{sa} \in \Phi_1$, $\theta_k^{ab} \in \Phi_2$ and $\theta_k^{bc} \in \Phi_3$ be the *actual transitions* or number of units traversed (on the lock's dial) between consecutive numbers of the combination key

k, i.e., θ_k^{sa}, θ_k^{ab} and θ_k^{bc} are the number of units traversed between 0 and a, between a and b, and between b and c, respectively. Let α_k^{sa}, α_k^{ab} and α_k^{bc} denote the corresponding angular displacements of the target user's wrist (ignoring the direction or sign) calculated from the segmented smartwatch gyroscope data. The inference framework comprises of a *training phase* and an *attack phase*. During the training phase, the adversary collects training data (from a set of human participants) comprising of a set of θ and corresponding α values for a sample set of combinations covering all possible transitions. As indicated by our preliminary results (Figure 4.2(a)), the relationship between angular displacements of the wrist and transitions on the lock's dial can be approximated by a linear function. Thus, the adversary can use the training data to learn such a linear function $\alpha = m\theta + n$ that best fits all (θ, α) points in each of the $[s, a]$, $[a, b]$ and $[b, c]$ transition ranges of the training data. The adversary can employ a *least squares* [120] technique in order to learn such a linear function (Figure 4.2(b)). Then during the attack phase, for an unknown combination key $\hat{k} = \langle \hat{a}, \hat{b}, \hat{c} \rangle$, the adversary first segments the gyroscope data and computes the corresponding angular displacements $\alpha_{\hat{k}}^{s\hat{a}}$, $\alpha_{\hat{k}}^{\hat{a}\hat{b}}$ and $\alpha_{\hat{k}}^{\hat{b}\hat{c}}$. The adversary's goal then is to determine a combination k', as an inference of $\hat{k}$, by first *approximating* or *estimating* the $\theta_{\hat{k}}^{s\hat{a}} \in \Phi_1$, $\theta_{\hat{k}}^{\hat{a}\hat{b}} \in \Phi_2$ and $\theta_{\hat{k}}^{\hat{b}\hat{c}} \in \Phi_3$ values from the corresponding angular displacements ($\alpha_{\hat{k}}^{s\hat{a}}$, $\alpha_{\hat{k}}^{\hat{a}\hat{b}}$ and $\alpha_{\hat{k}}^{\hat{b}\hat{c}}$, respectively). Let these approximations of $\theta_{\hat{k}}^{s\hat{a}}$, $\theta_{\hat{k}}^{\hat{a}\hat{b}}$ and $\theta_{\hat{k}}^{\hat{b}\hat{c}}$ be denoted as $\bar{\theta}^{s\hat{a}}$, $\bar{\theta}^{\hat{a}\hat{b}}$ and $\bar{\theta}^{\hat{b}\hat{c}}$, respectively. In order to accomplish this, the adversary employs the linear function ($\alpha = m\theta + n$) learned earlier. Once the transition in each phase has been estimated, k' can be computed as:

$$
\begin{aligned}
k' = \langle((-\bar{\theta}^{s\hat{a}} + s) \mod 40), \\
((\bar{\theta}^{\hat{a}\hat{b}} + (-\bar{\theta}^{s\hat{a}} + s)) \mod 40), \\
((-\bar{\theta}^{\hat{b}\hat{c}} + (\bar{\theta}^{\hat{a}\hat{b}} + (-\bar{\theta}^{s\hat{a}} + s))) \mod 40)\rangle
\end{aligned}
\tag{4.1}
$$

4.5.2 Safe Attack Model

Similar to the padlock, we can define $\Psi_1 = \{401, 402.., 500\}$, $\Psi_2 = \{201, 202.., 300\}$, $\Psi_3 = \{101, 102.., 200\}$ and $\Psi_4 = \{1, 2.., 100\}$ as the sets of *possible safe transitions* in phase 1, phase 2, phase 3 and phase 4 of the safe unlocking procedure, respectively. For a given 4-number safe combination $k = \langle a, b, c, d\rangle$, where $a, b, c, d \in \{0, 1.., 99\}$, let $\theta_k^{sa} \in \Psi_1$, $\theta_k^{ab} \in \Psi_2$, $\theta_k^{bc} \in \Psi_3$ and $\theta_k^{cd} \in \Psi_4$ be the *actual transitions* between consecutive numbers of the combination key k. Also, let α_k^{sa}, α_k^{ab}, α_k^{bc} and α_k^{cd} denote the corresponding angular displacements of the target user's wrist (ignoring the direction) calculated from the segmented smartwatch gyroscope data. Similar to the padlock case, the adversary collects training data (from a set of human participants) comprising of a set of θ and corresponding α values for a sample set of combinations covering all possible transitions, and uses it to learn a linear function (of the form of $\alpha = p\theta + q$) by employing a least squares [120] technique. Then during the attack phase, for an unknown combination key $\hat{k} = \langle\hat{a}, \hat{b}, \hat{c}, \hat{d}\rangle$, the adversary first segments the time-series gyroscope data and computes the corresponding angular displacements $\alpha_{\hat{k}}^{s\hat{a}}$, $\alpha_{\hat{k}}^{\hat{a}\hat{b}}$, $\alpha_{\hat{k}}^{\hat{b}\hat{c}}$ and $\alpha_{\hat{k}}^{\hat{c}\hat{d}}$. The adversary's goal then is to determine a combination k' as an inference of $\hat{k}$ by first estimating the $\theta_{\hat{k}}^{s\hat{a}} \in \Psi_1$, $\theta_{\hat{k}}^{\hat{a}\hat{b}} \in \Psi_2$, $\theta_{\hat{k}}^{\hat{b}\hat{c}} \in \Psi_3$ and $\theta_{\hat{k}}^{\hat{c}\hat{d}} \in \Psi_4$ values from the corresponding angular displacements. In

order to accomplish this, the adversary employs the linear function ($\alpha = p\theta + q$) learned earlier. Then the adversary computes k' as:

$$\begin{aligned} k' = \langle((\bar{\theta}^{s\hat{a}} + s) \mod 100), \\ ((-\bar{\theta}^{\hat{a}\hat{b}} + (\bar{\theta}^{s\hat{a}} + s)) \mod 100), \\ ((\bar{\theta}^{\hat{b}\hat{c}} + (-\bar{\theta}^{\hat{a}\hat{b}} + (\bar{\theta}^{s\hat{a}} + s))) \mod 100), \\ ((-\bar{\theta}^{\hat{c}\hat{d}} + (\bar{\theta}^{\hat{b}\hat{c}} + (-\bar{\theta}^{\hat{a}\hat{b}} + (\bar{\theta}^{s\hat{a}} + s)))) \mod 100)\rangle \end{aligned} \tag{4.2}$$

4.6 Probabilistic Attack Framework

One shortcoming of the deterministic framework is that it outputs only a *single prediction*, which if incorrect, is not very useful to the adversary. A *ranked list* of predictions ("*close*" to the actual combination) would be useful in reducing the search space and more desirable, especially if the combination predicted by the deterministic framework is incorrect. Empirical analysis of our deterministic framework (Section 4.7.2) shows that the inference error (for each inferred number in the combination) has a low standard deviation, which suggests that numbers neighboring an incorrect inference have a higher likelihood of being part of the real combination key than numbers farther away. We use this observation in the design of our probabilistic framework.

4.6.1 Ranking of Padlock Key Predictions

The goal of the probabilistic framework is to create an ordered list of inferred combinations, ranked based on the probability of a combination being the actual combination. We achieve this objective by giving priority to transitions closer to $\bar{\theta}^{s\hat{a}}$,

$\bar{\theta}^{\hat{a}\hat{b}}$ and $\bar{\theta}^{\hat{b}\hat{c}}$ (calculated by the deterministic model), than transitions further away from it. This is done by assigning probabilities to all possible transitions in Φ_1, Φ_2 and Φ_3 using three *normal distributions* $\mathcal{N}(\bar{\theta}^{s\hat{a}}, \sigma^2_{s\hat{a}})$, $\mathcal{N}(\bar{\theta}^{\hat{a}\hat{b}}, \sigma^2_{\hat{a}\hat{b}})$ and $\mathcal{N}(\bar{\theta}^{\hat{b}\hat{c}}, \sigma^2_{\hat{b}\hat{c}})$, respectively. The means and standard deviations of these distributions are learned from the deterministic model presented in Section 4.5.1. Specifically, we calculate probabilities $P(X|\alpha^{s\hat{a}}_{\hat{k}}) \sim \mathcal{N}(\bar{\theta}^{s\hat{a}}, \sigma^2_{s\hat{a}})$ for all possible transitions $X \in \Phi_1$ being the actual transition performed in phase 1, $P(Y|\alpha^{\hat{a}\hat{b}}_{\hat{k}}) \sim \mathcal{N}(\bar{\theta}^{\hat{a}\hat{b}}, \sigma^2_{\hat{a}\hat{b}})$ for all possible transitions $Y \in \Phi_2$ being the actual transition performed in phase 2, and $P(Z|\alpha^{\hat{b}\hat{c}}_{\hat{k}}) \sim \mathcal{N}(\bar{\theta}^{\hat{b}\hat{c}}, \sigma^2_{\hat{b}\hat{c}})$ for all possible transitions $Z \in \Phi_3$ being the actual transition performed in phase 3.

Once $P(X|\alpha^{s\hat{a}}_{\hat{k}})$, $P(Y|\alpha^{\hat{a}\hat{b}}_{\hat{k}})$ and $P(Z|\alpha^{\hat{b}\hat{c}}_{\hat{k}})$ for all possible transitions X, Y and Z are computed, the probability $P(\hat{k} = k')$ of each of the 64K possible combination keys k' being the actual combination $\hat{k}$ entered by the target user can be determined as:

$$P(\hat{k} = k') = P(X|\alpha^{s\hat{a}}_{\hat{k}})P(Y|\alpha^{\hat{a}\hat{b}}_{\hat{k}})P(Z|\alpha^{\hat{b}\hat{c}}_{\hat{k}}); \quad \forall(X, Y, Z) \tag{4.3}$$

Where k' can be obtained by substituting $\bar{\theta}^{s\hat{a}}$, $\bar{\theta}^{\hat{a}\hat{b}}$ and $\bar{\theta}^{\hat{b}\hat{c}}$ with X, Y and Z in Equation 4.1, respectively. All the 64K combinations k' can then be *ordered* or *ranked* using $P(\hat{k} = k')$, with a higher value of $P(\hat{k} = k')$ indicating that k' is more likely to be the actual combination $\hat{k}$. Such a ranked list of combinations, denoted as $\bar{\mathbb{K}}$, provides the adversary with a targeted search space to carry out the inference attack. If the actual combination key $\hat{k}$ lies in the top-r of $\bar{\mathbb{K}}$, then the attack framework is said to succeed after r attempts in the worst-case. The adversary would obviously like r to be as small as possible.

4.6.2 Ranking of Safe Key Predictions

The above probabilistic model for the padlock can be trivially extended to the safe. This is done by calculating probabilities $P(W|\alpha_{\hat{k}}^{s\hat{a}}); \forall W \in \Psi_1$, $P(X|\alpha_{\hat{k}}^{\hat{a}\hat{b}}); \forall X \in \Psi_2$, $P(Y|\alpha_{\hat{k}}^{\hat{b}\hat{c}}); \forall Y \in \Psi_3$ and $P(Z|\alpha_{\hat{k}}^{\hat{c}\hat{d}}); \forall Z \in \Psi_4$ using normal distributions $\mathcal{N}(\bar{\theta}^{s\hat{a}}, \sigma_{s\hat{a}}^2)$, $\mathcal{N}(\bar{\theta}^{\hat{a}\hat{b}}, \sigma_{\hat{a}\hat{b}}^2)$, $\mathcal{N}(\bar{\theta}^{\hat{b}\hat{c}}, \sigma_{\hat{b}\hat{c}}^2)$ and $\mathcal{N}(\bar{\theta}^{\hat{c}\hat{d}}, \sigma_{\hat{c}\hat{d}}^2)$, respectively. Then, the probability $P(\hat{k} = k')$ of each of the 100^4 possible combination keys k' being the actual combination $\hat{k}$ entered by the target user can be determined as:

$$P(\hat{k} = k') = P(W|\alpha_{\hat{k}}^{s\hat{a}})P(X|\alpha_{\hat{k}}^{\hat{a}\hat{b}})P(Y|\alpha_{\hat{k}}^{\hat{b}\hat{c}})P(Z|\alpha_{\hat{k}}^{\hat{c}\hat{d}}) \tag{4.4}$$

Where k' can be obtained by substituting $\bar{\theta}^{s\hat{a}}$, $\bar{\theta}^{\hat{a}\hat{b}}$, $\bar{\theta}^{\hat{b}\hat{c}}$ and $\bar{\theta}^{\hat{c}\hat{d}}$ with W, X, Y and Z in Equation 4.2, respectively. All 100^4 combinations k' can then be similarly ranked in a decreasing order using $P(\hat{k} = k')$.

4.6.3 Search Space Reduction

Although the theoretical combination space for both the Master Lock 1500T and the First Alert 2087F-BD are large enough to make manual brute-force attacks impractical, the padlock has some well-known design limitations. In practice, only a set of 4000 keys are used in the production design of Master Lock, as pointed out in a LifeHacker article [68]. Accordingly, after studying how our probabilistic attack model performs on the entire 40^3 key space, we also analyze how our attack can improve predictions within the already reduced space of $|\bar{\mathbb{K}}| = 4000$ combinations. We are not aware of similar limitations in the First Alert safe.

4.7 Evaluation

We conduct thorough empirical evaluations of the proposed inference frameworks in order to assess their performance under realistic lock operation scenarios. Our evaluation results are outlined next.

4.7.1 Experimental Setup

We evaluate the proposed inference frameworks by means of smartwatch gyroscope data collected from a set of human subject participants who performed unlocking operations on the Master Lock 1500T padlock and the First Alert 2087F-BD safe with the watch-wearing hand. For our experiments, we employed a Samsung Gear Live smartwatch which runs Android Wear 1.5 mobile OS and is equipped with an InvenSense MP92M 9-axis Gyro + Accelerometer + Compass sensor. The smartwatch's gyroscope sensor was sampled at 200 Hz, and the samples were transmitted over a Bluetooth connection to a paired Android smartphone (specifically, a Samsung I9500 Galaxy S4). The smartphone recorded the received sensor data stream into labeled files, which were later used for training and validation (testing). All preprocessing, training and testing were performed on a server equipped with dual Intel Xeon L5640 processors and 64 GB of RAM. During the data collection, participants are clearly explained the unlocking procedure for each lock. The locks are placed on a flat table and participants sit on a chair across the table while unlocking. For the first part of our evaluation (sections 4.7.2 and 4.7.3), we collect and use data from the participants' right hand (i.e., the right hand was used to unlock) in a controlled setting. In this setting, each combination is dictated one at a time to the participants who would then correctly enter it on the lock. Our only objec-

tive for collecting unlocking-related motion data from participants was to employ it for a realistic evaluation of the proposed inference frameworks. Our data collection procedure posed no safety or ethical risks to participants, and no private or personally identifiable information (including, combinations of personal locks/safes) was collected from participants. This study is approved by our institution's IRB.

4.7.2 Deterministic Attack Framework Results

We evaluate the performance of the deterministic framework by measuring the standard deviations of the inferred transitions $\bar{\theta}^{ij}$ from the corresponding ground-truths θ^{ij} for each phase of the unlocking operation. We specifically evaluated three different inference strategies: i) inferring transitions ($^{+}\bar{\theta}^{ij}$) solely using *positive displacements* ($^{+}\alpha^{ij}$), ii) inferring transitions ($^{-}\bar{\theta}^{ij}$) solely using *negative displacements* ($^{-}\alpha^{ij}$), and iii) *averaging* inferences ($\frac{^{+}\bar{\theta}^{ij}+^{-}\bar{\theta}^{ij}}{2}$) obtained individually using positive and negative displacements. Our objective is to determine if transition inference using any one of the above displacement parameter is better than the other.

4.7.2.1 Results for Padlock

The training dataset for the Master Lock 1500T padlock is composed of data collected from 3 participants (who are the authors, acting as the adversary). Each participant entered 40 different 3-digit combinations, covering all of the 120 possible transitions (40 in each of Φ_1, Φ_2 and Φ_3). This data entry was repeated 3 times by each participant, resulting in a total of 9 complete datasets which is used for training the deterministic attack model. The testing dataset was collected later from

a different set of 10 participants (non-authors)[2]. Each of these test participants entered 4 different 3-digit combinations covering 12 of the 120 possible transitions (4 in each of Φ_1, Φ_2 and Φ_3), and repeated the data entry 3 times. The combination of data collected from all the 10 participants resulted in 3 complete test datasets covering all the 120 possible transitions. The data collection task is a non-trivial and time-consuming process due to the high cognitive workload associated with entering new and previously unknown combinations which resulted in a significant number of input errors by the participants. All input errors during data-collection were closely monitored and eliminated from the final datasets, and participants were asked to re-enter combinations on which errors occurred. We took utmost care to ensure that our test dataset is complete (covering all transitions) and reasonably heterogeneous (from 10 different participants) to avoid any bias in the evaluation results. The evaluation results, outlined next, are using the averaged prediction over all the 3 test datasets.

Table 4.1 shows the linear least-squares fittings for α^{sa}, α^{ab} and α^{bc}, learned from the 9 training sets. These learned linear least-squares fitting parameters (m and n) are then used within the deterministic framework to infer the 120 unique transitions in the test dataset. Figure 4.4(a) (Right Hand results) shows the standard deviations in inference errors for the inferred transitions in phase 1 ($\bar{\theta}^{sa}$), in phase 2 ($\bar{\theta}^{ab}$) and in phase 3 ($\bar{\theta}^{bc}$). We can see that the inference averaging method ($\frac{^{+}\bar{\theta}^{ij}+^{-}\bar{\theta}^{ij}}{2}$) resulted in

[2]The training dataset for all experiments were collected independently and before the test participants were identified/recruited, which gives us the worst-case results. However, an adversary could be more successful by personalizing the training process for the user being targeted.

Table 4.1: Linear least-squares fittings for the padlock.

	m (Slope)	n (α-intercept)
$^{+}\alpha^{sa}$ (81-120):	0.0836	0.3272
$^{-}\alpha^{sa}$ (81-120):	-0.1269	0.3714
$^{+}\alpha^{ab}$ (41-80):	0.0854	0.9360
$^{-}\alpha^{ab}$ (41-80):	-0.1163	0.3301
$^{+}\alpha^{bc}$ (1-40):	0.0737	2.0387
$^{-}\alpha^{bc}$ (1-40):	-0.1173	0.0061

Table 4.2: Linear least-squares fittings for the safe.

	p (Slope)	q (α-intercept)
$^{+}\alpha^{sa}$ (401-500):	0.0153	19.5492
$^{-}\alpha^{sa}$ (401-500):	-0.0266	-8.8471
$^{+}\alpha^{ab}$ (201-300):	0.0010	7.9046
$^{-}\alpha^{ab}$ (201-300):	-0.0386	-2.3798
$^{+}\alpha^{bc}$ (101-200):	0.0170	3.6319
$^{-}\alpha^{bc}$ (101-200):	-0.0460	0.4906
$^{+}\alpha^{cd}$(1-100):	0.0305	1.7663
$^{-}\alpha^{cd}$ (1-100):	-0.0483	-0.1058

lowest error for the inference of transitions in phase 1 (specifically, 12.27 units) and phase 2 (8.49 units), respectively. However, inference using negative displacement ($^{-}\alpha^{bc}$) resulted in the lowest error in phase 3 (4.82 units). *We can also see that the inference of shorter transitions are more accurate than longer ones.* This observation is intuitive and could be attributed to the differences in the biomechanics of the diarthrodial joints [82] of the test and training participants. These joints play an important role during the unlocking operation and the errors due to biomechanical differences could add up for longer transitions, thus making their inference more error-prone.

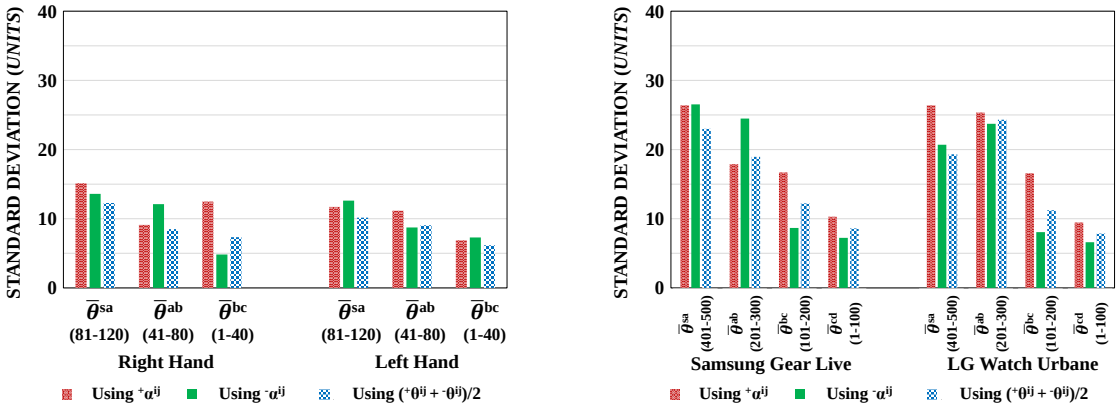


Figure 4.4: Standard deviations in inference error for (a) – three padlock phases, and (b) – four safe phases.

4.7.2.2 Results for Safe

The training dataset for the First Alert 2087F-BD safe is composed of data collected from 3 participants (who are the authors). Each participant entered 100 different 4-digit combinations, covering all of the 400 possible transitions (100 in each of Ψ_1, Ψ_2, Ψ_3 and Ψ_4), which resulted in 3 complete training datasets. Testing dataset was collected later from a set of 10 different participants (non-authors), where each participant entered 2 different 4-digit combinations covering 8 of the 400 possible transitions (2 in each of the transition sets $\{405, 410, 415, ...500\}$, $\{205, 210, 215, ...300\}$, $\{105, 110, 115, ...200\}$ and $\{5, 10, 15, ...100\}$). Each participant repeated entering each combination 3 times, which resulted in 3 partially complete test datasets of 80 evenly distributed transitions. Due to a slightly more complex and longer unlocking procedure of the safe (compared to the padlock), we observed a larger number of participant errors during combination entry. As before, all input errors were closely

monitored and removed from the final datasets. Due to a large combination space, in addition to the more complex unlocking procedure, we restricted ourselves to only partial test datasets for the safe. However, we made sure that the test dataset is uniform in terms of the distribution of the various transitions and the participants that recorded those transitions to avoid any bias in the evaluation results. The evaluation results, outlined next, are using the averaged prediction over all the 3 test datasets.

Table 4.2 shows the linear least-squares fittings for α^{sa}, α^{ab}, α^{bc} and α^{cd}, learned from the 3 training sets. These learned linear least-squares fitting parameters (p and q) are then used to infer the 80 unique transitions in the test datasets. The standard deviations in inference errors for the inferred transitions in phase 1 ($\bar{\theta}^{sa}$), in phase 2 ($\bar{\theta}^{ab}$), in phase 3 ($\bar{\theta}^{bc}$) and in phase 4 ($\bar{\theta}^{cd}$) are outlined in Figure 4.4(b) (Samsung Gear Live results). We can see that the inference averaging method resulted in the lowest error for the inference of transitions in phase 1 (specifically, 22.99 units), while inference using positive displacement ($^{+}\alpha^{ab}$) resulted in the lowest error for the inference of transitions in phase 2 (17.86 units). For transitions in phase 3 and phase 4, inference using the corresponding negative displacements (i.e., $^{-}\alpha^{bc}$ and $^{-}\alpha^{cd}$) resulted in lowest errors (8.66 and 7.23 units, respectively). *Similar to the padlock case, we can observe that inference of shorter transitions in safe combinations are more accurate. Moreover, we also observe that the standard deviations of inference errors for the safe are relatively higher compared to the padlock.* We believe that this is due to the higher concentration of numbers on the safe's lock dial, compared to the padlock's dial, for the same angular displacement.

4.7.3 Probabilistic Attack Framework Results

We evaluate the performance of the probabilistic attack model by evaluating the overall success probability of test combination keys being present in the top-r of their corresponding ranked inferred combination sets.

4.7.3.1 Padlock Key Predictions (64K)

We first evaluate the success probability of finding an entire padlock test combination key within the top-r of the corresponding set of 64K candidate keys, ranked using the probabilistic model. The 64K unique test combinations were obtained by *combination* of Φ_1, Φ_2 and Φ_3 datasets. In this case, rather than using all the three methods for the inference of the individual transitions of the test combination, i.e., inference using only positive displacements, only negative displacements, and averaging individual inferences, we optimize the overall combination inference by selecting the inference method with the lowest error in each phase, for inferring transitions of the test combination key in that phase. Thus for all the 64K test padlock combinations, the first two phases were inferred using inference averaging method, and the third phase using negative displacements. The value of r was increased from 50 to 64,000 in varying steps. Figure 4.5(a) shows the success probability of finding a test combination within the top-r of the corresponding probabilistically ranked (using Equation 4.3) set of candidate keys. 688 test combinations (out of a total of 64K test combinations) were found in the top-50 of their corresponding ranked inferred combination set, which equates to a 1.07% overall probability of success. Compared to this, the probability of correctly picking a test combination after 50 random guesses is only 0.078%. This implies that for $r = 50$ the proposed proba-

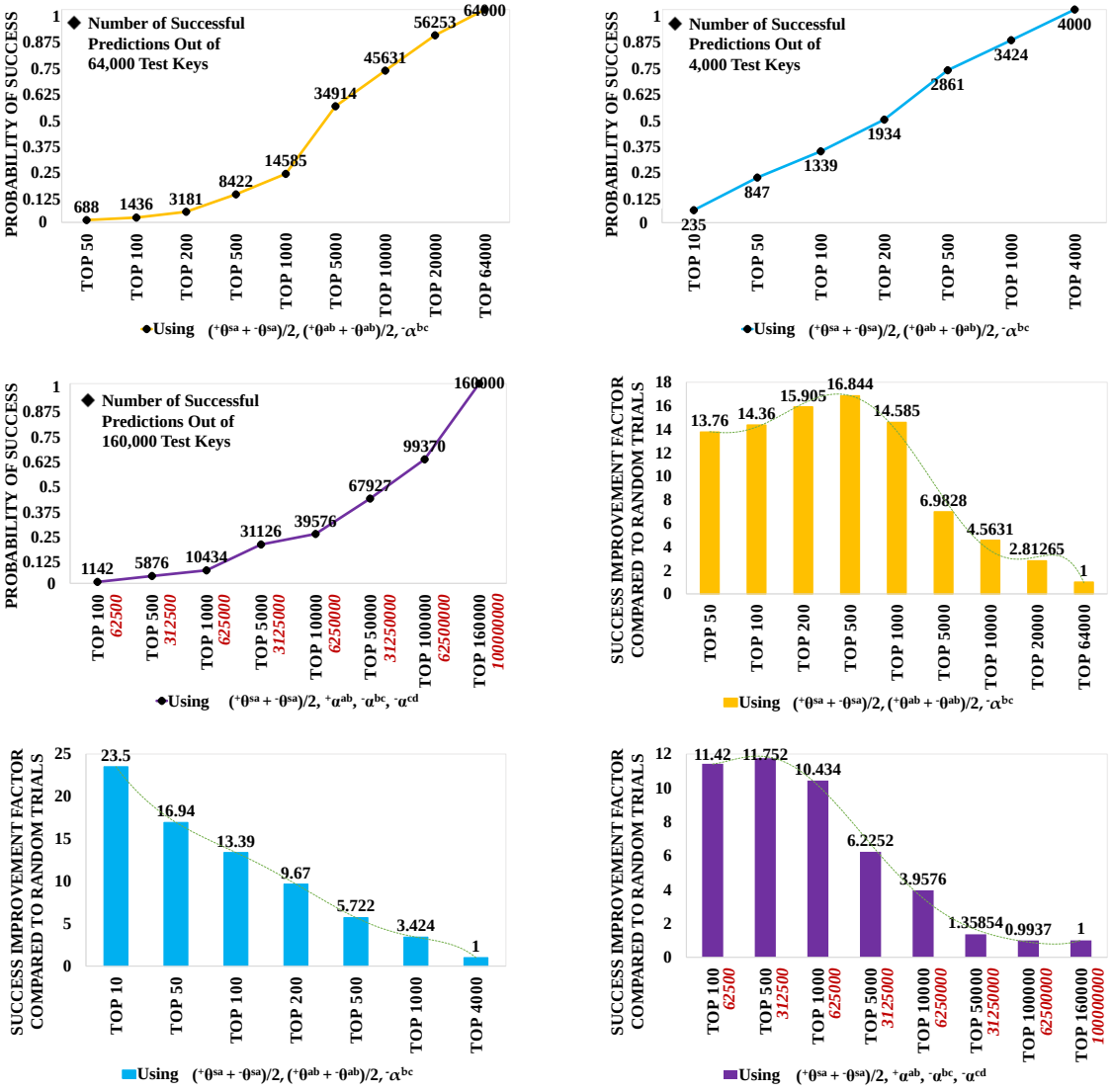


Figure 4.5: (a) − Top-r success probabilities for inferred padlock combinations using 64K test combinations; (b) − Top-r success probabilities for inferred padlock combinations using 4K test combinations; (c) − Top-r success probabilities for inferred safe combinations using 160K test combinations; (d), (e), (f) − Success improvement factors compared to random trials, for the padlock test set of 64K test combinations, padlock test set of 4K test combinations and safe test set of 160K test combinations, respectively.

bilistic model achieves an improvement by a factor of 13.76 over random guessing. Despite the low overall success probability, the above results are indicative of the fact that certain combinations (688 test combinations) are easier to infer than others. Figure 4.5(d) show similar improvements factors for all other top-r cases. These results indicate that an adversary can significantly reduce the search space, and still have high probability of success. As a result, *the cumulative probability of success using the probabilistic model is much higher with 'limited' number of trials, compared to random guessing or the deterministic attack.*

4.7.3.2 Padlock Key Predictions (4K)

We again evaluate the success probability of finding an entire test combination key within the top-r of the corresponding set of candidate keys ranked using the probabilistic model, but this time using the only the 4K implemented padlock combinations (outlined in Section 4.6.3) as test combinations. Similar to the 64K analysis, for all the 4K test padlock combinations, the first two phases were inferred using inference averaging method, and the third phase using negative displacements. The value of r was increased from 10 to 4,000 in varying steps. Figure 4.5(b) shows the success probability of finding a test combination within the top-r of the corresponding probabilistically ranked (using Equation 4.3) set of candidate keys. 235 test combinations (out of a total of 4K test combinations) were found in the top-10 of their corresponding ranked inferred combination set, which equates to a 5.87% overall probability of success. Compared to this, the probability of correctly picking a test combination (among all the implemented keys) after 10 random guesses is only 0.25%. This implies that for $r = 10$ the proposed probabilistic model achieves an

improvement by a factor of 23.5 over random guessing. Figure 4.5(e) show similar improvements factors for all other top-r cases. *These results indicate that an adversary can significantly reduce the combination search space by leveraging on both known mechanical flaws and eavesdropped wrist-movements.*

4.7.3.3 Safe Key Predictions (160K)

We next evaluate the success probability of finding an entire test combination key within the top-r of the corresponding set of candidate keys ranked using the probabilistic model. For this analysis, we test 160K safe combinations $k = \langle a, b, c, d \rangle$ distributed evenly across the entire 100^4 combination space ($\theta^{s\hat{a}} \in \{405, 410, 415, ...500\}$, $\theta^{\hat{a}\hat{b}} \in \{205, 210, 215, ...300\}$, $\theta^{\hat{a}\hat{b}} \in \{105, 110, 115, ...200\}$ and $\theta^{\hat{a}\hat{b}} \in \{5, 10, 15, ...100\}$). The 160K unique test combinations were obtained by *combination* of Ψ_1, Ψ_2, Ψ_3 and Ψ_4 datasets. Similar to padlock key predictions, we optimize the overall combination inference by selecting the inference method with the lowest error in each phase, for inferring transitions of the test combination key in that phase. Thus for all the 160K test safe combinations, the first phase was inferred using inference averaging method, the second phase was inferred using positive displacements, and the last two phases using negative displacements. The value of r was increased from 100 to 160,000 in varying steps. It should be noted that in the case of the safe, we only probabilistically rank the (evenly distributed) 160K keys appearing in the test set rather than the entire safe combination space of 100^4. This is primarily due to the computational challenge associated with computing probabilities for, and then ranking, 100 million combination keys for each of the 160K test combinations, which is an extremely time-consuming process. The adversary, however, does not have a similar problem

because the adversary has to rank the entire combination space of 100^4 for only a few test keys, which is relatively easier to compute. Figure 4.5(c) shows the success probability of finding a test combination within the top-r of the corresponding probabilistically ranked set of candidate keys. 5876 test combinations (out of a total of 160K test combinations) were found in the top-500 of their corresponding ranked inferred combination set, which equates to a 3.67% overall probability of success. Compared to this, the probability of correctly picking a test combination after 500 random guesses is only 0.31%. This implies that for $r = 500$ the proposed probabilistic model achieves an improvement by a factor of 11.42 over random guessing. A straightforward extrapolation of r (multiplying it with a factor of 5^4) puts the value of r at 312500 for achieving similar improvement if the entire combination space of 100^4 combinations is ranked. Readers should note that labels marked in red in Figure 4.5(c) are extrapolated values of r. Figure 4.5(f) show similar improvements factors for all other top-r cases. *These results indicate that the proposed framework can achieve significant reduction of the combination search space for the safe as well.*

4.7.4 Cross-Device Performance

So far we have evaluated our inference frameworks in a same-device setting where the same smartwatch hardware (Samsung Gear Live smartwatch with an InvenSense MP92M sensor) was used for collecting both the training and testing datasets. However in a practical setting, an adversary may be unaware of, or may not possess, the precise wrist-wearable hardware used by the target user. Thus, it is critical to assess the performance of the inference frameworks when different wrist-wearable hardwares are used for training and testing (attack) purposes. In other words, a

comparison of the earlier evaluation results with results using test data from a different smartwatch would tell us if the proposed inference frameworks are inter-operable across different devices. For brevity, we analyze the cross-device performance of the inference frameworks only for the First Alert 2087F-BD safe. For this, we collect the same set of test data for the safe as detailed in Section 4.7.2.2 by using a LG Watch Urbane smartwatch equipped with an on-board InvenSense M651 6-axis Gyro + Accelerometer sensor (sampled at 200 Hz) and running Android 2.0 mobile OS. We then employ the linear function $\alpha = p\theta + q$ (Table 4.2), trained from the data collected with a Samsung Gear Live (as outlined in section 4.7.2.2).

A comparison of the standard deviations in inference error (Figure 4.4(b)) does not show significant change in prediction results we observed earlier. A *pair-wise two-tailed t-test* [128] of all the values in both set of results, resulted in $t = 0.11; p = 0.915$. *The small value of t along with a high p value indicates that the mean difference between the two sets of results is not significant, with high probability.* As a result, the adversary can train the inference models using data from one device and use these trained models to carry out inference attacks on data from a different wrist-wearable device, provided data from this device is sampled at the same frequency.

4.7.5 Cross-Hand Performance

All evaluations of our inference models so far have been accomplished using training and testing datasets collected from subjects who only used their right hand for the unlocking operation. However in a practical setting, a target user may not perform the unlocking operation with the same hand that the adversary has trained its models on. Thus, it is important to assess the performance of the proposed inference

frameworks when training and testing data corresponding to the unlocking operation comes from different hands. In other words, we would like to analyze if the proposed inference models trained using unlocking data from one hand (say, right) can be used to infer combinations entered using the other hand (say, left). For this we collect the same test data for the padlock as detailed in Section 4.7.2.1, but this time the 10 participants wore the Samsung Gear Live smartwatch on their left hand and entered the test combinations on the padlock with their left hand. We then employ the linear function $\alpha = m\theta + n$, trained earlier using the right hand data (Table 4.1), to infer transitions in each phase using the deterministic model.

A comparison of the standard deviations in inference error (Figure 4.4(a)) does not show significant change in prediction results we observed earlier using same-hand predictions. *A pair-wise two-tailed t-test of all the values in both set of results, resulted in* $t = 1.33; p = 0.219$. *The value of t indicates that there exists minor difference between the two sets of results. However, due to the low p value, we cannot conclusively say that an adversary can focus on training a single model with either hand's data, and use it on both left and right handed targets.* That being said, it is not difficult for an adversary to train two different models (one per hand), and use the appropriate model per target user.

4.7.6 Real-Life Detection and Prediction

Next, we evaluate the performance of our unlocking activity recognition algorithm (Section 4.4.3) and inference framework under a *real-life setting*. To facilitate a real-life experiment with three new participants, we handed out a Samsung Gear Live smartwatch, a paired smartphone and a padlock, for them to take home. The

watch was installed with our recording application and the unlock activity recognition algorithm. We collected x-axis gyroscope data for the duration of approximately 1 day, during which the participants were instructed to perform at least three padlock unlock operations with a 3-digit combination of their choice (among the 4K mechanically valid combinations), at random intervals. *Overall, our unlock activity recognition algorithm yielded 100% recall and 80% precision, with a total of 12 true positives, 3 false positives and 0 false negative.* Interestingly, the 3 false positives were reportedly due to activities similar to padlock unlocking, such as when washing hands after rotating a washer tap/faucet, and while using a screw driver. Next, we evaluate the prediction accuracy of the secret combination entered by each participant by using the last three instance of their unlocking gyroscope time-series data, as extracted from the entire day's data. Applying the same inference model for ranking keys among the 4K implement keys, used in Section 4.7.3.2, the real key entered by the three participants were ranked at 42, 85 and 112 (out of 4000). This demonstrates the extent to which the proposed attacks can reduce the combination search space even in uncontrolled real-life settings.

4.8 Discussions

4.8.1 Characteristics of Inferred Combinations

Results of our deterministic attack model indicated that shorter transitions can be more accurately inferred than longer transitions. To see if this phenomenon carries over to full combinations as well, we analyzed the length (in terms of transition units) of the 235 padlock combinations (out of 4K) that were successfully inferred

within top-10 trials (Figure 4.5(b)). The shortest padlock combination can be of 123 transition units ($81 + 41 + 1$), where as the longest combination can be of 240 transition units (120+80+40). On this scale, 91.06% of the 235 padlock combinations that were successfully inferred within top-10 trials, were shorter than 150 transition units. Based on this observation, we can conclude that *key combinations that require less rotational displacement have better inference probability, and users should avoid purchasing locks preset with such combinations.*

4.8.2 Limitations

Starting Point: Without a known starting point the adversary will have to try all numbers on the dial as the starting point, thereby significantly increasing the average number of trials it would take to be successful. However, because the adversary will require physical access to the lock in order to try predicted keys, it is not unrealistic to assume that they can also learn or preset the starting point. The starting point can be any number on the dial; only the key inference functions (Equations 4.1 and 4.2) must be appropriately initialized according to that starting point. Moreover, the learned least-squares (Tables 4.1 and 4.2) are not affected by a change in the starting point during the attack phase.

Kinesiological Factors: Factors such as the grasping style, hand size, and muscle strength have a significant effect on the biomechanics of the diarthrodial joints of a target user performing an unlocking operation. While we did not encounter any participant in our study with significantly different unlocking styles, it is possible for an adversary to come across a target whose unlocking style is significantly different. However, a competent adversary may be able to train a variety of models based

on different kinesiological factors, and use an appropriate model for each target. Further user study is required to understand if unlocking styles can be classified in to characteristically unique groups.

Affected Users: Our attack assumes that the user wears his/her wrist-wearable on the hand used to unlock the padlock or safe. This may not always be the case, causing the attack to fail. While we did not find any statistics in the literature to deduce the percentage of users who use the same hand for both, according to an on-going online poll with about 5000 participants [3], approximately 38.23% of users prefer to wear watches on their dominant hand. Assuming most users use their dominant hand for unlocking padlocks and safes, a significant number of users can be affected by the proposed attack. Moreover, with the advent of fitness trackers (most of which also have gyroscope sensors), users tend to wear their watches and wrist-based fitness trackers on different hands. Regardless of the exact statistics, we hope that this work will create awareness of this threat.

Generalization of Results: The proposed attack frameworks can be easily extended to work with any other rotation-based mechanical combination lock, with different length of combination keys and different sequence of key entry directions (clockwise/counter-clockwise). But, according to the trends in our evaluation results, it can be challenging to infer longer combination keys (5 or more numbers) and on locks with more concentrated dials (more numbers on the dial face), with high accuracy. However, in a brief study of the most popular consumer-grade padlocks and safes, we found that 5 or more number combination padlocks and safes are very rare in the retail market (Table 4.3). There are several consumer-grade padlocks and

Table 4.3: Popular padlocks and safes retailed by Amazon and Walmart.

Product	Combination Length ↓	Numbers on Dial	Mechanical Limitations
Master Lock 1500iD Speed Dial	Unlimited	4	7501 unique states [48]
First Alert 2087F-BD Safe	4	100	Unknown
SentrySafe SFW082CTB	3	100	Unknown
Master Lock 1500T	3	40	4000 used combinations
Master Lock Padlock 1588D	3	20	Unknown

safes available for purchase, from different manufacturers and retailers. However, most of them employ similar, if not identical, working mechanisms. In Table 4.3, we list the most popular products with rotation-based locks, which are representative of its type. Similar locks produced by different manufacturers are faced with the same level of threat, unless the manufacture introduces additional design changes. For example, a Hollon Home Safe 310D and the First Alert 2087F-BD Safe, both have 4-number combinations with 100 numbers on the dial, resulting in 100^4 possible combinations. However, many larger enterprise-grade safes and vaults are equipped with larger dials which can translate in to more perceptible gyroscope readings on the wrist. As a result, a more accurate inference may be possible for locks with significantly larger dials. The principles used in our attack models can also be used to infer private activities pertaining to other forms of rotary wrist movements, such as numbers entered on a rotary telephone dial, driving trajectory on a steering wheel, etc.

4.8.3 Mitigations

Users can take few preventive measures to avoid falling victim to the proposed attack. A simple measure could be to use the hand without any wrist-wearable for unlocking, or to take off any wrist-wearables before unlocking. Users could also inject noise in the data by shaking their hand in between the unlock operation. More complex protection mechanisms can include dynamic access control of zero-permission sensors such as the gyroscope. As some of the previous works suggested [23, 71], a dynamic access control can take advantage of contextual information to automatically cut-off sensor access when users are detected to be vulnerable. A potential solution in this direction could be to use our unlocking activity recognition algorithm (presented in Section 4.4.3) in a real-time fashion, so as to disable the gyroscope after the first few *spins*.

4.8.4 Other Attack Vectors

Padlock and safe combinations are also susceptible to other forms of *non-intrusive* attacks, such as visual shoulder-surfing when the target user is unlocking. Visual access to a lock's dial when the user is unlocking, can result in more precise key inference than our wrist motion based inference framework. However, there is high likelihood that the user will notice a visual observer (human or camera), and shield their unlocking activity. It may also be possible to use visual on the user (instead of the lock) to perform timing based inference attacks [34]. As part of future work, we will investigate other potential side-channels and evaluate the possibility of combining multiple side-channel attack vectors to improve the overall key prediction accuracy.

4.9 Conclusion

In this chapter, we presented a new motion-based attack to infer mechanical lock combinations from smartwatch gyroscope data. A comprehensive evaluation using a commercial padlock and safe demonstrated that our framework can significantly reduce the combination search space for an adversary. The combination key search space can be further reduced in case of the padlock by leveraging on mechanical design flaws. We also observed that the performance of the proposed inference frameworks do not significantly degrade when model training and inference tasks are carried out using different smartwatch hardwares or different unlocking hands. Finally, we also demonstrated the efficacy of the proposed inference attack in a real-life setting.

Parts of this chapter appeared in [73].

CHAPTER 5
PROTECTING USER INTERACTIONS: DESIGN-TIME

5.1 Introduction

In order to thwart the privacy threats validated in last three chapters, effective and usable techniques for detection and mitigation of wearable device misuse will be critical and urgently needed. Consequently, we propose *design-time* protection measures in this chapter, which tries to prevent inference attacks by altering the interaction interfaces. Specifically, we present and evaluate the effectiveness and usability of *RandomPad* for protecting mobile keypad interactions, and *EyePad* for protecting external keyboard interactions.

5.2 RandomPad - Protecting Mobile Keypad Interactions

Users have been increasingly using their mobile devices and smartphones to enter personal and private information, such as, PIN, credit card numbers, passwords and telephone numbers. However, touchscreen-based numeric keypads on these mobile devices and smartphones are becoming increasingly more vulnerable to *side-channel keystroke inference attacks*, which results in a serious invasion of privacy of mobile users. Kune et al. [34] leveraged on a common assumption that an audio feedback to the user is imparted for each button pressed, and demonstrated the possibility of inferring keystroke sequences based on time delays between keystrokes. Yue et al. [122] used computer vision to analyze the shadow formation around the fingertip to automatically locate the touched points. Simon et al. [98] used microphone to detect

touch events, while the camera is used to estimate the smartphone's orientation, and correlate it to the position of the digit tapped by the user. Sun et al. [105] used video recordings of the backside of a tablet to infer typed keystrokes, based on the observation that keystrokes on different positions of the tablet's soft keyboard cause its backside to exhibit different motion patterns. Zhang et al. [125] analyzed finger smudges left on the touch screen surface to infer touch patterns, with remarkable success.

Motion sensors, such as, accelerometer and gyroscope, represent another class of side-channels for accomplishing keystroke inference attacks that have been highly researched. Tapping at different locations on a touchscreen results in unique movements of the mobile device which can be captured by eavesdropping on-board motion sensors. Cai et al. [20] were one among the first to use this observation to train multi-class classifiers for each of the ten spatially separated numbers of a keypad, and were able to correctly predict up to 70% of test keystrokes. Owusu et al. [86] extended the side-channel attack from numeric keypads to soft QWERTY keyboards. In Chapter 2 used a smartwatch to demonstrate that an external device's motion sensors can also be used to infer keystrokes made on a mobile keypad [74]. Lack of any access control to motion sensors on existing mobile (and wearable) operating systems further improves the feasibility of such motion side-channel based inference attacks.

Interestingly, all the above attacks share one common assumption: *the numeric keypad employed by the target user has a standardized key layout (Figure 5.1) known to the adversary.* Intuitively, this means that if the keypad layout is changed from the standardized layout unbeknownst to the adversary then the above attacks will

perform poorly. Thus, such a dynamic keypad layout strategy is an appealing defense strategy against side-channel keystroke inference attacks. However, as an adversary can also re-train the attack framework for the new keypad layout, changing the keypad layout just once (or in a predictable fashion) will not be an effective defense. In order to prevent an adversary from knowing the keypad layout in use at any given time, this change in layout should be *randomized*. In Section 5.5, we present different keypad randomization strategies, in terms of key size, sequence and location. The primary goal of these strategies is to reposition the on-screen keys such that an adversary cannot correctly predetermine the keypad layout in use at any given time. Without an accurately predetermined keypad layout, the adversary will be unable to train or set up the attack framework, and an improperly trained attack framework will result in erroneous inference of keystrokes. Interestingly, a major smartphone manufacturer recently introduced a custom authentication method called "Random PIN entry" [66], which implements a randomization strategy in order to restrict side-channel attacks.

While randomized keypads could provide an effective defense against keystroke inference attacks, it also raises usability concerns. The default keypad (Figure 5.1) is widely used and many users have gradually become habituated to the static layout. Thus, randomizing the keypad brings two new challenges: (a) users may be uncomfortable typing on a keypad different from the one they are habituated to, and (b) as the keypad changes randomly, users will always face an unfamiliar keypad. If users are discomforted to a level where they may opt not to use random keypads for the sake of privacy, then it cannot be proposed as an effective defense mechanism

against inference attacks. Therefore, before recommending the use of randomized keypads for privacy protection, it is critical to evaluate the usability of the various layout randomization strategies. In order to achieve this goal, we comprehensively assess the *usability* and *perceived workload* of typing on keypads generated by each of the proposed randomization strategies with the help of actual typing experiments involving a diverse set of human subjects. We also compare the *rate of mistyping* among all strategies, and attempt to determine whether one strategy is more usable than the others. In addition to this, we also evaluate if usability is positively influenced by distinguishable visual features, i.e., by coloring each key with an ascending shade of gray. Finally, by comparing their empirical ease-of-usage with their analytical security assurance, we attempt to study the privacy-usability trade-off (if one exists) in using different types of randomized keypads.

5.3 RandomPad - Attack Description

We consider the scenario of a potential victim typing on a smartphone's *numeric touchscreen keypad* (Figure 5.2) in the presence of an adversary whose goal is to infer the keystrokes typed by the victim. For *on-device* inference attacks using device sensors as information side-channels, the adversary may bug or eavesdrop on the target's smartphone [20, 86, 98] and/or paired smartwatch (Chapter 2) by installing a malicious application which records the activity of certain on-board sensors. This step can be achieved by exploiting known software vulnerabilities or by tricking the victim into installing a *Trojan* or a malicious code hidden within a useful application. The malicious application also maintains a *covert communication channel*

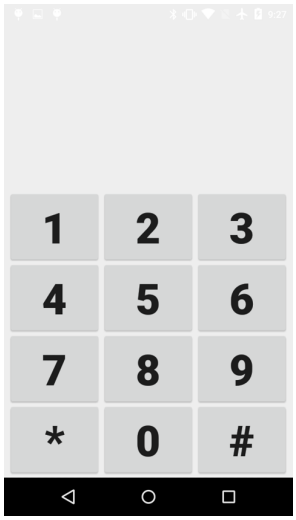


Figure 5.1: Default keypad.

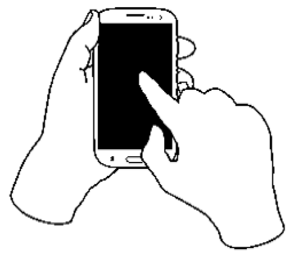


Figure 5.2: A common typing scenario.

with the adversary, and periodically uploads the eavesdropped data to an adversarial server by means of this channel. For *external* inference attacks [34, 105, 122, 125], the adversary captures relevant keystroke characteristics from a physically close position, using appropriate eavesdropping devices, such as, microphones or wireless transceivers. We assume that the adversary has sufficient storage and computational resources to process the eavesdropped data and successfully carry out both types of attacks. However, ***we assume that the adversary cannot visually eavesdrop or observe the keypad (and the victim's keystrokes)*** and does not have the ability to install system level key-logging applications to directly obtain the typed keystrokes.

5.4 RandomPad - Related Work

5.4.1 Protection Against Side-Channel Attacks

Due to the increasing use of various sensors in mobile and wearable devices as information side-channels to accomplish privacy invasive inference attacks, the topic of defending against such attacks has gained prominence. Cai et al. [22] drew attention on the limitations of current mobile systems in mitigating side-channel attacks. They also pointed out the following desirable properties in any defense solution: (i) **Security:** the solution must protect against side-channel inference attacks, (ii) **Usability:** ideally, the solution should require no extra effort from users and if extra effort is unavoidable, it should not disrupt the users' work flow, (iii) **Backward and Forward Compatibility:** the solution should require no or minimal modification to existing applications and operating systems, (iv) **Performance:** the solution should have no or minimal overhead, and (v) **Versatility:** the solution should be deployable on various types of mobile hardware, software, and user interfaces. If a defense solution fails to fulfill any of these properties, it may not be well accepted by users.

Controlling access to sensors that has the potential to be used as side channels is one form of defense mechanism that can be used. However, as mobile and wearable systems currently offer very limited access control options (mostly restricted to location and microphone sensors), fine-grained access control to all sensors will require major modifications to these systems. Context-aware access controls [25] could relieve users from manually changing and adjusting access settings, however they would add significant performance and energy overhead, are non-versatile and difficult to setup and would require major modifications to current operating systems.

Furthermore, sensor access controls do not protect against applications that gain legitimate access to these sensors. Enforcing system-wide reduced sensor sampling rate or disabling sensors is one suggested defense against on-device keystroke inference attacks [74, 86]. However, while system-wide down-sampled or disabled sensors may provide protection, it may disrupt useful non-malicious applications as well. *Moreover, neither access control, nor limiting sampling rate, protects against external inference attacks.* There has been limited work on protecting smartphone users against external side-channel attacks. Shrestha et al. [97] proposed the injection of noise in motion sensor readings, in order to protect against motion sensor based inference attacks. However, their solution is ineffective against most other classes of keystroke inference attacks. Alternate forms of authentication (e.g., biometrics) are also becoming popular, but the vast majority of mobile devices are not equipped with the enabling hardware and/or software. Thus, mobile users will continue to use touch screen-based keypads to enter sensitive information, including authentication data, and there is an increasing need for a keypad protection mechanism that satisfies most of the design criteria identified by Cai et al. [22].

5.4.2 Protection by Randomization

Randomizing the keypad prevents an adversary from predetermining the keypad layout, which can serve as an effective defense against both external and on-device attacks. Randomized keypads are already known to be used commercially in electronic door access control systems [101], although with limited flexibility in terms of available set of randomization strategies. Ryu et al. [93] were among the first to study randomized keypads and they observed that their randomized keypad resulted

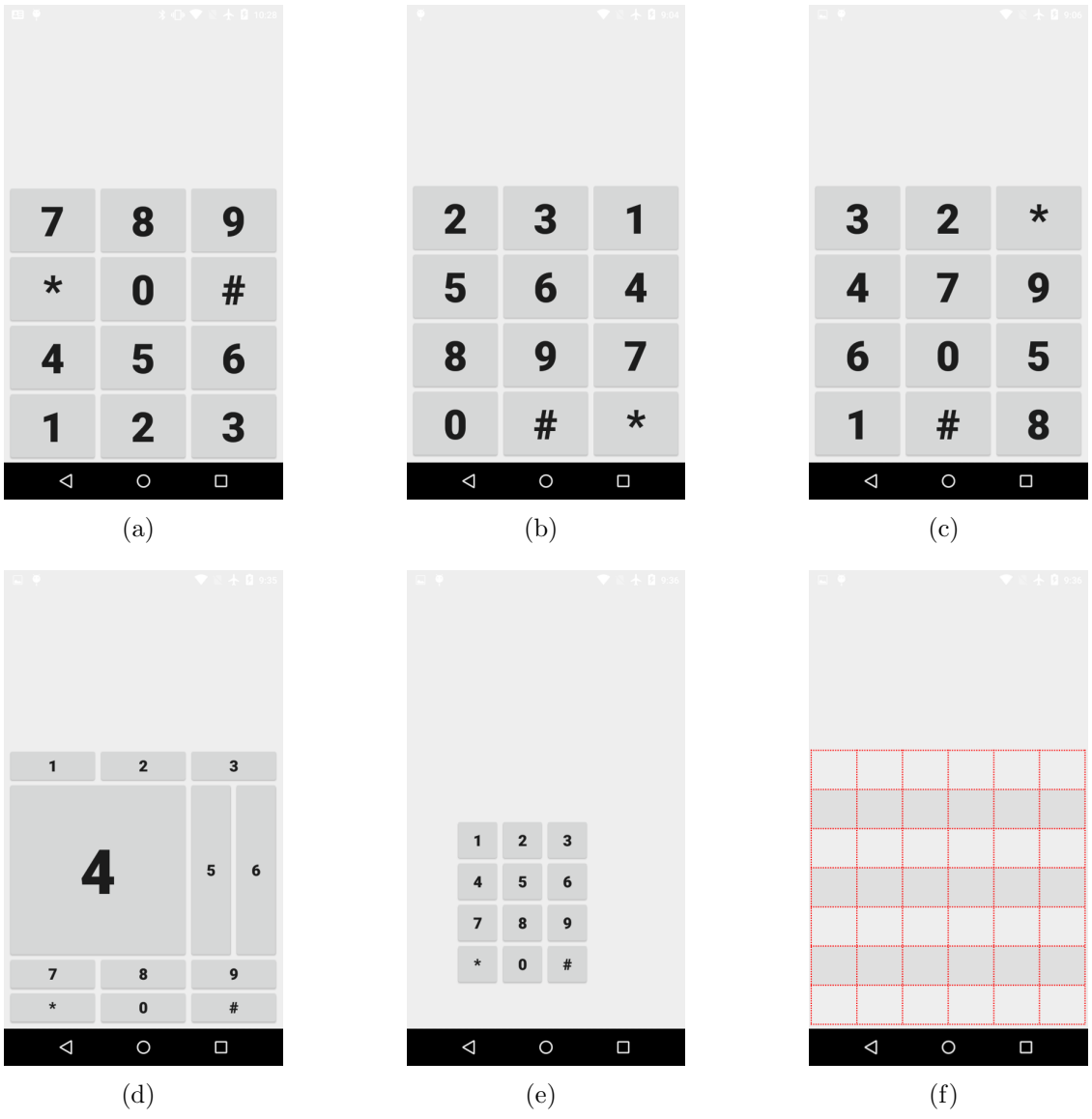
in longer completion times compared to a conventional keypad. However, their study was not geared towards mobile devices, considered only one randomization strategy and did not comprehensively evaluate user workload and other usability parameters except completion times. In this research effort, we propose, implement, and comprehensively evaluate different randomized keypads (or *RandomPad*) for mobile devices. RandomPad does not add significant overhead on system performance as it is essentially a rearranged keypad layout. It can be easily implemented as a third party application on popular mobile operating systems such as Android and iOS, without requiring support from operating system developers. RandomPad can also be versatile, when implemented according to scalable design principles [49]. With RandomPad, we analyze the remaining two design properties outline in Section 5.4.1: *security* and *usability*.

5.5 RandomPad - Randomization Strategies

We outline five representative strategies that span the entire spectrum of strategies from purely-random to partially-random keypad layouts, i.e., the latter preserves some characteristics of the default layout. For stronger security, keypad randomization can be performed either at the beginning of every keystroke or at the beginning of each typing session.

5.5.1 Key Sequence Randomization

The default keypad follows a sequence of ascending numbers. Key sequence randomization strategies reposition the keys by changing the order of keys, by not



(a) (b) (c)

(d) (e) (f)

Figure 5.3: Examples of (a) RR, (b) CR, (c) IKR, (d) KSR and (e) KLR; (f) The hidden 7×6 grid layout used in KSR and KLR.

following the ascending order. Following are the three key sequence randomization strategies we use in our study, all of which keep the key sizes unchanged:

- **Row Randomization (RR)**: In row randomization (RR), rows from the default keypad are randomly ordered while preserving the order of the numbers within each row. Figure 5.3(a) shows an example of RR.
- **Column Randomization (CR)** In column randomization (CR), columns from the default keypad are randomly ordered while preserving the order of the numbers within each column. Figure 5.3(b) shows an example of CR.
- **Individual Key Randomization (IKR)** In individual key randomization (IKR), individual keys are randomly re-arranged without maintaining any column or row order. Figure 5.3(c) shows an example of IKR.

5.5.2 Key Size Randomization (KSR)

A key size randomization (KSR) strategy preserves the sequence of numbers on the default keypad. Instead, the randomization factor is incorporated within the size of each key. Changing the key sizes also repositions them from their default locations on screen. In our design of KSR, we use a hidden 7×6 grid layout (scaled to fit the width of the screen), as shown in Figure 5.3(f). One randomly selected key is enlarged to appear as a 4×4 block on the grid, other keys in the same row as the large key appear as 4×1 blocks, and all other keys appear as 1×2 blocks on the grid. Figure 5.3(d) shows an example of the KSR strategy. Note that the 7×6 grid layout is not visible to users; only the overlaid keys are visible. As the default sequence is

preserved, it may be necessary to randomize key sizes after each key press to prevent relative positioning based attacks.

5.5.3 Keypad Location Randomization (KLR)

A keypad location randomization (KLR) strategy also preserves the sequence of numbers of the default keypad. The randomization factor is instead incorporated in the location of the keypad, because changing the keypad location repositions all keys from their default locations on the screen. Figures 5.3(e) shows an instances of the keypad location randomization we consider in our study. In this case, we again use a hidden 7×6 grid layout similar to KSR. Each key appears as a 1×1 block, and the entire keypad appears as a randomly selected contiguous 4×3 block on the grid (16 possibilities). The distribution of keys on the hidden 7×6 grid layout happens to be same for both KSR and KLR (Figure 5.4). Similar to KSR, keypad randomization in KLR may need to be done at every key press to prevent relative positioning based attacks. Moreover, key sequence randomization strategies can also be combined with KSR and KLR for additional security.

5.5.4 Security Analysis

Next, we probabilistically analyze the security offered by these five randomization strategies. For this analysis we assume that the adversary is able to cluster keystroke positions not just on a default sized 4×3 keypad, but also for smaller blocks of a 7×6 layout (used in KSR and KLR), without any error (i.e., 100% accuracy). The adversary is also assumed to know the randomization strategy currently in use and that a new randomized keypad layout within the corresponding strategy is used by

the user (victim) for every keystroke (that the adversary is attempting to infer). As the keypad layout is randomized, the best an adversary can do is to guess the mapping between the randomized and default keys. We derive the *successful guessing probability of the adversary* as an indication of the *security assurance* or *guarantee* each strategy provides under such a strong attack scenario. The lower this probability for a particular randomization strategy, the higher is its security assurance.

Consider a twelve key (including "$*$" and "#" keys) keypad in IKR. The probability that an adversary guesses the mapping of a digit correctly is $\frac{1}{12}$ and the probability of correctly guessing the entire mapping (of all the keys) is $\frac{1}{12!}$. However, in case of RR and CR, the adversary can improve its guessing, which is intuitive. Knowing that keys within a row remain in order, for a RR keypad, the adversary only needs to guess the row mapping. The probability that an adversary correctly guesses a row (and thus keys in it) is $\frac{1}{4}$, and all four rows is $\frac{1}{4!}$. Similarly for CR, the probability that the adversary correctly guesses a column (and thus keys in it) is $\frac{1}{3}$, and all three columns is $\frac{1}{3!}$. Due to crossover regions on the keypad, accurately guessing key mappings in KSR and KLR is a bit more complicated. In case of KSR, one large key displaces the position of other keys, leading to non-zero probabilities of multiple keys being at the same on-screen location. As the key distribution possibilities are same for KLR, it also shares the same probability distribution as KSR. Assuming that a keystroke touch can occur uniformly on any of the 42 (7×6) on-screen blocks, the probability of an adversary correctly guessing a key's position is:

$$\frac{1}{42} \sum \frac{1}{N_{i,j}}, \forall i, j \tag{5.1}$$

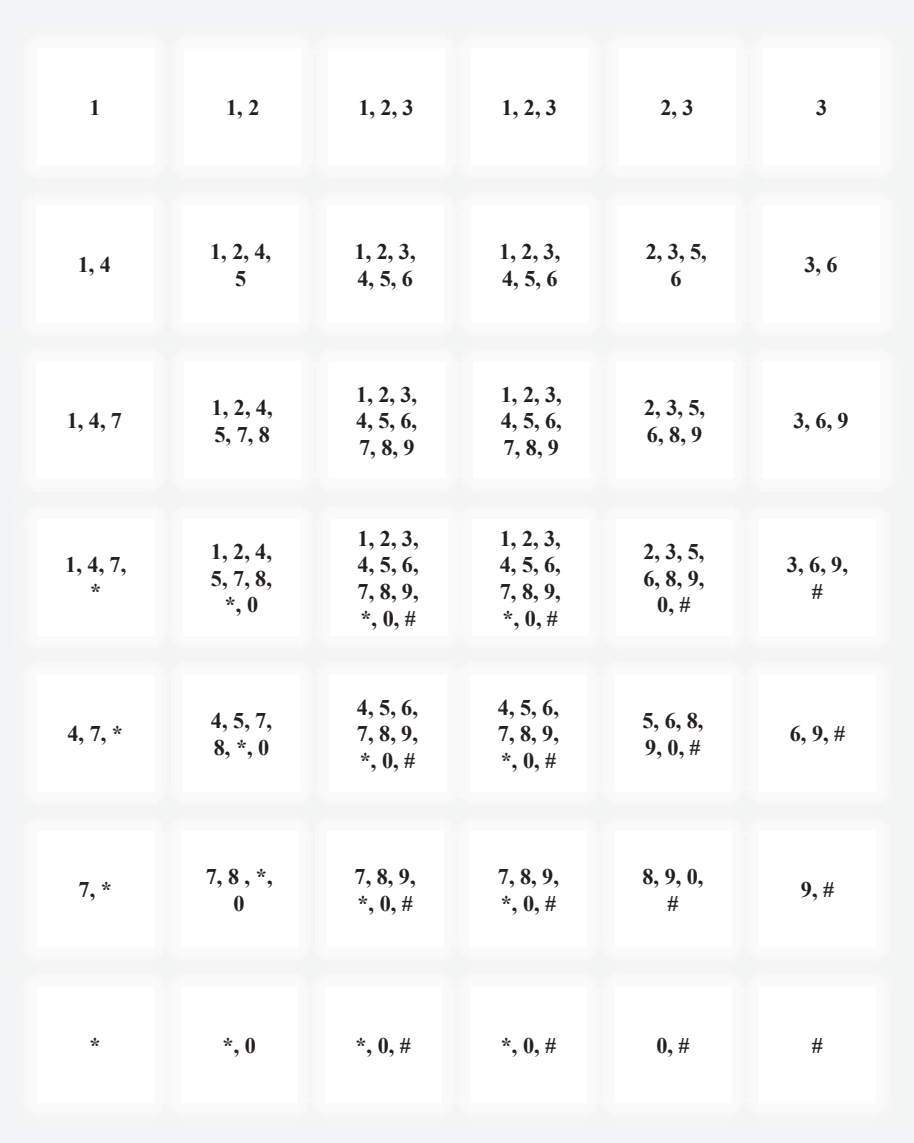


Figure 5.4: KSR and KLR on-screen key distribution possibilities on the hidden 7×6 grid layout.

Table 5.1: Security assurance of the five proposed randomization strategies. Lower rank is better security.

Randomization Strategy	Correct Entire Keypad Guessing Probability	Security Assurance Rank
CR	$\frac{1}{3!} = 0.16667$	5
IKR	$\frac{1}{12!} = 2.08 \times 10^{-9}$	1
KLR	$\frac{1}{16} = 0.0625$	3
KSR	$\frac{1}{12} = 0.08333$	4
RR	$\frac{1}{4!} = 0.04167$	2

where, $N_{i,j}$ is the number of keys that could possibly occupy block i, j. Solving Equation 5.1 using the distribution of keys (Figure 5.4) results in a success probability of 0.34193. Guessing the entire keypad in KSR and KLR is relatively uncomplicated, as the adversary has to guess only the large key in KSR (probability $\frac{1}{12}$) and one among the sixteen possible locations of the keypad in KLR (probability $\frac{1}{16}$). Table 5.1 in ranks the adversary's success probabilities for the different randomization strategies. *These probabilities represent a best-case scenario for the adversary.*

5.6 RandomPad - Human Factors

The above security analysis shows that randomizing keypad layouts is an effective protection strategy against side-channel keystroke inference attacks. It is also efficient from a system performance perspective, easy to implement and versatile. However, a significant concern remains to be answered: "*Will users employ and effectively be able to use such a protection mechanism*"? As the proposed protection mechanism is simply a different and highly dynamic user-interface, we attempt to answer this broad question by using principles and techniques from the area of

human-computer interaction (HCI) [30] and cognitive psychology [56]. Designing usable input interfaces, e.g., keypads, for mobile devices has been a significant technical challenge [47]. For mobile devices, the main constraint in designing usable input interfaces is the screen size, however earlier research has shown that smaller keypad sizes do not negatively affect the efficiency or accuracy of user input [95]. Thus, in this work we will focus only on how random positions and sizes of the keys on the keypad impact their usability. In our quest for answering the above usability question, we will primarily focus on measuring user effort and workload while using these randomized input interfaces (or RandomPad) by means of well-known quantitative and qualitative metrics, as discussed below. We would also like to investigate if certain design changes would improve or reduce user workload.

Keypads with randomized key sequences (e.g., RR, CR, IKR keypads) pose a unique challenge to human cognition. Users may often find themselves searching for a particular key, which would slow down overall typing speed. Physiological factors, such as, visual acuity, light accommodation, dexterity, working memory, and reaction times [37, 104] can further impact this. Thus, time required for the typing task completion and the number of errors during the task are some of the metrics that will be used to evaluate user-effort while using RandomPad. Another commonly used HCI technique to empirically measure the user-effort of interfaces, and thus its usability, is *eye-tracking*. Eye-tracking devices can capture fixation duration and number of fixations while the user is interacting with the interface. The average *fixation duration* indicates how long it takes for users to encode the visual information, which is influenced by the readability of the characters, such as,

font size, font style, spacing and contrast of background and foreground, etc. [90]. The *number of fixations* to complete a task is correlated with the difficulty to locate the target (within the task). In this work, we also use these metrics (captured by means of an eye-tracking device) to quantify the difficulty of the user in locating the keys on RandomPad.

Besides these, we also analyze the usability of RandomPad by employing user-provided subjective workload and usability measures. For instance, *NASA-TLX* [41] is a well-known scale for subjectively measuring *mental workload*. Mental workload measures the subjective experience of the effort to complete a task [15]. A high mental workload is often detrimental to task performance and can reduce the chances of the product or interface being adopted or used by users. The NASA-TLX is a multidimensional scale to measure the perceived workload, including, the mental, physical and temporal demand, overall performance, frustration level and effort. We employ the NASA-TLX scale in our experiments to capture the mental workload of participants after they have used RandomPad. Similarly, we also measure the overall usability of the RandomPad design by using another subjective scale called the *System Usability Scale (SUS)* [19]. The SUS is a 10-item 5-point scale, which produces a usability score ranging from 0 to 100, with a larger value indicating a more usable interface. We feel that a combination of task completion performance measures, eye fixation measures using eye-tracking, and subjective mental workload and usability measures will provide a converging evidence to illustrate the usability of RandomPad, and thus provide some answers to the broad usability related question posed earlier.

As discussed earlier, certain physiological or environmental factors may impact human cognition of the interface, and thus its usability. Pattison and Stedmon [88] suggested that certain physiological factors impacting interface usage can be combated with a design that has improved illumination and provides certain distinguishing visual cues/feedback to the user. Luminance differences and contrasting shades (e.g., using a gray-scale) have been particularly successful in capturing user attention [4, 103], as well as, in distinguishing objects in medical diagnostic images [52, 109]. This motivated us to adopt a similar approach where our goal is to evaluate whether usability of our RandomPad interface improves when additional visual cues are provided to the users, for instance, by using contrasting shades of gray to represent each of the keys. More specifically, we study an enhancement to RandomPad, where the keys on the randomized keypad are colored with an ascending shade of gray, i.e., shade of key "0" being the lightest (#D8D8D8) and "9" being the darkest (#000000). The luminance of keys between "0" and "9" are increased in uniform steps. Figure 5.5 shows an exemplary instance of IKR keypad in our ascending gray-scale scheme. Note that keys "*" and "#" are excluded from this particular usability study. As the key sequence is preserved in KSR and KLR, we do not expect any potential benefit from the gray-scale enhancement, and thus, are not evaluated either.

5.7 RandomPad - Study

Our study comprised of lab-based experiments involving human subjects, where participants performed typing tasks on a set of assigned smartphone numeric key-

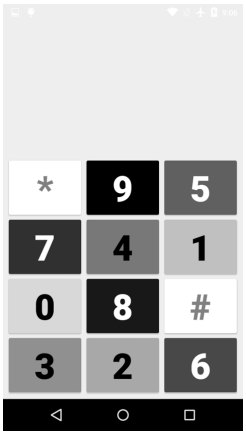


Figure 5.5: Aiding usability with contrast: Instance of IKR keypad in ascending gray-scale.

pads. Each participant is randomly assigned only one of the five keypad randomization strategy, and all experiments administered to each participant are based on the assigned strategy. The assigned strategy is uniformly distributed (balanced) across all participants. The entire experiment for each participant is divided into two sessions: *Natural Typing* and *Dictated Typing*. In each scenario, the participant types using the default, randomized (with the randomly assigned strategy) and gray-scale (assigned only if the participant was assigned RR, CR or IKR) keypads. Our experiment (methodology and data collection process) was reviewed and approved by Wichita State University's Institutional Review Board.

5.7.1 Participants

Our study was conducted by recruiting 100 participants, the majority of whom were affiliated with our university. As an incentive, students were offered participation credits which would partially satisfy certain academic requirements, while

Table 5.2: Demographics and preferences of participants.

Gender	56% Female, 44% Male
Occupation	33% Employed, 67% Student
Smartphone Ownership Duration	26% Less than 5 Years, 74% More than 5 Years
Current Smartphone	59% iOS (iPhone), 1% Android
Willingness to Use Random Keypad (Before Study)	22% In Favor, 78% Not in Favor

non-students were compensated with $10. The participants first completed a pre-survey demographic questionnaire, which included questions on smartphone usage and privacy preferences. Then an introductory video, explaining the concept of randomizing keypads and how it can help protect against certain eavesdropping or side-channel attacks, was shown to them. Participants were then introduced to the specific randomization strategy assigned to them using another video. However, they were not introduced to the remaining (non-assigned) randomization strategies. This was done to prevent any bias in their response(s). Participant demographic information and preferences are outlined in Table 5.2. Interestingly, when shown a sample random keypad screenshot (according to the assigned randomization strategy), less than 25% were in favor of using the random keypad for typing sensitive information. Note that this response was recorded before they were introduced to the side-channel keystroke inference attacks and how randomization can help protect against it.

5.7.2 Apparatus

Our experiments were conducted by using an Android implementation of the RandomPad application, designed specifically for this study. The application would

display the keypad (Figure 5.3) and a short instruction of the task to perform. As the participants type on the keypad, the application records the dictated number (if applicable) along with the typed number and the corresponding time-stamps. The application was also programmed to the flow of our experiments, and it automatically enforced certain aspects of the experiments, such as random ordering of the natural and dictated typing scenarios, rest periods between various parts in each scenarios, pausing to record responses to the NASA-TLX and SUS scales, etc. The order in which the two experimental scenarios (discussed next) are presented to the participants is *counterbalanced* to prevent bias in the results. The keypad design in our application (for each randomization strategy) followed well-accepted standards [53], for example, the smallest height and width of a key was 57dp, which is comfortably higher than the standard minimum of 48dp. We used the Moto E smartphones (1st generation) in our study. The Moto E features a 4.3 inch touch screen with 540×960 pixels ($\sim$ 256 ppi pixel density). We also used the head-mounted Ergoneers Dikablis Professional Eye-Tracking system, equipped with two eye movement tracking cameras and a forward scene camera, to measure participants' eye activity while typing.

5.7.3 Session 1: Dictated Typing (DT)

In this experimental session, participants were prompted with visually and acoustically dictated sequences of pseudo-random single digit numbers. Length of each number sequence was uniformly varied between 3 (representing length of credit card security codes), 4 (representing length of phone unlock codes), 5 (representing length of zip codes), 7 (representing length of phone numbers without area code), 8 (representing length of birth dates), 10 (representing length of phone numbers with area

code), and 16 (representing length of credit card numbers). This session of the study is further divided into three *parts*: *Default Keypad Typing*, *Randomized Keypad Typing*, and *Gray-scale Randomized Keypad Typing*. Each part consisted of ten *activities*, where in each activity the participants typed the dictated sequence of single digit numbers on the displayed keypad. There was a ten second time separation between each activity and a one minute separation between the three parts, allowing participants enough opportunities to rest.

5.7.3.1 Task

The primary task for participants is to follow the dictation and type the dictated digits on the displayed keypad. Each activity begins when participants are ready and they tap on the "Start" button on the smartphone screen. Immediately after tapping the "Start" button, the keypad appears and dictation starts. Participants can see the dictated digits on screen or hear the corresponding audio prompt, or both. No time restriction is imposed on participants, and new digits are dictated after the participant presses a key in response to the last dictated digit.

5.7.3.2 Part 1.1 – Default Keypad

In this part, participants type on the default keypad, with no randomization in key size or sequence. This serves as a reference point for our performance and accuracy evaluation.

5.7.3.3 Part 1.2 – Randomized Keypad

In this part, the RandomPad application generates and displays an instance of random keypad, using the randomization strategy assigned to the participant. For KSR, KLR randomization strategies, a new instance of random keypad is generated and displayed after every key press. For other (RR, CR, IKR) randomization strategies, the instance of random keypad generated at the beginning of each activity is used for the entire activity.

5.7.3.4 Part 1.3 – Gray-scale Randomized Keypad

This part is administered only to those participants who are assigned RR, CR, and IKR strategies. The RandomPad application generates and displays an instance of random keypad, using the randomization strategy assigned to the participant. Additionally, keys on the randomized keypad are shaded with an ascending shade of gray, with color of key "0" being lightest and "9" the darkest, as discussed earlier.

5.7.4 Session 2: Natural Typing (NT)

In this session, participants were instructed to type information already known to them at their own natural pace. Participants were asked to type their residence area code (3 digits), zip code (5 digits), phone number without area code (7 digits), birth date (8 digits), or phone number with area code (10 digits), in random order. This session is also divided into three parts, i.e., Default Keypad Typing, Randomized Keypad Typing, and Gray-scale Randomized Keypad Typing, with each part consisting of ten activities. The time intervals between parts and activities remain the same as before.

5.7.4.1 Task

The primary task for participants is to type familiar numbers on the random keypad. Before beginning each typing activity, the participants are visually communicated about the number they have to type in that activity. The activity begins when participants are ready and tap on the "Start" button on the smartphone screen. Immediately after tapping the "Start" button, the keypad appears and participants are expected to start typing. No time restriction is imposed on participants, and activity finishes when the participants are finished typing in the expected number of digits (based on what was asked to type).

5.7.4.2 Parts

Similar to the dictated typing session (i.e., session 1), the natural typing session (i.e., session 2) has three parts: (i) Part 2.1 – Default Keypad, (ii) Part 2.2 – Randomized Keypad, and (iii) Part 2.3 – Gray-scale Randomized Keypad. The description of the activities performed by the participants and the keypads used in each of these parts is similar to the previous session; the only difference is that rather than typing a dictated sequence of numbers in each activity, the participants type a known sequence of numbers (as outlined before) at their own pace.

5.7.5 Procedure and Data Collection

Participants were seated in a lab environment and given a smartphone installed with the RandomPad application. Before beginning each session of the study, a short video was shown to the participants explaining the task to be completed in each part. If participants made a mistake during typing, they were instructed to continue to

the next number without attempting to rectify it. The mistyping is recorded for evaluating accuracy in typing. To measure accuracy during the Natural Typing session, the residence area code (3 digits), zip code (5 digits), phone number without area code (7 digits) and date of birth (8 digits) were collected from each participant beforehand in the pre-survey. As discussed in sections 5.6, subjective usability and mental workload perceptions of participants is captured using the SUS and NASA-TLX scales. The SUS and NASA-TLX surveys were completed by the participants after each part of either session 1 or session 2, whichever came temporally later (as the order of the Dictated Typing and Natural Typing sessions is counterbalanced). After finishing both sessions of the experiment, the participants completed a post-survey.

5.8 RandomPad - Results

In this section, we outline results from both the natural and dictated typing sessions of our experiments.

Q1: Do randomized keypads increase the task completion time when compared to the default keypad?

We investigate the difference in task completion times between default and randomized keypads with the null hypothesis that their means are not significantly different. Figures 5.6 and 5.7 show the average time taken by the participants to type a key, in the dictated and natural typing sessions, respectively. The results are further categorized by the keypad randomization type. It is evident that the aver-

age task completion time on random keypads is increased in both cases, compared to the default keypad. However, the overall task completion time is less in natural typing, compared to dictated typing. This is most likely due to the extra cognitive task performed to follow the dictation while typing. Among the five randomization strategies, typing on IKR (mean differences w.r.t. default keypad, $d^{\mu DT}_{IKR} = +249.78$ ms, $d^{\mu NT}_{IKR} = +140.66$ ms) and KSR ($d^{\mu DT}_{KSR} = +263.61$ ms, $d^{\mu NT}_{KSR} = +137.07$ ms) took relatively more time compared to CR ($d^{\mu DT}_{CR} = +105.14$ ms, $d^{\mu NT}_{CR} = +98.75$ ms), KLR ($d^{\mu DT}_{KLR} = +116.32$ ms, $d^{\mu NT}_{KLR} = +107.75$ ms), and RR ($d^{\mu DT}_{RR} = +106.60$ ms, $d^{\mu NT}_{RR} = +144.62$ ms). In two-tailed paired sample t-test [100], the combined mean increase in time taken by participants to type a key are $d^{\mu DT} = +168.3$ ms and $d^{\mu NT} = +125.8$ ms, with $p < 0.001$ in both DT and NT. Since $p < 0.05$ (the assumed significance level), the null hypothesis is rejected. In other words, we found that randomized keypads do increase task completion times, by approximately 21% for dictated and 16% for natural typing.

Q2: Do randomized keypads increase the error rate in the primary task?

We investigate the difference in typing accuracy between default and randomized keypads with the null hypothesis that their means are not significantly different. Figures 5.8 and 5.9 shows the average accuracy of the typed numbers, for the natural and dictated typing sessions, respectively. The results are further categorized by the keypad randomization type. In two-tailed paired sample t-test, the combined mean decrease in typing accuracy are $d^{\mu DT} = -0.53\%$ and $d^{\mu NT} = -0.77\%$, with $p = 0.06$ in DT and $p = 0.003$ in NT. As $p > 0.05$ for dictated typing, the null hypothesis is marginally accepted. However, the null hypothesis is rejected in case of natural

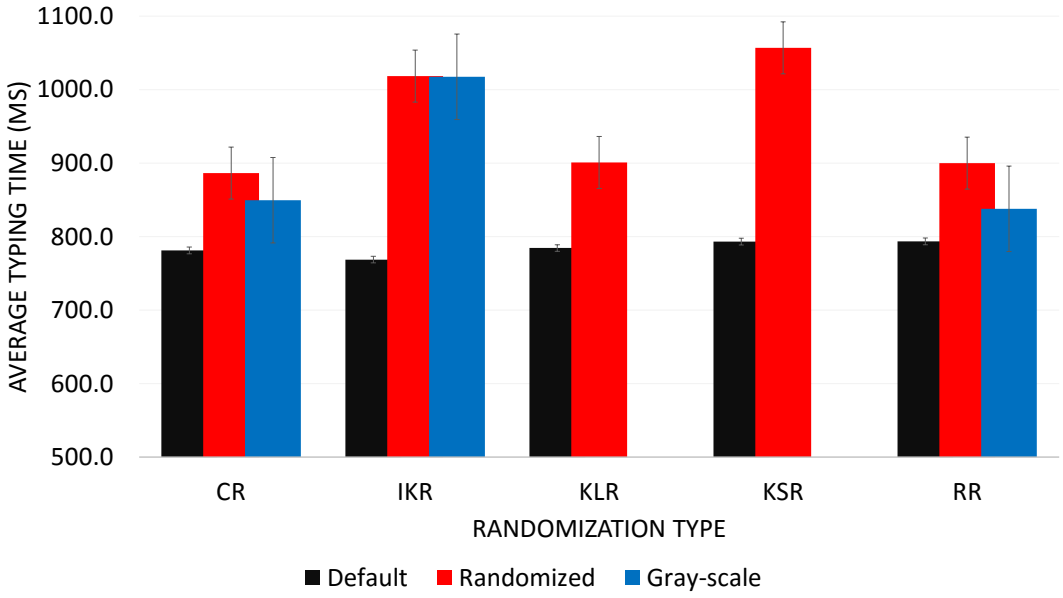


Figure 5.6: Average time taken per key typed in Dictated Typing.

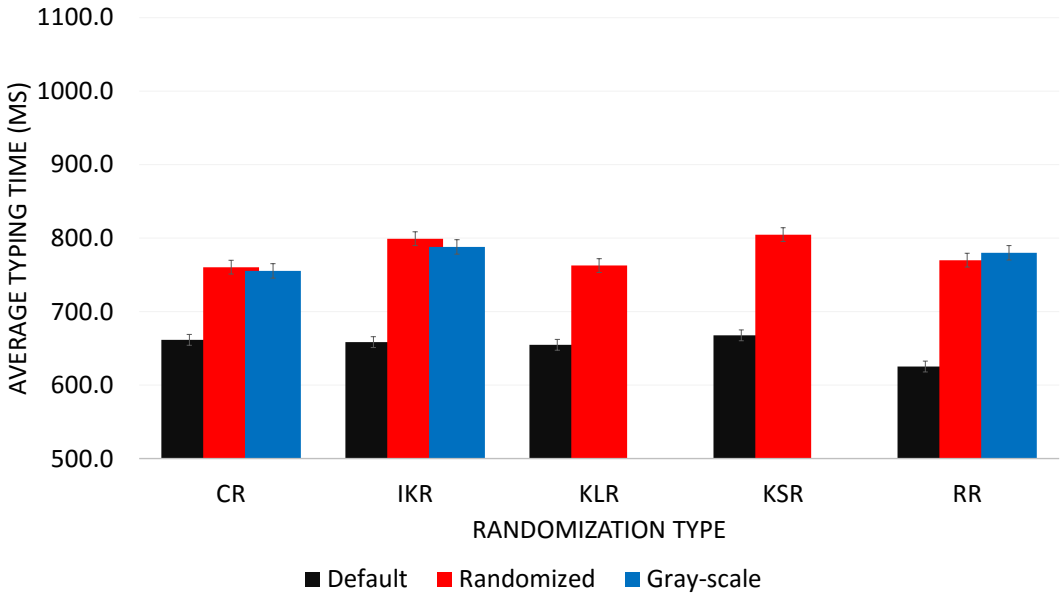


Figure 5.7: Average time taken per key typed in Natural Typing.

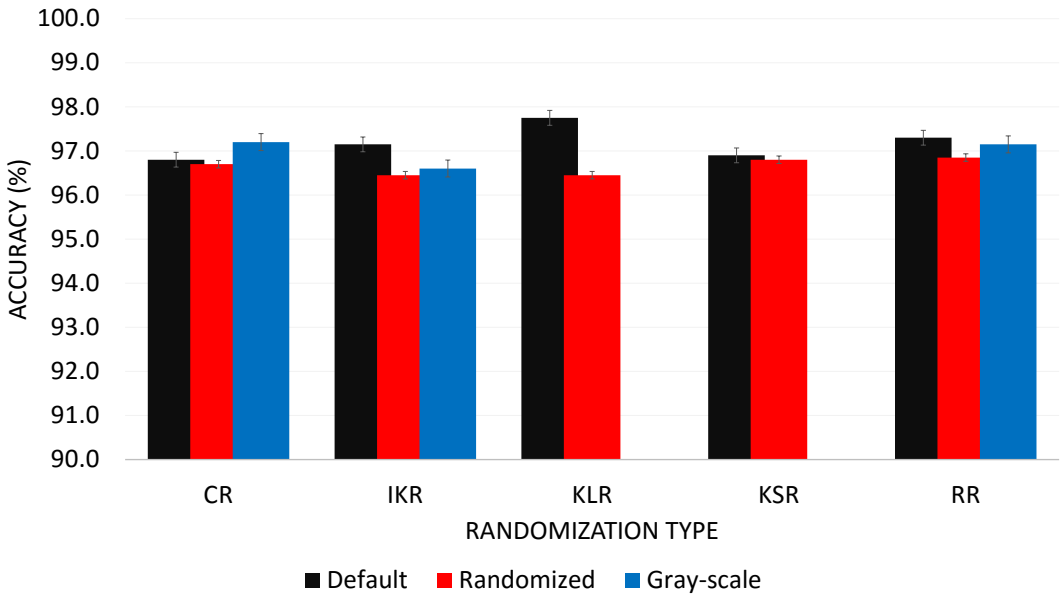


Figure 5.8: Dictated Typing accuracy.

typing, which means the typing accuracy may be lower on the randomized keypads. Nonetheless, mean accuracies in all five randomization strategies are above 95% for both default and randomized keypads, with mean difference less than 1% compared to the default keypads. As the participants' primary task was to type the number sequences correctly, and not to type as fast as possible, it may be concluded that the task completion time was traded off for higher accuracy by the participants.

Q3: Is there a learning curve associated with randomized keypads?

In order to analyze if the typing performance (speed and accuracy) improves with more usage of the randomized keypad, we compare the average per key typing time for the first and last ten numbers typed with random keypads, in the natural typing session. We observed that the average task completion time is significantly

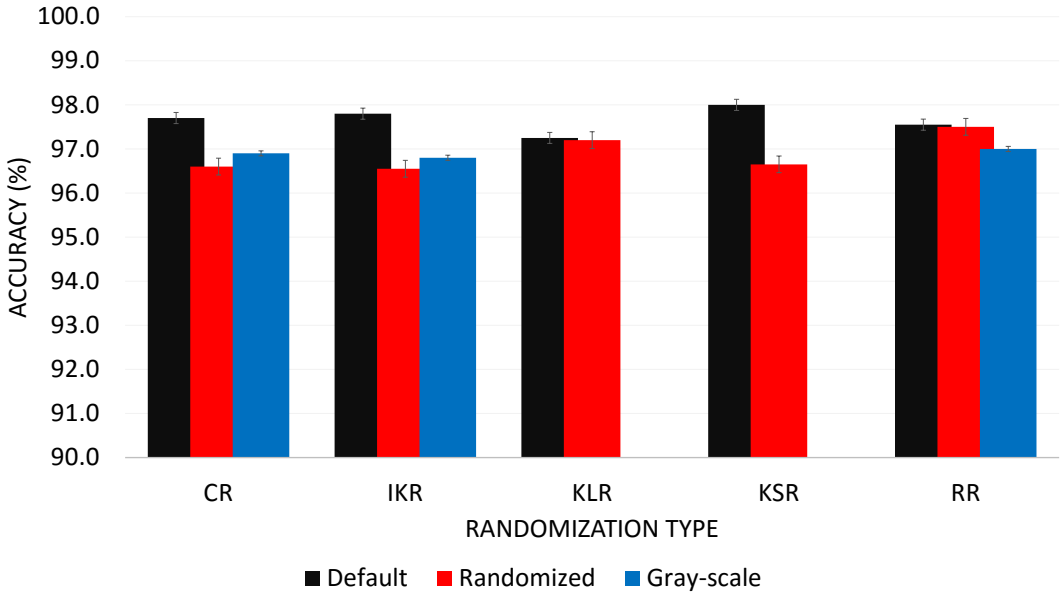


Figure 5.9: Natural Typing accuracy.

lower for the last ten numbers (compared to the first ten numbers), for all five randomization strategies. The overall mean drop in per key typing time is recorded as $d^{\mu NT}_{L10-F10} = -163.09$ ms, with $p < 0.001$. However, we did not observe any significant improvement in accuracy. This suggests that there exists a learning curve in using RandomPad, primarily to learn the randomization strategy, rather than memorizing an instance. As we see marked improvements within a relatively short experimental duration, we are optimistic that randomized keypad usage performance will only further improve with prolonged use.

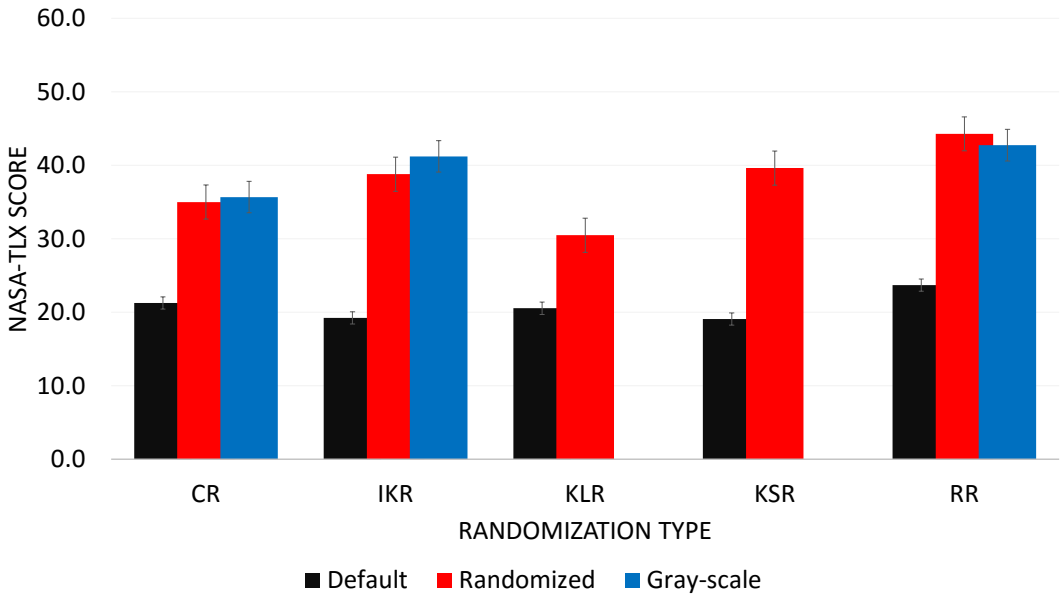


Figure 5.10: NATA-TLX scores for all the five randomization strategies. Dictated and Natural Typing are combined. Lower scores signify lesser workload for the user.

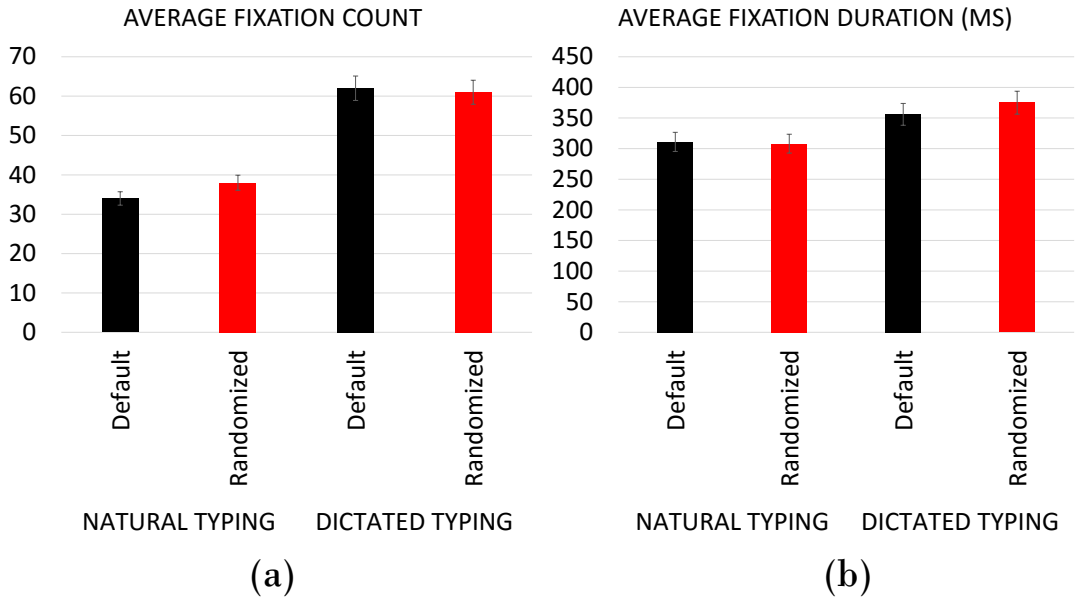


Figure 5.11: (a) Average fixation count and (b) average duration per fixation, for Natural and Dictated Typing. Lower scores signify lesser workload for the user.

Q4: How much more effort do randomized keypads take, compared to the default keypad?

We investigate the difference in user effort required to type on the default versus randomized keypads with the null hypothesis that the mean of their NASA-TLX scores are not significantly different. However, the randomized keypads scored higher on the NASA-TLX scale for all five randomized keypads (Figure 5.10), with an overall mean difference $d^{\mu} = +16.87$ and $p < 0.001$. This suggests that participants required considerably more perceived effort to use the randomized keypads. This observation is somewhat expected because of their familiarity with the default keypad. KLR is reported to take the least effort ($d^{\mu}_{KLR} = +9.94$) compared to the other four randomization strategies.

In addition to subjective metrics such as NASA-TLX, we complement our results for workload by means of quantitative metrics such as fixation counts and fixation duration from the eye tracking data. Due to the significant setup time and overhead involved, we collected eye tracking data from only a subset of participants (more specifically, 53 participants). Figure 5.11 shows the average fixation count and fixation duration recorded during the experiments involving these participants. Higher fixation count indicates that the participants had to view more areas of interest (AOI) before they were able to locate the target key. The average fixation count on randomized keypads is increased (+4) in case of natural typing, but marginally decreased (-1) in case of dictated typing. On the other hand, the average fixation duration (time spent per AOI) increased (+19 ms) in case of dictated typing, but marginally decreased (-3 ms) in case of natural typing. These results show that,

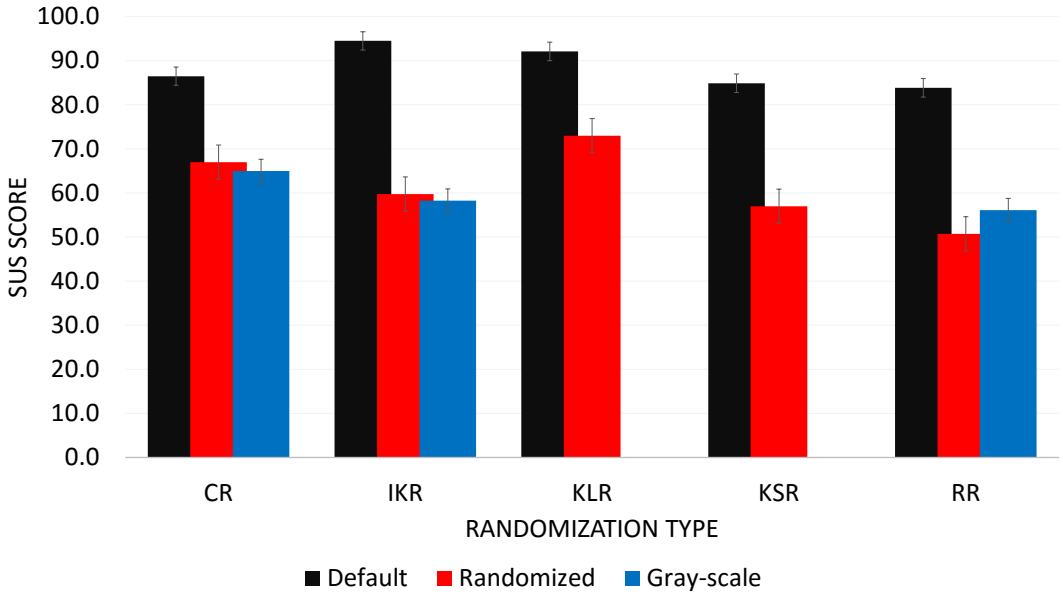


Figure 5.12: SUS scores for all the five randomization strategies. Dictated and Natural Typing are combined. Higher scores signifiy better usability.

in certain scenarios, randomized keypads do increase the difficulty in locating keys, resulting in increased workload.

Q5: How much less usable randomized keypads are, compared to the default keypad?

We investigate the difference in perceived usability of the default versus randomized keypads with the null hypothesis that the mean of their SUS scores are not significantly different. However, the randomized keypads faired lower than the default keypad (Figure 5.12). The overall mean difference $d^{\mu} = -25.80$ and $p < 0.001$ suggests that participants felt that the default keypad is more usable. This observation is also somewhat expected because of their familiarity with the default keypad.

KLR is again reported to be the most usable ($d^{\mu}_{KLR} = -19.12$) compared to the other four randomization strategies.

In the SUS scale, a score lower than 50 indicates unacceptable for use; a SUS score larger than 70 indicates acceptable for use; A score between 50 to 70 indicates marginally usable. RR (mean SUS score = 56.13) is significantly lower on the SUS scale compared to the other five randomization strategies. On the other end, KLR (mean SUS score = 73) provides acceptable usability and CR (mean SUS score = 67) is just below the 70 mark. Thus, from a perceived usability perspective, KLR is preferred by the users over other randomization strategies. Again, we are optimistic that randomized keypads can achieve better usability scores if users get familiarized with the distinct layouts.

Q6: Does gray-scale shading of randomized keypads improve usability?

On the NASA-TLX (Figure 5.10) and SUS (Figure 5.12) scores, there are no significant differences between the randomized keypads without gray-scale shading versus randomized keypads with gray-scale shading, indicating that contrasting gray-scale shades on the keypad does not lower the perceived workload or improve the perceived usability. However, we observe that gray-scale shaded randomized keypads marginally lower the task completion times. The average task completion time on gray-scale randomized CR, IKR and RR keypads during dictated typing session is 901.70 ms versus 935.09 ms for just randomized CR, IKR and RR keypads. This shows that our gray-scale shading scheme does not significantly improve the usability of the RandomPad interface. This may be because in our gray-scale shading scheme the contrast between the shade of the digits and that of the background is not optimal

for certain keys, which can create difficulty during the reading of those keys [129]. The usability of gray-scale keypads could be potentially improved by adjusting and optimizing this contrast between the different shades.

Q7: Are smartphone users interested in adopting randomized keypads?

In the initial pre-survey recorded before the participants were introduced to side-channel keystroke inference attacks, only 22% of the participants reported that they would be willing to use a randomized version of the keypad. On being more informed about the dangers of side-channel keystroke inference attacks and how randomized keypads help protect against such attacks, and after completing the experimental trials, as many as 80% of the participants reported in the post-survey that they would be willing to use a randomized keypad in order to protect their privacy. Those participants who reported "No" to this question (i.e., were not willing to use the randomized keypad) in the post-survey reported that they were more familiar, therefore more comfortable, with the default keypad and that the randomization (of the keypad) was confusing to them. Those who changed their answer to "Yes" in the post-survey (i.e., were willing to use the randomized keypad) primarily reported that their reason for using an unfamiliar interface, such as, a randomized keypad, would be to primarily enhance their privacy and to prevent hackers from stealing their personal information.

5.9 RandomPad - Discussions

In this section, we discuss some of the implications of our study, and how researchers and developers can use our evaluation results in order to implement and/or improve RandomPad.

5.9.1 Privacy-Usability Trade-Off

In our evaluation, it was clear that RandomPad negatively affects users' performance, workload and perceived usability. While this was intuitive and an expected result, the effect on usability, although present, was not large enough to make the interfaces completely unusable. It should also be noted that 80% of the participants were still willing to use RandomPad on a regular basis in order to input their sensitive information. As participants were introduced to only one randomization strategy (to receive an unbiased opinion about each strategy), it is also likely that they may like another strategy better. Therefore, we analyzed the privacy-usability trade-off of the five different randomization strategies based on security assurance ranking (Table 5.1) and usability ranking (calculated using typing speed, workload and perceived usability)[1]. Table 5.3 shows the usability ranking calculations of the five different randomization strategy. Comparing Table 5.1 and 5.3, we see that KLR ranks relatively highest on both (3 + 1 = 4) tied with IKR (1 + 3 = 4), followed by RR (2 + 4 = 6), and CR (5 + 2 = 7), and KSR (4 + 4 = 8), respectively. In other

[1]As accuracy was marginally varying, we exclude it as a factor in the usability ranking calculations.

Table 5.3: Usability rankings of the five proposed randomization strategies calculated using average typing speed (lower better; dictated and natural typing combined), workload (lower better) and perceived usability (higher better). Lower least rank is better usability.

Randomization Strategy	Typing Speed Rank	Workload Rank	Perceived Usability Rank	Summed Usability Rank (Least Rank)
CR	1	2	2	5 (2)
IKR	4	3	3	10 (3)
KLR	2	1	1	4 (1)
KSR	5	4	4	13 (4)
RR	3	5	5	13 (4)

words, KLR and IKR provides the best balance between security and usability, while KSR provides the least.

5.9.2 Recommendations to Developers

Users type sensitive information only a fraction of the time they use a keypad. Having an always-on randomized keypad may be an inconvenience to the users, whom may then choose to not use randomized keypads altogether. A good design should have an easily accessible and user-controllable (soft) switch to turn on or off the key randomization, as and when desired by the user. Whenever users feel that the information they are going to type is sensitive in nature, they should be able to easily turn on the randomization of the keypad. After they finish typing the sensitive information, or in the case they are typing non-sensitive information, they should be able to easily turn off the randomized keypad and continue to use the default keypad.

5.9.3 Limitations

Even though RandomPad is able to protect users against several types of side-channel keystroke inference attacks, it fails to protect against visual eavesdropping, also known as *shoulder-surfing*. There are certain authentication schemes that can defend against visual eavesdropping [119], but they (i) require more effort from users and (ii) cannot be used to type sensitive information other than device unlock codes.

5.10 RandomPad - Conclusion

With the increasing number of side-channel attacks targeting mobile keypads, user privacy is at stake. With RandomPad, we proposed to used randomized keypads for typing sensitive information on mobile device keypads. Randomized keypads are able to sufficiently alter keystroke characteristics, such that most of the side-channel attacks will fail. However, with users accustomed to the default keypad for years, randomized keypads face usability issues. Therefore, we comprehensively evaluate the usability of randomized keypads, with the help of 100 participants. We found that randomized keypads can increase task completion time. We also found that randomized keypads are perceived to be less usable and more work. However, the learning curve associated with randomized keypads can improve user performance and usability with prolonged use. Interestingly, even with the degraded usability of randomized keypads, participants were willing to use it for improved privacy.

5.11 EyePad - Protecting External Keyboard Interactions

Physical QWERTY keyboards are the most widely adopted input interface for personal and portable computing systems. These keyboards have also been a constant target for various forms of "*shoulder surfing*" attacks, where the goal of an adversary is to obtain or infer users' keystrokes by directly, but surreptitiously, observing the typing user (and the keyboard) or eavesdropping on certain information directly related to the typing activity being performed. The first case, where the adversary has a covert visual access to the typing user, is a more common and easy-to-execute threat. Such threats are also the most difficult to protect against, especially by means of traditional cryptography-based or other information manipulation and hiding techniques. For instance, Roth et al. [92] proposed an oracle-based multi-round protocol for PIN entry by color coding keys into two shades (black and white). This scheme takes advantage of limitations in human cognitive capabilities to overcome shoulder surfing, however, Kwon et al. [64] recently showed that covert attention and perceptual grouping can improve information processing by humans, thus rendering Roth et al.'s approach ineffective.

Alternatively, there exists other forms of shoulder surfing attacks that, rather than relying on the direct visual channel, take advantage of indirect information channels (or side-channels) to infer users' keystrokes. For instance, Vuagnoux et al. [113] use electromagnetic emanations from external keyboards (both wired and wireless) to infer keystrokes, whereas, Berger et al. [16] have accomplished a similar feat by using acoustic emanations originating due to typing on these keyboards. As another variation of non-visual shoulder surfing, Marquardt et al. [77] utilized

the vibrations sensed by a smartphone accelerometer (positioned in the proximity of the target keyboard) to infer a users' keystrokes on the keyboard. Maiti et al. [71] proposed a similar attack by taking advantage of motion information available from wrist-wearable devices such as smartwatches. More recently, Ali et al. [6] demonstrated the ability to infer keystrokes by observing the unique changes in the radio signal channel statistics caused during typing.

Interestingly, the success of all of the above attacks rely on one common assumption: the adversary has knowledge of the layout, and in some cases, even the exact model, of the keyboard used by the target user. This assumption, at least the former, is reasonable as most modern QWERTY keyboards have a standard layout of keys. Intuitively, this means that if the keyboard layout is changed from the default to something different, and if this new or changed layout is not known to the adversary, then at least the above side-channel or non-visual attacks will not succeed. In other words, a dynamic keyboard layout strategy is an appealing defense strategy against side-channel keystroke inference or shoulder surfing attacks. Such a strategy is also not far-fetched as a similar concept is currently being used in other types of commercial products, for instance, to enhance the security of electronic door access control systems [101]. Ryu et al.[93] also performed a usability evaluation of such randomized numeric keypads.

Despite the promise, there are two critical technical challenges with respect to implementing this solution for external or physical QWERTY keyboards. First, the layout of these keyboards cannot be easily modified; it is possible to modify the mapping between the physical keys (on the keyboard) to the actual character they

represent, however such a keyboard will be extremely challenging to use as the users will have to memorize the mapping between the physical keys on the keyboard and the actual characters they represent. Second, even if somehow it was possible to dynamically change the physical layout of the keyboard, such a change would not protect against shoulder surfing attacks by an adversary that has covert visual access of the target keyboard (or the user typing on the keyboard).

With EyePad, we overcome the above technical challenges and propose a system for randomizing external keyboard layouts by making a novel and interesting use of augmented reality devices. One key advantage of our proposal is that it is able to overcome all forms of shoulder surfing attacks, including those possible through direct visual access of the target keyboard. Our proposal consists of several keyboard layout randomization strategies, each of which assigns a unique non-standard position to the keys on the keyboard which is unknown to the adversary. The randomized keyboard is then projected to the typing user by means of an augmented reality wearable device. As the randomized keyboard is visually superimposed over the actual physical keyboard, and is visible only to the typing user through the augmented reality device, it acts as an effective countermeasure against both side-channel and visual channel-based keystroke inference or shoulder surfing attacks. We implement our system on the commercially available EPSON Moverio BT-200 [31] augmented reality device and validate its performance and effectiveness by means of preliminary empirical usage data from a small number of test subjects.

5.12 EyePad - Related Work

Protection against shoulder surfing attacks have received significant attention in the literature, with several different solution directions proposed and analyzed. For instance, Kumar et al. [62] proposed EyePassword, where orientation of the user's pupils were used for password entry. The authors further showed that such an approach requires only marginal additional time over using a keyboard and that the error rates due to this approach is similar to those of using a keyboard. In order to thwart shoulder surfing attacks against traditional alphanumeric passwords, graphical passwords was also proposed as an alternative where the user was expected to select a predetermined image or set of images in a particular order [87, 55]. Human subject studies showed that such graphical passwords were perceived to provide reasonable protection against visual shoulder-surfing attacks [106], however it was later showed that those conclusions were not completely valid [125, 65]. More recently, Yan et al. [119] proposed CoverPad, a leakage-resilient password entry system for touchscreen mobile devices, where a user is expected to cover the touchscreen (by hand) to securely read a hidden message that contains information on removing the correlation between the actual password (or PIN) and the one entered by the user. One common theme in most (if not all) past research efforts in this direction is that they focus only on preventing shoulder surfing attacks against authentication information such as passwords or PINs. Our proposed design and system protects all kinds of textual inputs, including, but not limited to, passwords and authentication information, against both visual and side-channel shoulder surfing attacks.

5.13 EyePad - Adversary Model

We consider the scenario of a target user typing on an external or physical QWERTY keyboard and an adversary who intends to carry out a shoulder surfing attack on the user in order to infer his/her keystrokes. The attacker may carry out the shoulder surfing attack using various channels. He may have a covert visual access of the physical keyboard and the user's typing activity. This could be achieved by the adversary surreptitiously watching the target user's keyboard as he is typing or by gaining access (either legally or in an unauthorized fashion) to a video feed of the user's keyboard and typing activity by means of a camera or a surveillance device. We assume that the information being typed is protected from visual eavesdropping of the display screen (or monitor). This is a reasonable assumption as most applications obfuscate confidential on-screen information or text such as passwords and PINs by symbols or special characters (e.g. asterisk). Alternatively, the monitor could also be protected using a privacy screen. It should be noted that these measures do not protect against an adversary eavesdropping on the keyboard and user's keystrokes. If a visual channel is unavailable to the adversary, he may attempt to accomplish the keystroke inference attack using other forms of information side-channels, such as, electromagnetic emanations from the keyboard [113], vibrations [77] or acoustic [16] signals captured during the typing activity, by observing the changes in the radio signal channel statistics [6] or by capturing the motion information of the typing hand [71]. As discussed later, our protection mechanism involves the use of commercial off-the-shelf augmented reality glasses such as EPSON BT-200. We assume that the

display of this augmented reality device is visible only to the target user, and that this device is secured from the adversary.

5.14 EyePad - Proposed Defense Model

Consider the scenario where a user wants to type a sensitive piece of information on an external QWERTY keyboard in the presence of an eavesdropping adversary, as shown in Figure 5.13. To obscure keystrokes from the eavesdropping adversary, we propose the use of randomized keyboard layouts in cohort with an augmented reality device. In our proposal, the user privately sees a randomized (using strategies explained later) keyboard layout augmentation over the actual keyboard, where keys are positioned differently from the default QWERTY layout, by means of an augmented reality device or glasses (shown in Figure 5.13). The augmentation is done such that the randomized keys are superimposed over the existing keys of the physical keyboard. This can be achieved with the help of marker (Figure 5.15) or character recognition [2] of individual keys on the physical keyboard. Also, the augmented reality device establishes a temporary secure wireless link with the computer (with which the keyboard is attached to) so as to communicate the key mapping between the randomized augmented layout and the underlying QWERTY layout. This secure link can be established using widely available wireless technologies, such as Bluetooth, and made secure using symmetric encryption protocols, such as AES.

Whenever the user presses a key based on observation of the augmented layout, the computer uses the mapping between the randomized and QWERTY layouts to substitute the character typed on the physical keyboard with the correspondingly

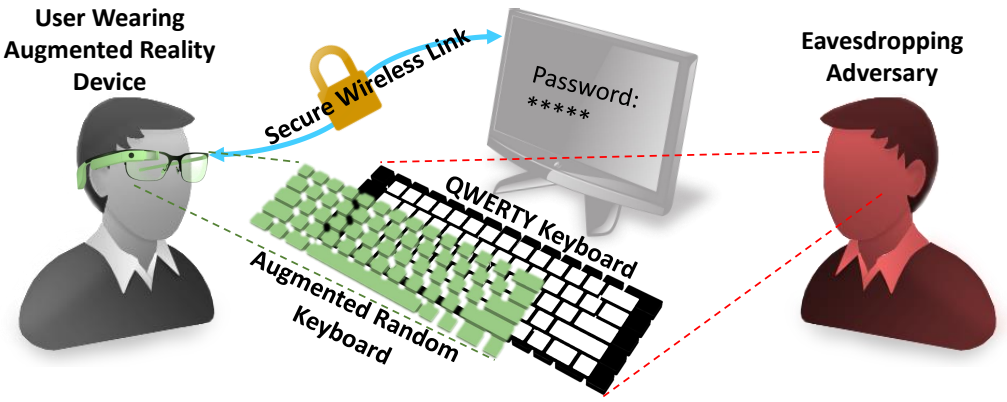


Figure 5.13: The proposed defense model, where the user wearing the augmented reality device sees and types on the randomized augmented keyboard. The eavesdropping adversary can observe only the default QWERTY layout of the physical keyboard.

placed key in the augmented layout. The adversary, however, can only eavesdrop on the physical keyboard having the default QWERTY layout. As the adversary does not see (or is unable to eavesdrop on) the augmented layout and does not have access to the mapping, it cannot infer the character actually registered by the computer system.

5.14.1 Randomization Strategies

To prevent keystroke inference attacks, an important task in the proposed system is to ensure that the layout of the augmented characters is unpredictably different from the default QWERTY layout. Moreover, as an adversary can gain semantic knowledge from multiple observations and re-train his attack framework, changing the augmented keyboard layout just once (or in a very predictable or insignificant fashion) will not be an effective defense. To prevent an adversary from knowing the

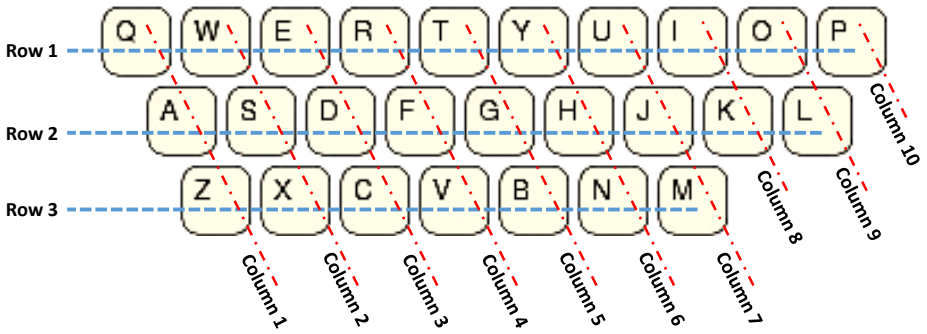


Figure 5.14: Assumed rows and columns for RS and CS strategies.

keyboard layout in use at any given time, the change in layout should be *randomized*. Accordingly, in our proposed system, every time the user wants to type sensitive text, a newly randomized keyboard layout is augmented over the physical keyboard. The new mapping of the randomized layout to the underlying physical keys is also updated accordingly on the computer side by means of the secure communication link. In EyePad, we focus on randomization of just the twenty-six alphabets (Figure 5.14), however it could be easily extended to all keys. Below, we list a few representative (by no means an exhaustive list) randomization strategies that can be used to change the keyboard layout:

(i) Individual Key Randomization (IKR): This strategy randomly assigns positions to each alphabet or letter on the augmented keyboard layout, without any relation to its actual position on the QWERTY layout. An instance of IKR is shown in Figure 5.16.

(ii) Row Shifting (RS): In this strategy, the alphabets in each row of the QWERTY layout (rows in Figure 5.14) are circularly left or right shifted by a random number of keys on the augmented layout. In other words, each alphabet on the

Figure 5.15: A QWERTY keyboard with alphabetic markers glued on top of the corresponding alphabet keys. As a result, the keyboard can be used both in the regular QWERTY or with the random augmented layout.

Figure 5.16: Randomized augmented keyboard using the IKR strategy, as observed by the typer on the EPSON Moverio BT-200.

Figure 5.17: Randomized augmented keyboard using the RS strategy, as observed by the typer on the EPSON Moverio BT-200.

Figure 5.18: Randomized augmented keyboard using the CS strategy, as observed by the typer on the EPSON Moverio BT-200.

augmented layout is found on the same row as in the QWERTY layout, however its position is shifted left or right by a random number of keys.

(iii) Column Shifting (CS): In this strategy, the alphabets in each column of the QWERTY layout (columns in Figure 5.14) are circularly top or bottom shifted by a random number of keys on the augmented layout. In other words, in CS each alphabet on the augmented layout is found on the same column as in the QWERTY layout, however its position is shifted top or bottom by a random number of keys. As the column (correspondingly, row in RS) of each alphabet and the order of alphabets in each column (correspondingly, row in RS) is maintained in CS, intuitively it appears that it may be comparatively easier for a user to search for an alphabet on the CS and RS layouts. We want to validate if this is true in practice, and thus the reason for choosing these two layouts in addition to IKR.

While several additional randomization strategies can be envisioned, for conciseness we limit the current discussion to just the above three strategies.

5.14.2 Security Analysis

As the keyboard layout is randomized, the best an adversary (assumed to know the randomization strategy used by it's target) can do is guess the mapping between the randomized and QWERTY layouts. We use the successful guessing probability to indicate the level of security assurance each randomization strategy provides in the presence of an eavesdropping adversary. For a particular randomization strategy, the lower this probability is, the higher the security assurance provided by it.

In IKR, the probability that an adversary correctly guesses the mapping of a particular alphabet is $\frac{1}{26}$, i.e., uniformly distributed. Moreover, the probability that

the adversary guesses the entire mapping correctly is $\frac{1}{26!} = 2.4 \times 10^{-27}$, which is negligibly small. However, in case of RS and CS, the adversary can improve it's guessing, based on the relative positioning of key within a row and column, respectively. Knowing that keys within a shifted row remain in (circular) order, for a row shifted keyboard (RS), the adversary only needs to guess the random length of shifting. The probability that an adversary correctly guesses the length of a row's shifting is $\frac{1}{10}$, $\frac{1}{9}$, and $\frac{1}{7}$, for rows 1, 2, 3, respectively (as labeled in Figure 5.14). Therefore, the probability that the adversary guesses the mapping for all 26 alphabets correctly is $\frac{1}{10} \times \frac{1}{9} \times \frac{1}{7} = 1.5 \times 10^{-3}$.

Similarly, for CS, the probability that the adversary correctly guesses the length of random shifting is $\frac{1}{3}$ for columns 1 to 7, $\frac{1}{2}$ for columns 8 and 9, and 1 for column 10 (as labeled in Figure 5.14). Therefore, the probability that the adversary guesses the mapping for all 26 alphabets correctly is $(\frac{1}{3})^7 \times (\frac{1}{2})^2 \times (1)^1 = 1.1 \times 10^{-4}$. Thus, given the adversary knows the strategy being used, IKR is probabilistically the most secure while RS is the least secure randomization strategy. However, in practice the adversary will not know the randomization strategy currently in use, thus making these strategies even more secure. The security of the system could be further improved by re-randomizing or reshuffling the keyboard at regular intervals by using a particular randomization technique (and parameters). However, if the keyboard layout is changed too often, the usability may suffer drastically, because the keyboard will change even before users get habituated to the current one. This trade-off between security and usability is what we intend to study by means of experiments involving human participants.

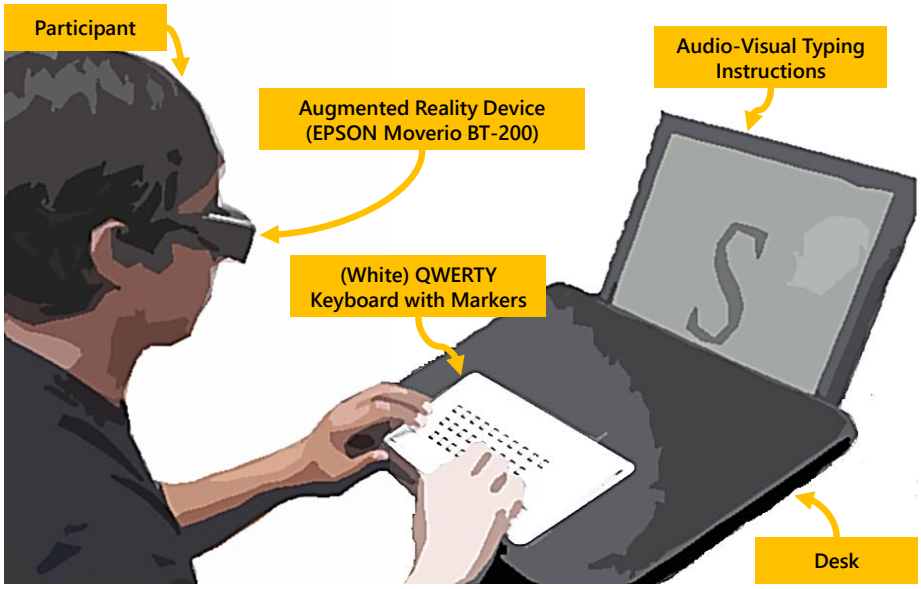


Figure 5.19: The experimental setup, where a participant is typing on the randomized augmented keyboard.

5.15 EyePad - Evaluation

To validate the feasibility of the proposed system, we implement a proof-of-concept prototype and perform preliminary experimentation to evaluate system efficiency and performance parameters such as task completion times and typing accuracy. Next, we first present our prototype and experimental setup followed by results from our evaluation.

5.15.1 Study Design

We perform some preliminary evaluation of our proof-of-concept implementation with the help of data collected from human participants who use our prototype for

typing. Below, we specify our experimental setup, tasks performed by the participants, and the empirical parameters used in the evaluation.

Experimental Setup: Figure 5.19 depicts the setup used in our evaluation. We recruited thirteen participants; all of them were familiar with typing on a QWERTY keyboard. The participants were seated in front of a keyboard, with a display screen in the background. We chose to use the Anker A7726121 Bluetooth keyboard because of its generic design. The keyboard was connected to the computer and the alphabet keys were covered with corresponding alphabetic markers (Figure 5.15). As a result, the keyboard was usable even as a regular QWERTY keyboard. Participants wore the EPSON BT-200 augmented reality device during the experiment. The EPSON BT-200 is equipped with a front facing camera with a resolution of 640×480 pixels, which enables augmented reality applications. The BT-200 also features the Android 4.1 platform, and our implementation of the augmented randomized keyboard was installed as an application. Our implementation of the augmented randomized keyboard uses the ARToolKit library [59]. We would like to stress, however, that in practice a specialized and expensive AR hardware, such as, the EPSON BT-200, is not required. We have also implemented an alternate smartphone application of our proposed augmented randomized keyboard which can be installed by users on their AR-friendly smartphones and used in conjunction with an affordable augmented reality viewer such as Google Cardboard.

Task: The participants were directed with audio-visual instructions on what to type on the keyboard. In the first part of the experiment, each participant typed all twenty six alphabets of English language in random order. In the second part of

the experiment, each participant typed five familiar words: first name, last name, hometown, address street, and area of work. In the third part of the experiment, each participant typed an experimental password of their choice. For the second and third parts, ground truth was collected beforehand, in order to calculate typing accuracy. Participants repeated all three parts of the experiment four times; in default QWERTY (without the augmented randomized keyboard turned on), IKR, CS, and RS. The default QWERTY typing serves as a base line to compare results obtained in the other three scenarios, where participants type using the augmented randomized keyboard. The order of the four typing scenarios was counterbalanced across participants [13], so as to minimize the chances of order effects. For consistency, the same instances of randomized keyboards (each for IKR, CS, and RS) are used by all participants. Participants were also given practice sessions before each part of the experiment, in order to allow them to get familiarized with the keyboard being used.

Empirical Parameters: In order to evaluate our implementation, we measure two usage-related parameters for each participant. For evaluating efficiency, we measure the participants' *typing speed* both on the standard QWERTY layout and on the proposed randomized layouts. Typing speed is measured as the average typing time (in seconds) per character for all the 100 typed characters. We use the computer's clock to log these time intervals. For evaluating performance, we measure the participants' *typing accuracy* for both the standard QWERTY and the randomized layouts. Typing accuracy is measured by enumerating the number of errors during typing by comparing each character instructed to be typed with the character actually typed by the participant. In addition to usage-related parameters, we also

measure the users' perceived workload by using a standard metric such as NASA Task Load Index (NASA-TLX) [41].

5.15.2 Results

We outline results and observations from our experiments below.

Typing Speed: The average time taken by all thirteen participants to type a key on the default QWERTY keyboard (with augmentation turned off) was 2.03, 1.80, and 2.37 seconds for random letters, familiar words, and password, respectively. Readers should note that this measurement includes the time taken by participants to hear/see the alphabet to type, search of the corresponding alphabet on the keyboard, and then key it. When the randomized keyboard augmentation was turned on with the IKR randomization strategy, the average time taken by the thirteen participants to type a key increased to 3.13, 3.15 and 3.36 seconds, respectively. Following a similar trait, in cases of CS and RS randomization strategies, the mean time taken by the thirteen participants to type a key increased (with respect to the QWERTY layout) to 2.58, 2.93 and 3.20 seconds, and 2.94, 2.84 and 3.19 seconds, respectively. Averaged results from each typing scenario are presented in Figure 5.20. These results suggest that there is a notable increase in task completion time with the use of randomized augmented keyboards. As mentioned by some of the participants who are habitual with touch-typing, significant time was used up in searching for particular alphabets on the randomized (IKR) keyboard. A noteworthy observation is that the typing speed is slightly higher on keyboards randomized with RS and CS strategies, compared to IKR. Intuitively, this is due to the fact that a subset of

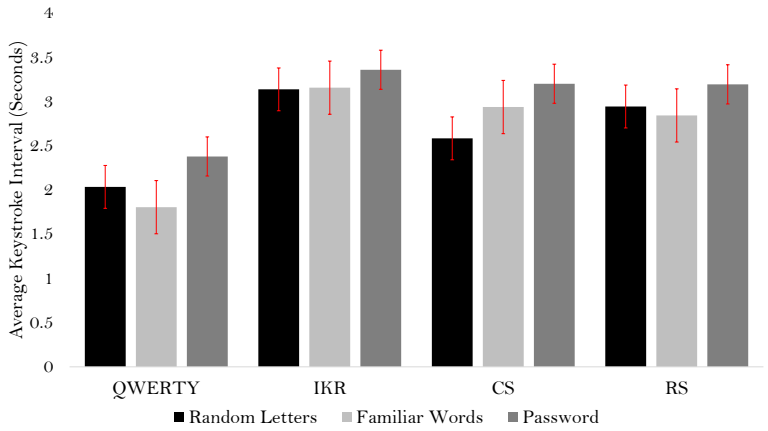


Figure 5.20: Average time taken by the thirteen participants to type random letters, familiar words, and password, using default QWERTY, IKR, CS, and RS layouts.

keys stay *relatively* in the same position as on the QWERTY layout. Therefore, it somewhat eases the process of key search.

Typing Accuracy: The average typing accuracy for all thirteen participants in typing a key on the default QWERTY keyboard (with augmentation turned off) was 94.37%, 93.78%, and 99% for random letters, familiar words, and password, respectively. When the randomized keyboard augmentation was turned on with the IKR randomization strategy, the average accuracy for all thirteen participants in typing a key dropped marginally to 93.19%, 93.19%, 98.53%, respectively. However, typing accuracies in CS (92.89%, 94.08%, 98.53%) and RS (93.78%, 94.37%, 97.76%) randomization strategies were similar to the QWERTY keyboard, if not better. Averaged results from each typing scenario are presented in Figure 5.21. After the experiment was completed, one of the participants expressed concerns about the lag in rendering of the keys, especially noticeable when the user moves his/her head.

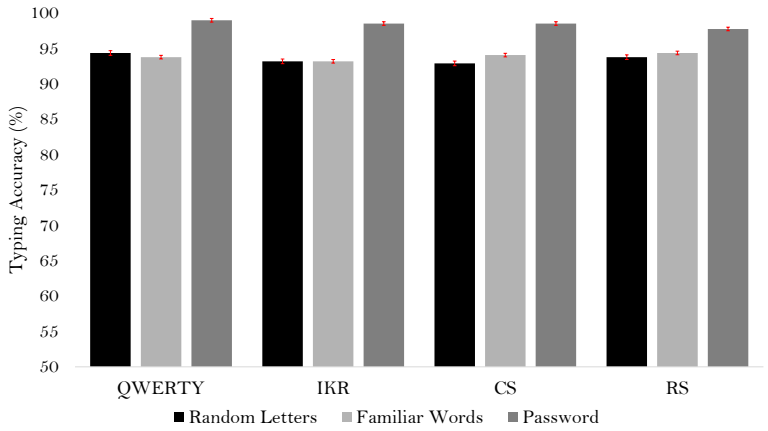


Figure 5.21: Average typing accuracy achieved by the thirteen participants to type random letters, familiar words, and password, using default QWERTY, IKR, CS, and RS layouts.

The delay in rendering may have confused the participants, and lead to longer task completion times and/or more errors in typing. Therefore, results suggest that if some of the issues with our proof-of-concept prototype are resolved, typing accuracy can be comparable to typing on default QWERTY keyboards. Readers may notice that password typing took the longest and was also more accurately typed than the random letters and familiar words. This occurrence is primarily because the participants had to carefully recall and type the experimental password (chosen at the beginning of the study), which most likely is not their real password.

Perceived Task Load: The NASA-TLX is a multidimensional scale to measure the perceived workload, including, the mental, physical and temporal demand, overall performance, frustration level and effort. We employ this scale in our experiments to capture the task load imposed on participants in using the augmented random keyboard. Figure 5.22 shows the average overall score as well as the six individual

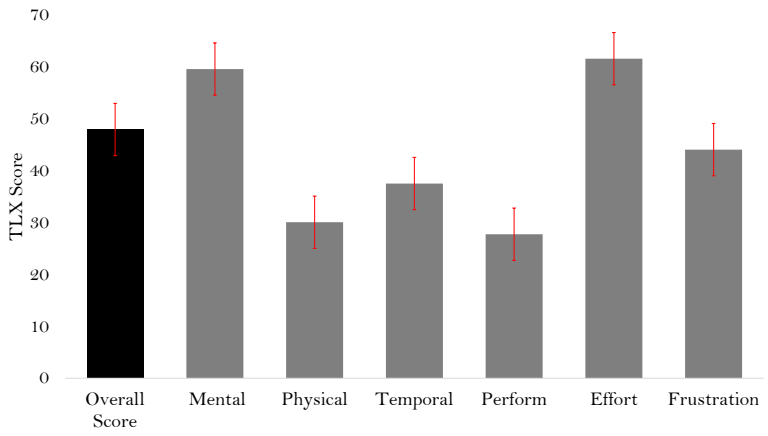


Figure 5.22: Results from the NASA-TLX assessment, taken by participants after completing the study.

sub-scales. Using augmented random keyboard was perceived by participants to be mentally demanding and complex (59.61 - Mental). Participants also felt that the task required significant effort to accomplish (61.61 - Effort). Participants were also not entirely satisfied with the performance of our implementation (27.76 - Perform). However, the physical activity required and time pressure felt due to the pace at which the tasks were being completed are notably low (30.07 - Physical, 37.53 - Temporal). Participants felt moderately content, relaxed, and complacent during the task (44.07 - Frustration).

5.16 EyePad - Discussion

Generalization to Other Keyboards: One advantage of our proposed design is that it can be easily generalized and deployed across different types of keyboards/keypads. The use of character recognition, instead of the exemplary marker

recognition used in our prototype, will enable such a generalized design. One application of such a generalized design can be found in systems such as ATM machines. Numeric keypads on ATMs, due to their open or unrestricted locations, are the most prone to shoulder surfing attacks. The proposed system could be used in this scenario, where a users' augmented reality device could communicate with the ATM by means of a secured wireless channel to exchange a per-transaction randomized layout. This layout can then be augmented over the actual numeric keypad of the ATM machine and made visible only to the user by means of his/her augmented reality device.

Hardware Limitations: The hardware and software of the augmented reality device plays a crucial role in the design and implementation of the proposed system. For example, the camera resolution of the EPSON BT-200 is extremely low (640×480 pixels), which makes marker recognition error-prone and difficult, especially if the user is at a distance from the keyboard (and the markers). We were also restricted by the limitations of the processor on the EPSON BT-200 which resulted in a noticeable lag in rendering when the user moved his/her head. We are hopeful that these limitations will be resolved with advances in augmented reality device technology.

Usability: As evident from our preliminary evaluation, typing on a randomized augmented reality keyboard requires some extra time and effort from the user. The next logical advancement in this direction would be to conduct a comprehensive usability study with the help of a significant number of participants, natural typing experiments, and standard usability metrics, such as SUS [19].

5.17 EyePad - Conclusion

We proposed EyePad, a novel technique to overcome various forms of shoulder surfing attacks against a user typing on an external physical QWERTY keyboard. Our proposal augments a randomized key layout, unknown to the adversary, over the actual QWERTY keyboard, which only the typing user can see by means of an augmented reality device. Our preliminary experimentation involving three different randomization strategies showed that keyboard randomization and augmentation does increase the time required by users to complete their typing tasks. In certain cases, it also introduced additional errors during typing. Despite its promise, these issues along with the usability of the proposed system requires further investigation.

Parts of this chapter appeared in [72, 76].

CHAPTER 6
PROTECTING USER INTERACTIONS: RUN-TIME

6.1 Introduction

As evident from our experimental results in Chapters 2, 3 and 4, the threat to privacy posed by side-channel attacks using wearable devices is substantial. However, there have been very limited efforts from the research community to effectively defend against such side-channel attacks in a user-friendly fashion. None of the recent works on side-channel keystroke inference attacks propose or implement a practical protection mechanism. Some of the previous work using smartphone sensors as side-channels, briefly suggest operating system developers to provide users with fine-grained control over application's permissions to every sensors [23]. But without knowing which application is malicious, the user may have to toggle sensor access back and forth for all the installed applications. Other research efforts vaguely suggest to restrict the precision at which applications are allowed to access the sensors [86, 79, 74]. However, regulating sensor precision will result in poor application performance, for example, gaming applications will have slow controls and response, mapping applications will be delayed/inaccurate, etc. Moreover, some sensors (such as camera and microphone) will be rendered unusable at very low sampling rates. In the more recent work using smartwatches as a side-channel [115], Wang et al. completely overlooked the necessity of having protection mechanisms. In this chapter, we not only demonstrate the feasibility of keystroke inference attacks using

smartwatches as a side-channel, but we also design, implement, and evaluate a new context-aware protection framework to defend against such attacks.

While design-time protection mechanisms presented in Section 5 are able to prevent such attacks in some form, they require specially designed interfaces, which may not be readily available. In this chapter, our goal is to show the effectiveness and usability of *run-time* protection measures running on the wearable devices and using contextual information to dynamically regulate zero-permission sensor data, when users are detected to be vulnerable to a known inference attack. In this direction, we propose and implement a new context-aware protection framework which can automatically activate various protection mechanisms whenever typing activity on an external keyboard is detected. We also empirically evaluate the protection framework in real-life usage scenario.

6.2 Run-Time Protection for External Keyboards

Our proposed attack for external keyboards in Chapter 3 demonstrates the need for reforms on how sensors on smartwatches, and other wearable devices, are accessed by applications. Even innocuous sensors can be used as side-channels to indirectly infer private information. However, there is no straightforward remedy to such privacy threats. In this section, we present a smart countermeasure to prevent such attacks in future.

The simplest way to protect against the presented attack would be to remove the smartwatch from wrist while typing. But repetitive removal of the watch (and remembering when to remove) can become a burden for the user, as a result of which,

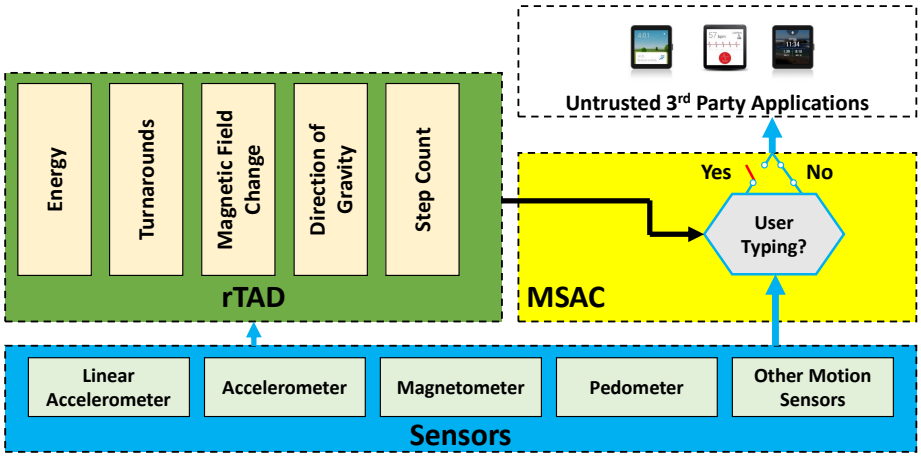
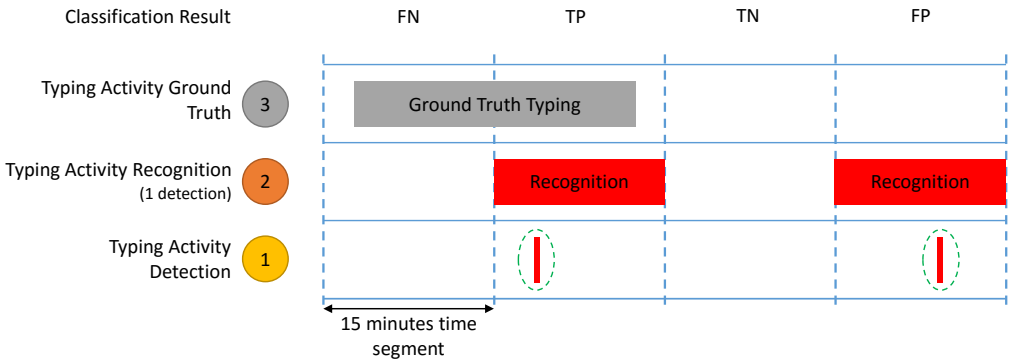
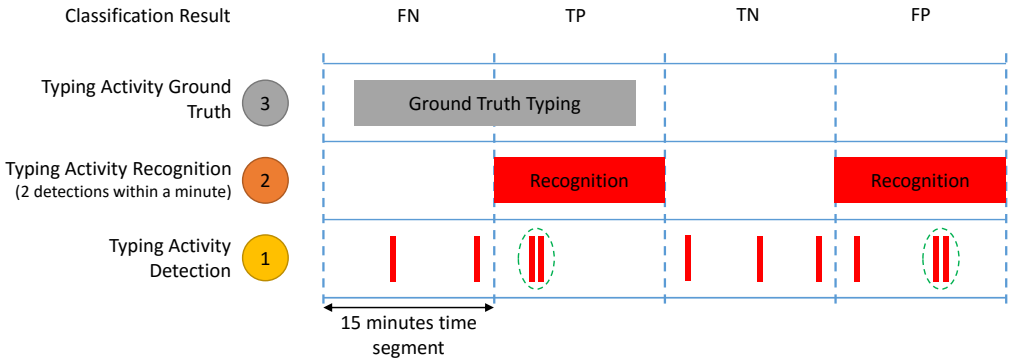


Figure 6.1: The protection framework against keystroke inference attacks. Third party applications get unrestricted access to motion sensors only when rTAD reports that the user is not typing at the moment.

the user may choose to ignore the threat altogether. To draw a favorable balance between utility, usability and privacy while using wearable devices, we need smarter sensor access controls. We feel that sensor access controls need to be context-aware in order to automatically manage an application's sensor permissions, without having the user to manually change these settings repetitively. As part of our efforts to prevent smartwatch based side-channel inference attacks demonstrated earlier in Chapter 3, we design, implement and evaluate a context-aware access control framework for smartwatch sensors. The framework (Figure 6.1) consists of two key components: (i) a real-time typing activity detection (rTAD) and (ii) a motion sensor access-controller (MSAC). Preliminary evaluations of the framework lead us to very promising results.



(a) N=1



(b) N=2

Figure 6.2: From bottom to top, (1) the 10 second detection windows where typing was detected are marked in red vertical lines, (2) when N detections occurs within a minute, typing activity is recognized for that 15 minute time segment, and (3) the ground truth collect by prompting the participant.

6.2.1 Typing Activity Recognition

Detecting when a smartwatch user is typing on a keyboard is not as straightforward as detecting contexts such as location or temperature. Running complex machine learning based classification on very limited processors of smartwatches is not a practical solution. Moreover, rTAD must be real-time so that protection measures can be activated proactively. The second bottleneck is the limited battery capacity. Sampling sensors at high frequency and performing complex computations discharges the smartwatch battery rapidly, requiring frequent recharge of the device. For example, continuous sampling of the accelerometer and gyroscope at 50 Hz on our Samsung Gear Live smartwatch completely drains the battery in less than an hour of use, which will severely affect the usability. From these observations it is evident that we have to identify features which are easy to compute and compatible with low sensor sampling rates. However, reducing sampling frequency also means compromising the accuracy of classification. To compensate the reduction in sampling frequency, we design features using a assorted set of motion sensors (sampled at approximately 15 Hz) in order to make a highly perceptive decision. Following are the five feature we incorporate in our proposed rTAD component:

- **Energy:** Activity measured in terms of cumulative linear accelerometer readings. An unworn watch lying on a table has zero energy, while an athlete's watch has high energy. Typing activity typically results in low but non-zero energy. We apply a low pass filter over the linear accelerometer to eliminate high-frequency noise caused by environmental factors.

- **Turnarounds:** Major positive to negative (or vice versa) changes on linear accelerometer readings signify the turnarounds adjoining transitional movements between key presses. Multiple turnarounds in close time proximity can be associated with many activities, such as brushing teeth, eating, playing drums, etc. As a result, we need additional features to distinguish typing from other similar activities.

- **Magnetic Field Change:** Wrists are not rotated significantly when a user types on a QWERTY keyboard, while sitting in front of a stationary desk. Rapid change in north, east and nadir vectors implies non-typing activity.

- **Direction of Gravity:** Gravity generally remains dominant on z-axis of accelerometer while typing on a horizontally placed keyboard. Any major fluctuations or gravity on x-axis or y-axis implies other activities.

- **Step Count:** We assume that the user will be stationary while typing on a computer keyboard. Thus, whenever step count increases, we rule out typing activity.

At the end of every 10 seconds, rTAD conducts a binary classification of weather the user typed in the last 10 seconds or not. All features for the binary classification resets at the starting of the next 10 second window. The cutoff parameters for *Energy* and *Turnarounds* features are calculated using the test data collected in Section 5.15, whereas cutoff parameters for *Magnetic Field Change* and *Direction of Gravity* features are calculated heuristically. Cutoff parameter for *Step Count* is straightforward, because any increase in the pedometer count indicates walking. The

Table 6.1: The rTAD's binary classification uses the following parameters. At the end of each 10 second windows, if any of the features are outside these parameter ranges, then non-typing activity is identified, and vice versa.

Feature	**Parameters Ranges**
Energy	>= 10 and <= 200, after applying low-pass filter
Turnarounds	>= 6
Magnetic Field Change	<= 2 samples with change in north direction
Direction of Gravity	>= 5 samples with fluctuations, or gravity on x-axis or y-axis
Step Count	<= $PreviousStepCount$

exact cutoff parameters of each feature used in our evaluation of rTAD can be found in Table 6.1.

Like many other activity recognition problems, there is an inverse relationship between *precision* (number of actual typing instances divided by number of all identified typing instances) and *recall* (number of identified actual typing instances divided by number of actual typing instances), where it is possible to increase one at the cost of reducing the other. A common approach to draw a favorable balance between false positives and false negatives is to 'recognize' an activity only when multiple instances of the activity are detected in close time proximity [107]. However, the use of rTAD is very different than most informative activity detection applications. The purpose of rTAD is to enable countermeasures against keystroke inference attacks as soon as typing activity is identified. In other words, rTAD's goal is to maximize

recall, but not to an extent where high false positives start affecting the utility of other non-malicious applications installed on the smartwatch. We evaluate rTAD in two different settings (visually explained in Figure 6.2):

- **N=1:** Typing activity is recognized whenever a 10 second window is classified as a typing window. As a result, countermeasures against keystroke inference attacks can be initiated as early as 10 seconds from when the user starts typing.
- **N=2:** Typing activity is recognized when two or more 10 second windows are classified as typing windows within a minute. Countermeasures against keystroke inference attacks can be initiated no sooner than 20 seconds from when the user starts typing.

6.2.2 Protection

Once rTAD identifies that the user is typing, countermeasures against keystroke inference attacks can be activated automatically in an non-intrusive fashion. And since the protection mechanism is activated only when the user is identified to be typing on a keyboard, the utility of the motion sensors is not affected when user is actively using other applications on the smartwatch (such as playing games that use motion sensors). Such smart protection measures can be undertaken by the MSAC implemented in the operating system itself, or as a trusted middle-ware. For the framework to work, we assume that all third party applications get access to motion sensor data only via the MSAC and the MSAC has the ability to modify or restrict the flow of motion sensor data. Although this assumption requires change in operating system architecture, it should be a rudimentary task for operating system

developers. Also, it should be noted that this assumption does not require changes in existing third party application, as long as the APIs to access motion sensors remain unchanged. Since MSAC requires a change in the operating system architecture, we are unable to implement a working MSAC. However, below we list out some of the strategies that the MSAC can adopt when rTAD reports that the user is typing:

- **Complete Blocking:** This strategy is the safest as it will completely block the side-channel, but it can also harm the utility of other non-malicious applications that may want to perform passive computing with motion data.

- **Reduced Sampling Rate:** When a user types for significant amount of time in a day, complete blocking of the motion sensor data from third party applications can greatly harm the utility of other non-malicious applications. In order to preserve some of the utility, MSAC can provide third party applications access to motion sensors at a reduced sampling rate. Restricting the precision at which applications are allowed to access the sensors reduces the efficiency of side-channel attacks [86, 79].

- **Random Out of Order Blocks:** A smarter MSAC can send out of order blocks of sensor readings to third party applications. Random out of order blocks of sensor data can greatly lower the inference accuracy of side-channel attacks, but may still preserve utility for certain non-malicious application. For example, a daily calorie counter may not be significantly affected by out of order blocks of sensor readings. That is because the calorie count will be accurate as long as all the motions are captured by the application, even if out

of order. Size of block and randomization algorithm will play a signification role in determining how much an adversary can recover versus the utility of randomly ordered blocks.

There can be other strategies that the MSAC can adopt as well. We think that it will be best if users are allowed to choose among the MSAC protection strategies, suitable for their personal lifestyles.

6.2.3 Evaluation

We implement and evaluate our proposed rTAD, because the effectiveness of the entire protection mechanism relies on rTAD. To evaluate rTAD, we use the same smartwatch setup detailed in Section 3.4.3. Preliminary evaluation involved 4 participants with varied lifestyles wearing the watch for long durations. If the rTAD application does not recognize typing activity, it prompts the participant every 15 minutes to collect ground truth. If the rTAD application does recognize typing activity, it prompts the user immediately for ground truth. In case the user continues to type for long period of time, the rTAD application does not ask the user for ground truth for 15 minutes after the initial detection. This avoids annoyance to the participants and results in equitable ground truth collection. In real usage, the user will not be prompted for ground truth, instead the MSAC will automatically start acting as soon as typing activity is reported by rTAD.

One problem that we encountered when evaluating rTAD was that in certain cases the *Magnetic Field Change* feature acted unexpectedly, introducing a lot of error in classification. We observed that the unexpected behavior occurred only while the

participant typed on a laptop. Further investigation revealed that the magnet inside the laptop's hard drive (which are normally installed directly under the keyboard) was responsible for this unexpected behavior. Since desktop keyboards are generally placed away from the hard drives, the *Magnetic Field Change* feature performed in an expected fashion in that case. For the remainder of the evaluation we do not use the *Magnetic Field Change* feature because it will be hard for the participants to remember and distinguish between laptop and desktop typing. However, as laptops featuring non-magnetic solid state drives are becoming popular, the *Magnetic Field Change* feature may eventually become useful in future.

The combined *true positives (TP)*, *true negatives (TN)*, *false positives (FP)*, and *false negatives (FN)* results from the 4 participants are shown in Figure 6.3. To better visualize the difference between the two settings, the values in Figure 6.3 are normalized with respect to the total number of ground truth collected in each setting. As explained with examples in Figure 6.2, TP signifies that the user was typing and rTAD correctly identified that the user was typing, and if rTAD failed to recognize that the user was typing, it was recorded as FN. Similarly, TN signifies that the user was not typing and rTAD correctly identified that the user was not typing, and if rTAD identified that the user was typing, it was recorded as FP.

In case of N=1, we observe lesser FN and higher TP, but at the cost of higher FP. In case of N=2, we observe lower FP, but at the cost of lower TP and higher FN. In other words, rTAD can gain recall by trading-off precision, and vice versa. Nevertheless, in both settings rTAD achieved high recall values, which asserts it's effectiveness in the protection framework.

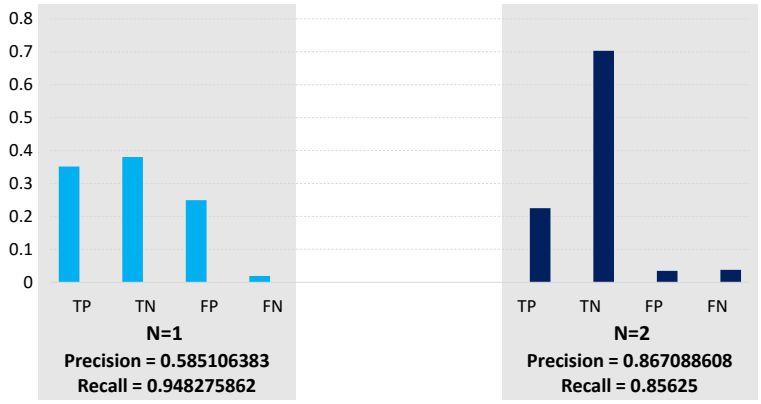


Figure 6.3: Normalized true positive (TP), true negative (TN), false positive (FP), and false negative (FN), along with precision and recall values.

6.2.4 Discussions

- **Left or Right:** Our keystroke inference attack framework, presented earlier in Chapter 3, requires the adversary to have a differently trained framework for targets wearing watch on their right hand. However, the design of rTAD (and thus the whole protection framework) makes it independent of which hand the smartwatch is worn on. As a result, rTAD can start working out of the box, without manual setup.

- **Usability:** Our primary focus while designing the protection framework was usability. We work towards a low processor intensive design, which in turn consumes low battery power. We identify activities similar to typing on keyboard, and try to minimize false positives. The protection mechanism works in an non-intrusive fashion as well, and we envision that the entire setup process in a real-life implementation will be very simple.

- **Non-Motion Sensor Features:** We restrict ourselves to features from sensors accessible by any third party application without explicit permissions from the user. While it may be beneficial for rTAD to use other sensors as well (such as, using the GPS sensors to track if the user is stationary), we decided otherwise. The primary reason behind this decision is that, in case rTAD is not implemented within the operating system, but installed as a trusted middle-ware, we do not want rTAD to have access to additional sensors. So, even if the trusted middle-ware misbehaves, it will not have access to any more sensor data than what is currently accessible to all third party applications.

6.3 Conclusion

We proposed and evaluated a run-time protection framework to automatically regulate zero-permission sensor access, aimed to improve typing activity privacy without degrading the utility of wearable devices. Our run-time protection mechanism can also be trivially extended to protect users performing other sensitive activities. For example, when users are opening a combination padlock or safe as presented in Chapter 4, a run-time protection framework can utilize the unlocking activity recognition technique (Section 4.4.3) to detect the sensitive activity and start regulating the zero-permission sensors.

Parts of this chapter appeared in [71].

CHAPTER 7
CONCLUSION

The first part of this dissertation was focused on developing a comprehensive technical understanding of the privacy risks associated with inference of private user interactions with other *cyber* and *physical* systems, primarily using wrist-wearables. A detailed evaluation of novel attack frameworks validated the feasibility of inference attacks on both cyber interfaces, such as mobile keypads and computer keyboards, and on physical systems, such as combination padlocks and safes. The feasibility of these attacks implies that manufacturers and developers should conduct critical security and privacy analysis before introducing any new wearable device and/or zero-permission sensor. It also indicates that new technologies, beyond wearables, should also be throughly scrutinized before they make their way to consumers.

The second part of this dissertation was aimed to protect user interactions by proposing new protection mechanisms, which took two different strategies. The proposed *design-time* protection mechanisms tries to prevent inference attacks by altering the interaction interfaces, and *run-time* protection mechanisms use contextual information to dynamically regulate zero-permission sensor data when users are detected to be vulnerable to a known inference attack. The usability of some of these protection mechanisms were comprehensively studied with the help of human subjects.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] IEEE Recommended Practices for Speech Quality Measurements. *IEEE Transactions on Audio and Electroacoustics* (1969).

[2] Abawi, Daniel F, Bienwald, Joachim, and Dorner, Ralf. Accuracy in Optical Tracking with Fiducial Markers: An Accuracy Function for ARToolKit. In *ACM International Symposium on Mixed and Augmented Reality (ISMAR)* (2004).

[3] aBlogtoWatch. Poll: What is Your Hand-Orientation & What Wrist Do You Wear Your Watch On? `www.ablogtowatch.com/poll-your-hand-orientation-what-wrist-wear-your-watch/`. Online; accessed 2017-06-07.

[4] Adelson, Edward H. Perceptual Organization and the Judgment of Brightness. *Science 262* (1993).

[5] Agrawal, Dakshi, Archambeault, Bruce, Rao, Josyula R, and Rohatgi, Pankaj. The EM Side-channel(s). In *Cryptographic Hardware and Embedded Systems* (2002).

[6] Ali, Kamran, Liu, Alex Xiao, Wang, Wei, and Shahzad, Muhammad. Keystroke Recognition using WiFi Signals. In *ACM International Conference on Mobile Computing and Networking (MobiCom)* (2015).

[7] Altun, Kerem, and Barshan, Billur. Human activity recognition using inertial/magnetic sensor units. In *International Workshop on Human Behavior Understanding* (2010), Springer.

[8] Asonov, Dmitri, and Agrawal, Rakesh. Keyboard Acoustic Emanations. In *IEEE Symposium on Security and Privacy* (2004).

[9] Aviv, Adam J, Sapp, Benjamin, Blaze, Matt, and Smith, Jonathan M. Practicality of accelerometer side channels on smartphones. In *ACM Annual Computer Security Applications Conference (ACSAC)* (2012).

[10] Backes, Michael, Chen, Tongbo, Duermuth, Markus, Lensch, Hendrik, and Welk, Martin. Tempest in a teapot: Compromising reflections revisited. In *IEEE Symposium on Security and Privacy* (2009).

[11] Backes, Michael, Dürmuth, Markus, Gerling, Sebastian, Pinkal, Manfred, and Sporleder, Caroline. Acoustic Side-Channel Attacks on Printers. In *USENIX Security Symposium* (2010).

[12] Backes, Michael, Durmuth, Markus, and Unruh, Dominique. Compromising Reflections-or-How to Read LCD Monitors Around the Corner. In *IEEE Symposium on Security and Privacy* (2008).

[13] Bailey, Rosemary A. *Design of comparative experiments*, vol. 25. Cambridge University Press, 2008.

[14] Barisani, Andrea, and Bianco, Daniele. Sniffing Keystrokes with Lasers/Voltmeters. *Black Hat USA* (2009).

[15] Beck, E, Christiansen, M, Kjeldskov, Jesper, Kolbe, Nikolaj, and Stage, Jan. Experimental evaluation of techniques for usability testing of mobile systems in a laboratory setting. In *Australian Conference on Human-Computer Interaction* (2003).

[16] Berger, Yigael, Wool, Avishai, and Yeredor, Arie. Dictionary Attacks using Keyboard Acoustic Emanations. In *ACM Conference on Computer and Communications Security (CCS)* (2006).

[17] Blaze, Matt. Rights amplification in master-keyed mechanical locks. *IEEE Security & Privacy* (2003).

[18] Blaze, Matt. Safecracking for the computer scientist. *U. Penn CIS Department Technical Report* (2004).

[19] Brooke, John, et al. SUS-A quick and dirty usability scale. *Usability evaluation in industry 189*, 194 (1996).

[20] Cai, Liang, and Chen, Hao. Touchlogger: Inferring keystrokes on touch screen from smartphone motion. In *USENIX Summit on Hot Topics in Security (HotSec)* (2011).

[21] Cai, Liang, and Chen, Hao. On the practicality of motion based keystroke inference attack. In *International Conference on Trust and Trustworthy Computing* (2012), Springer.

[22] Cai, Liang, Machiraju, Sridhar, and Chen, Hao. Defending against sensor-sniffing attacks on mobile phones. In *ACM MobiHeld Workshop* (2009).

[23] Cappos, Justin, Wang, Lai, Weiss, Rebecca, Yang, Yi, and Zhuang, Yanyan. BlurSense: Dynamic Fine-Grained Access Control for Smartphone Privacy. In *IEEE Sensors Applications Symposium* (2014).

[24] Caruana, Rich, Niculescu-Mizil, Alexandru, Crew, Geoff, and Ksikes, Alex. Ensemble selection from libraries of models. In *International Conference on Machine Learning (ICML)* (2004).

[25] Conti, Mauro, Nguyen, Vu Thien Nga, and Crispo, Bruno. CRePE: Context-Related Policy Enforcement for Android. In *Information Security*. Springer, 2010.

[26] Crager, Kirsten, Maiti, Anindya, Jadliwala, Murtuza, and He, Jibo. Information leakage through mobile motion sensors: User awareness and concerns. In *European Workshop on Usable Security (EuroUSEC)* (2017).

[27] Deisenroth, Marc Peter, and Ohlsson, Henrik. A general perspective on gaussian filtering and smoothing: Explaining current and deriving new algorithms. In *IEEE American Control Conference* (2011).

[28] Denney, K., Uluagac, A. S., Akkaya, K., and Bhansali, S. A novel storage covert channel on wearable devices using status bar notifications. In *IEEE Annual Consumer Communications Networking Conference (CCNC)* (2016).

[29] Dewri, Rinku, Annadata, Prasad, Eltarjaman, Wisam, and Thurimella, Ramakrishna. Inferring trip destinations from driving habits data. In *ACM Workshop on Privacy in the Electronic Society* (2013).

[30] Dix, Alan. *Human-Computer Interaction*. Springer, 2009.

[31] EPSON. Moverio BT-200. `www.epson.com/MoverioBT200`. Online; accessed 2017-06-07.

[32] Faruque, Al, Abdullah, Mohammad, Chhetri, Sujit Rokka, Canedo, Arquimedes, and Wan, Jiang. Acoustic Side-Channel Attacks on Additive Manufacturing Systems. In *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)* (2016).

[33] Felt, Adrienne Porter, Finifter, Matthew, Chin, Erika, Hanna, Steve, and Wagner, David. A survey of mobile malware in the wild. In *ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM)* (2011).

[34] Foo Kune, Denis, and Kim, Yongdae. Timing Attacks on PIN Input Devices. In *ACM Conference on Computer and Communications Security (CCS)* (2010).

[35] Friedman, Jeffrey. Tempest: A Signal Problem. *NSA Cryptologic Spectrum* (1972).

[36] Fukano, Jun, Mashita, Tomohiro, Hara, Takahiro, Kiyokawa, Kiyoshi, Takemura, Haruo, and Nishio, Shojiro. A next location prediction method for smartphones using blockmodels. In *IEEE Virtual Reality (VR)* (2013).

[37] Haigh, Ruth. The Ageing Process: A Challenge for Design. *Applied Ergonomics 24*, 1 (1993).

[38] Halevi, T., and Saxena, N. A closer look at keyboard acoustic emanations: Random passwords, typing styles and decoding techniques. In *ACM Symposium on Information, Computer and Communications Security (ASIACCS)* (2012).

[39] Hall, Mark, Frank, Eibe, Holmes, Geoffrey, Pfahringer, Bernhard, Reutemann, Peter, and Witten, Ian H. The weka data mining software: an update. *In ACM SigKDD Explorations Newsletter 11*, 1 (2009).

[40] Han, Jun, Owusu, Emmanuel, Nguyen, Le T, Perrig, Adrian, and Zhang, Joy. Accomplice: Location inference using accelerometers on smartphones. In *IEEE International Conference on COMmunication Systems & NETworkS* (2012).

[41] Hart, Sandra G, and Staveland, Lowell E. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. *Advances in Psychology 52* (1988).

[42] Hayashi, Yuichi, Homma, Naofumi, Miura, Mamoru, Aoki, Takafumi, and Sone, Hideaki. A threat for tablet pcs in public space: Remote visualization of screen images using em emanation. In *ACM Conference on Computer and Communications Security (CCS)* (2014).

[43] Hemminki, Samuli, Nurmi, Petteri, and Tarkoma, Sasu. Accelerometer-based transportation mode detection on smartphones. In *ACM Conference on Embedded Networked Sensor Systems* (2013).

[44] Ho, Bo-Jhang, Martin, Paul, Swaminathan, Prashanth, and Srivastava, Mani. From pressure to path: Barometer-based vehicle tracking. In *ACM BuildSys* (2015).

[45] Hojjati, Avesta, Adhikari, Anku, Struckmann, Katarina, Chou, Edward, Tho Nguyen, Thi Ngoc, Madan, Kushagra, Winslett, Marianne S, Gunter, Carl A, and King, William P. Leave Your Phone at the Door: Side Channels that Reveal Factory Floor Secrets. In *ACM Conference on Computer and Communications Security (CCS)* (2016).

[46] Holmes, Ashton, Desai, Sunny, and Nahapetian, Ani. Luxleak: capturing computing activity using smart device ambient light sensors. In *ACM Workshop on Interacting with Smart Objects (SmartObjects)* (2016).

[47] Huang, Kuo-Ying. Challenges in Human-Computer Interaction Design for Mobile Devices. In *IAENG World Congress on Engineering and Computer Science* (2009).

[48] Huebler, Michael. The New Master Lock Speed Dial /ONE Combination Padlock - An Inside View. In *Hacking at Random* (2009).

[49] Humayoun, Shah Rukh, Hess, Steffen, Kiefer, Felix, and Ebert, Achim. Patterns for Designing Scalable Mobile App User Interfaces for Multiple Platforms. In *British Human Computer Interaction Conference* (2014), BCS.

[50] IDC.com. Wearables Aren't Dead, They're Just Shifting Focus as the Market Grows 16.9% in the Fourth Quarter, According to IDC. `www.idc.com/getdoc.jsp?containerId=prUS42342317/`. Online; accessed 2017-06-07.

[51] Iqbal, Muhammad Usman, and Lim, Samsung. Privacy implications of automated gps tracking and profiling. *IEEE Technology and Society Magazine 29*, 2 (2010).

[52] Ishida, Masamitsu, Frank, Paul H, Doi, Kunio, and Lehr, James L. High Quality Digital Radiographic Images: Improved Detection of Low-Contrast Objects and Preliminary Clinical Studies. *Radiographics 3*, 2 (1983).

[53] Jackson, Wallace. Android UI Layout Conventions, Differences and Approaches. In *Pro Android UI*. Springer, 2014.

[54] Jahrer, Michael, Töscher, Andreas, and Legenstein, Robert. Combining predictions for accurate recommender systems. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)* (2010).

[55] Jermyn, Ian, Mayer, Alain, Monrose, Fabian, Reiter, Michael K., and Rubin, Aviel D. The Design and Analysis of Graphical Passwords. In *USENIX Security Symposium* (1999).

[56] Jones, Matt, and Marsden, Gary. *Mobile Interaction Design*. John Wiley & Sons, 2006.

[57] Karatas, Cagdas, Liu, Luyang, Li, Hongyu, Liu, Jian, Wang, Yan, Tan, Sheng, Yang, Jie, Chen, Yingying, Gruteser, Marco, and Martin, Richard. Leveraging Wearables for Steering and Driver Tracking. In *IEEE International Conference on Computer Communications (INFOCOM)* (2016).

[58] Karen Scarfone. The Next BYOD Challenge for Feds: Wearables. `https://fedtechmagazine.com/article/2017/05/next-byod-challenge-feds-wearables`. Online; accessed 2017-06-07.

[59] Kato, Hirokazu. Inside ARToolKit. In *IEEE International Workshop on Augmented Reality Toolkit* (2007).

[60] Kuhn, Markus G. Optical Time-Domain Eavesdropping Risks of CRT Displays. In *IEEE Symposium on Security and Privacy* (2002).

[61] Kuhn, Markus G, and Anderson, Ross J. Soft Tempest: Hidden Data Transmission Using Electromagnetic Emanations. In *Information Hiding, Lecture Notes in Computer Science* (1998).

[62] Kumar, Manu, Garfinkel, Tal, Boneh, Dan, and Winograd, Terry. Reducing Shoulder-Surfing by Using Gaze-Based Password Entry. In *Symposium On Usable Privacy and Security (SOUPS)* (2007).

[63] Kwapisz, Jennifer R, Weiss, Gary M, and Moore, Samuel A. Activity recognition using cell phone accelerometers. *ACM SigKDD Explorations Newsletter 12*, 2 (2011).

[64] Kwon, Taekyoung, Shin, Sooyeon, and Na, Sarang. Covert Attentional Shoulder Surfing: Human Adversaries Are More Powerful than Expected. *IEEE Transactions on SMC: Systems 44*, 6 (2014).

[65] Lashkari, Arash Habibi, Abdul Manaf, Azizah, Masrom, Maslin, and Daud, Salwani Mohd. Security evaluation for graphical password. In *Digital Information and Communication Technology and Its Applications: International Conference (DICTAP)* (2011), Springer.

[66] LG User Guide. Lock screen. `www.lg.com/us/mobile-phones/VS985/Userguide/426.html`. Online; accessed 2017-06-07.

[67] Li, Mengyuan, Meng, Yan, Liu, Junyi, Zhu, Haojin, Liang, Xiaohui, Liu, Yao, and Ruan, Na. When csi meets public wifi: Inferring your mobile phone password via wifi signals. In *ACM Conference on Computer and Communications Security (CCS)* (2016).

[68] Lifehacker. Crack a Master Combination Padlock Redux. `www.lifehacker.com/5376442/crack-a-master-combination-padlock-redux/`. Online; accessed 2017-06-07.

[69] Liu, Li, Peng, Yuxin, Wang, Shu, Liu, Ming, and Huang, Zigang. Complex Activity Recognition Using Time Series Pattern Dictionary Learned from Ubiquitous Sensors. *Information Sciences 340* (2016).

[70] Liu, Xiangyu, Zhou, Zhe, Diao, Wenrui, Li, Zhou, and Zhang, Kehuan. When Good Becomes Evil: Keystroke Inference with Smartwatch. In *ACM Conference on Computer and Communications Security (CCS)* (2015).

[71] Maiti, Anindya, Armbruster, Oscar, Jadliwala, Murtuza, and He, Jibo. Smartwatch-based keystroke inference attacks and context-aware protection mechanisms. In *ACM Symposium on Information, Computer and Communications Security (ASIACCS)* (2016).

[72] Maiti, Anindya, Crager, Kirsten, Jadliwala, Murtuza, He, Jibo, Kwiat, Kevin, and Kamhoua, Charles. Randompad: Usability of randomized mobile keypads for defeating inference attacks. In *IEEE EuroS&P Workshop on Innovations in Mobile Privacy & Security (IMPS)* (2017).

[73] Maiti, Anindya, Heard, Ryan, Sabra, Mohd, and Jadliwala, Murtuza. Towards Inferring Mechanical Lock Combinations using Wrist-Wearables as a Side-Channel. *ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)* (2018).

[74] Maiti, Anindya, Jadliwala, Murtuza, He, Jibo, and Bilogrevic, Igor. (Smart)Watch Your Taps: Side-channel Keystroke Inference Attacks Using Smartwatches. In *ACM International Symposium on Wearable Computers (ISWC)* (2015).

[75] Maiti, Anindya, Jadliwala, Murtuza, He, Jibo, and Bilogrevic, Igor. Side-Channel Inference Attacks on Mobile Keypads using Smartwatches. *IEEE Transactions of Mobile Computing* (2018).

[76] Maiti, Anindya, Jadliwala, Murtuza, and Weber, Chase. Preventing shoulder surfing using randomized augmented reality keyboards. In *IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)* (2017).

[77] Marquardt, Philip, Verma, Arunabh, Carter, Henry, and Traynor, Patrick. (sp)iPhone: Decoding Vibrations From Nearby Keyboards Using Mobile Phone Accelerometers. In *ACM Conference on Computer and Communications Security (CCS)* (2011).

[78] Matic, Aleksandar, Osmani, Venet, and Mayora, Oscar. Speech activity detection using accelerometer. In *IEEE International Conference on Engineering in Medicine and Biology Society (EMBC)* (2012).

[79] Michalevsky, Yan, Boneh, Dan, and Nakibly, Gabi. Gyrophone: Recognizing Speech from Gyroscope Signals. In *USENIX Security Symposium* (2014).

[80] Michalevsky, Yan, Schulman, Aaron, Veerapandian, Gunaa Arumugam, Boneh, Dan, and Nakibly, Gabi. Powerspy: Location tracking using mobile device power analysis. In *USENIX Security Symposium* (2015).

[81] Miluzzo, Emiliano, Varshavsky, Alexander, Balakrishnan, Suhrid, and Choudhury, Romit Roy. Tapprints: your finger taps have fingerprints. In *ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)* (2012).

[82] Mow, Van C, Ratcliffe, Anthony, and Woo, Savio LY. *Biomechanics of Diarthrodial Joints*, vol. 1. Springer Science & Business Media, 2012.

[83] Narain, Sashank, Vo-Huu, Triet D, Block, Kenneth, and Noubir, Guevara. Inferring user routes and locations using zero-permission mobile sensors. In *IEEE Symposium on Security and Privacy* (2016).

[84] Nguyen, L., Cheng, H., Wu, P., Buthpitiya, S., and Zhang, Y. Pnlum: System for prediction of next location for users with mobility. In *Nokia Mobile Data Challenge Workshop* (2012).

[85] Ortiz, Reyes, and Luis, Jorge. *Smartphone-Based Human Activity Recognition.* Springer Theses, 2015.

[86] Owusu, Emmanuel, Han, Jun, Das, Sauvik, Perrig, Adrian, and Zhang, Joy. ACCessory: Password Inference using Accelerometers on Smartphones. In *ACM Workshop on Mobile Computing Systems and Applications (HotMobile)* (2012).

[87] Passfaces. Two Factor Authentication - Graphical Passwords. `www.realuser.com`. Online; accessed 2017-06-07.

[88] Pattison, Matthew, and Stedmon, Alex W. Inclusive Design and Human Factors: Designing Mobile Phones for Older Users. *Psychnology Journal 4*, 3 (2006).

[89] Quisquater, Jean-Jacques, and Samyde, David. ElectroMagnetic Analysis (EMA): Measures and Countermeasures for Smart Cards. In *Smart Card Programming and Security, Lecture Notes in Computer Science* (2001).

[90] Rayner, Keith, Slattery, Timothy J, and Bélanger, Nathalie N. Eye movements, the perceptual span, and reading speed. *Psychonomic Bulletin & Review 17*, 6 (2010).

[91] Rossi, Mirco, Feese, Sebastian, Amft, Oliver, Braune, Nils, Martis, Sandro, and Troster, G. Ambientsense: A real-time ambient sound recognition system for smartphones. In *IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)* (2013).

[92] Roth, Volker, Richter, Kai, and Freidinger, Rene. A PIN-entry Method Resilient Against Shoulder Surfing. In *ACM Conference on Computer and Communications Security (CCS) 2004.*

[93] Ryu, Young Sam, Koh, Do Hyong, Aday, Brad L, Gutierrez, Xavier A, and Platt, John D. Usability Evaluation of Randomized Keypad. *Journal of Usability Studies 5*, 2 (2010).

[94] Schlegel, Roman, Zhang, Kehuan, Zhou, Xiao-yong, Intwala, Mehool, Kapadia, Apu, and Wang, XiaoFeng. Soundcomber: A Stealthy and Context-Aware Sound Trojan for Smartphones. In *The Network and Distributed System Security Symposium (NDSS)* (2011).

[95] Sears, Andrew, and Zha, Ying. Data Entry for Mobile Devices using Soft Keyboards: Understanding the Effects of Keyboard Size and User Tasks. *International Journal of HCI 16*, 2 (2003).

[96] Shoaib, Muhammad, Bosch, Stephan, Incel, Ozlem Durmaz, Scholten, Hans, and Havinga, Paul JM. Complex human activity recognition using smartphone and wrist-worn motion sensors. *Sensors 16*, 4 (2016).

[97] Shrestha, Prakash, Mohamed, Manar, and Saxena, Nitesh. Slogger: Smashing motion-based touchstroke logging with transparent system noise. In *ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)* (2016), ACM.

[98] Simon, Laurent, and Anderson, Ross. Pin skimmer: inferring pins through the camera and microphone. In *ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM)* (2013), ACM.

[99] Smulders, Peter. The Threat of Information Theft by Reception of Electromagnetic Radiation from RS-232 Cables. *Computers & Security 9*, 1 (1990).

[100] Snedecor, George W. Statistical methods: Applied to experiments in agriculture and biology. Iowa State University Press.

[101] Software House. Scramble Keypad SP-100. `www.swhouse.com/products/`. Online; accessed 2017-06-07.

[102] Song, Chen, Lin, Feng, Ba, Zhongjie, Ren, Kui, Zhou, Chi, and Xu, Wenyao. My Smartphone Knows What You Print: Exploring Smartphone-Based Side-Channel Attacks Against 3D Printers. In *ACM Conference on Computer and Communications Security (CCS)* (2016).

[103] Spehar, Branka, and Owens, Caleb. When Do Luminance Changes Capture Attention? *Attention, Perception and Psychophysics 74*, 4 (2012).

[104] Steenbekkers, LPA, Dirken, JM, and Beijsterveldt, CEMV. Design-Relevant Functional Capacities of the Elderly, Assessed in the Delft Gerontechnology Project. In *Triennial Congress of the International Ergonomics Association* (1997).

[105] Sun, Jingchao, Xiaocong, Chen, Yimin, Zhang, Jinxue, Zhang, Yanchao, and Zhang, Rui. VISIBLE: Video-Assisted Keystroke Inference from Tablet Backside Motion. In *The Network and Distributed System Security Symposium (NDSS)* (2016).

[106] Tari, Furkan, Ozok, Ant, and Holden, Stephen H. A Comparison of Perceived and Real Shoulder-Surfing Risks Between Alphanumeric and Graphical Passwords. In *Symposium On Usable Privacy and Security (SOUPS)* (2006).

[107] Thomaz, Edison, Essa, Irfan, and Abowd, Gregory D. A Practical Approach for Recognizing Eating Moments with Wrist-mounted Inertial Sensing. In *ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)* (2015).

[108] Tiwari, Vishnu Shankar, Arya, Arti, and Chaturvedi, S. Route prediction using trip observations and map matching. In *IEEE International Advance Computing Conference (IACC)* (2013).

[109] Tizon, Xavier, and Smedby, Örjan. Segmentation with Gray-Scale Connectedness Can Separate Arteries and Veins in MRA. *Journal of Magnetic Resonance Imaging 15*, 4 (2002).

[110] Uluagac, A Selcuk, Subramanian, Venkatachalam, and Beyah, Raheem. Sensory channel threats to cyber physical systems: A wake-up call. In *IEEE Conference on Communications and Network Security (CNS)* (2014).

[111] UXmatters. How do users really hold mobile devices? `www.uxmatters.com/mt/archives/2013/02/how-do-users-really-hold-mobile-devices.php`. Online; accessed 2017-06-07.

[112] Van Eck, Wim. Electromagnetic Radiation from Video Display Units: An Eavesdropping Risk? *Computers & Security 4*, 4 (1985).

[113] Vuagnoux, Martin, and Pasini, Sylvain. Compromising electromagnetic emanations of wired and wireless keyboards. In *USENIX Security Symposium* (2009).

[114] Wang, Chen, Guo, Xiaonan, Wang, Yan, Chen, Yingying, and Liu, Bo. Friend or Foe?: Your Wearable Devices Reveal Your Personal Pin. In *ACM Symposium on Information, Computer and Communications Security (ASIACCS)* (2016).

[115] Wang, He, Lai, Ted Tsung-Te, and Roy Choudhury, Romit. Mole: Motion leaks through smartwatch sensors. In *ACM International Conference on Mobile Computing and Networking (MobiCom)* (2015).

[116] Wen, Hongyi, Ramos Rojas, Julian, and Dey, Anind K. Serendipity: Finger gesture recognition using an off-the-shelf smartwatch. In *ACM CHI Conference on Human Factors in Computing Systems* (2016).

[117] Xu, Chao, Pathak, Parth H, and Mohapatra, Prasant. Finger-writing with smartwatch: A case for finger and hand gesture recognition using smartwatch. In *ACM Workshop on Mobile Computing Systems and Applications (HotMobile)* (2015).

[118] Xu, Zhi, Bai, Kun, and Zhu, Sencun. Taplogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors. In *ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)* (2012).

[119] Yan, Qiang, Han, Jin, Li, Yingjiu, Zhou, Jianying, and Deng, Robert H. Designing Leakage-resilient Password Entry on Touchscreen Mobile Devices. In *ACM Symposium on Information, Computer and Communications Security (ASIACCS)* (2013).

[120] York, Derek. Least-Squares Fitting of a Straight Line. *Canadian Journal of Physics 44*, 5 (1966).

[121] Yu, Tuo, Jin, Haiming, and Nahrstedt, Klara. Writinghacker: Audio based eavesdropping of handwriting via mobile devices. In *ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)* (2016).

[122] Yue, Qinggang, Ling, Zhen, Fu, Xinwen, Liu, Benyuan, Ren, Kui, and Zhao, Wei. Blind Recognition of Touched Keys on Mobile Devices. In *ACM Conference on Computer and Communications Security (CCS)* (2014).

[123] Zhang, Mi, and Sawchuk, Alexander A. A feature selection-based framework for human activity recognition using wearable multimodal sensors. In *ICST International Conference on Body Area Networks* (2011).

[124] Zhang, Shumei, McCullagh, Paul, Nugent, Chris, and Zheng, Huiru. Activity monitoring using a smart phone's accelerometer with hierarchical classification. In *IEEE International Conference on Intelligent Environments* (2010).

[125] Zhang, Yang, Xia, Peng, Luo, Junzhou, Ling, Zhen, Liu, Benyuan, and Fu, Xinwen. Fingerprint Attack Against Touch-enabled Devices. In *ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM)* (2012).

[126] Zhu, Tong, Ma, Qiang, Zhang, Shanfeng, and Liu, Yunhao. Context-free attacks using keyboard acoustic emanations. In *ACM Conference on Computer and Communications Security (CCS)* (2014).

[127] Zhuang, Li, Zhou, Feng, and Tygar, J. D. Keyboard acoustic emanations revisited. *ACM Transactions on Information and System Security* (2009).

[128] Zimmerman, Donald W. Teacher's Corner: A Note on Interpretation of the Paired-Samples t Test. *Journal of Educational and Behavioral Statistics 22*, 3 (1997).

[129] Zuffi, Silvia, Brambilla, Carla, Beretta, Giordano, and Scala, Paolo. Human computer interaction: Legibility and contrast. In *International Conference on Image Analysis and Processing (ICIAP)* (2007).