FRIEND OR FOE? EVALUATING SENSOR-BASED INFORMATION SIDE-CHANNELS AND COVERT COMMUNICATION CHANNELS ON MODERN WEARABLE DEVICES

by

RAVEEN WIJEWICKRAMA, M.Sc.

DISSERTATION
Presented to the Graduate Faculty of
The University of Texas at San Antonio
In Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE

COMMITTEE MEMBERS:
Murtuza Jadliwala, Ph.D., Chair
Xiaoyin Wang, Ph.D.
Wei Wang, Ph.D.
Palden Lama, Ph.D.
Anindya Maiti, Ph.D.

THE UNIVERSITY OF TEXAS AT SAN ANTONIO
College of Sciences
Department of Computer Science
August 2024

DEDICATION

To all those who brightened my days with memes and reels, your humor and support carried me through the toughest moments. This dissertation is dedicated to you.

ACKNOWLEDGEMENTS

I am deeply grateful to my supervisor, Dr. Murtuza Jadliwala, for his endless support, patience, and guidance over the past few years. His mentorship has been invaluable in shaping my research and academic journey. I would also like to extend my gratitude to Dr. Anindya Maiti for his continuous mentoring throughout my PhD. A special thank you to Dr. Xiaoyin Wang, Dr. Wei Wang, Dr. Palden Lama and Dr. Rocky Slavin for being a part of my PhD dissertation committee. Your constructive feedback and suggestions have been valuable in refining my work. My sincere thanks to Suzanne Tanaka from the Institute for Cyber Security for her continuous help throughout my program, from managing my funding to research equipment purchases, and even offering invaluable life advice. I'm also thankful to Susan Allen from the Department of Computer Science for handling my course-related paperwork. I would like to acknowledge the past and present members of SPriTELab, including Kavita, Nisha, and others, for their support. My friends, Buwaneka, Comfort, Senal, Nibras, Imanee and Kavita, have been a tremendous support system throughout my PhD program, and I am deeply thankful for their encouragement. I also want to thank my friends Buddhi and Aruni for all their support in the last one and half years of my PhD. A special thank should also go to my family and taxpayers of both Sri Lanka and USA for funding my education. Finally, I am forever grateful to my family. To my parents and my two sisters, your unwavering support and belief in me have been my greatest source of strength. This accomplishment would not have been possible without you. And of course, a "stomachfelt" thank you to ramen for being a constant source of sustenance when I needed it most!

This Doctoral Dissertation was produced in accordance with guidelines which permit the inclusion as part of the Doctoral Dissertation the text of an original paper, or papers, submitted for publication. The Doctoral Dissertation must still conform to all other requirements explained in the Guide for the Preparation of a Doctoral Dissertation at The University of Texas at San Antonio. It must include a comprehensive abstract, a full introduction and literature review, and a final overall conclusion. Additional material (procedural and design data as well as descriptions of equipment) must be provided in sufficient detail to allow a clear and precise judgment to be made of the importance and originality of the research reported. It is acceptable for this Doctoral Dissertation to include as chapters authentic copies of papers already published, provided these meet type size, margin, and legibility requirements. In such cases, connecting texts, which provide logical bridges between different manuscripts, are mandatory. Where the student is not the sole author of a manuscript, the student is required to make an explicit statement in the introductory material to that manuscript describing the students contribution to the work and acknowledging the contribution of the other author(s). The signatures of the Supervising Committee which precede all other material in the Doctoral Dissertation attest to the accuracy of this statement.

August 2024

FRIEND OR FOE? EVALUATING SENSOR-BASED INFORMATION SIDE-CHANNELS AND COVERT COMMUNICATION CHANNELS ON MODERN WEARABLE DEVICES

Raveen Wijewickrama, Ph.D.
The University of Texas at San Antonio, 2024

Supervising Professor: Murtuza Jadliwala, Ph.D.

The proliferation of smart wearables, such as smartwatches, fitness bands, and smart-glasses, has witnessed exponential growth in recent years, providing users with diverse capabilities beyond those of traditional smartphones. These wearable devices are equipped with a wide array of on-board sensors, including motion sensors (such as accelerometers and gyroscopes) and user interaction sensors (e.g., microphones). These sensors convert various physical, biological, and environmental inputs into measurable electrical signals, enabling applications ranging from simple step counting and sleep time monitoring to complex tasks such as continuous activity recognition and on-body authentication. Despite their immense utility, these sensors can be exploited as private information side-channels, inadvertently exposing users to privacy risks if exploited by malicious entities. To tackle these privacy challenges, this dissertation investigates realistic privacy attacks through sensory side-channels on wearables and proposes robust defense and mitigation techniques. Furthermore, it explores secure applications and communication protocols in smart wearables enabled by these sensors. The first research effort of this dissertation explores a potential secure application of motion sensors in wrist-wearables as a handwriting-based user authentication mechanism. It assesses the feasibility and practical deployability of existing authentication techniques that utilize motion sensors on wrist-wearables. The investigation employs data collected from human subjects in realistic and unconstrained settings, highlighting potential challenges that hinder widespread adoption of such authentication methods in mainstream mobile applications and services. The second work in this dissertation explores the possibility of using motion sensors and embedded vibration motors in wrist-wearables to establish a low bandwidth secure side-channel for

communication between smartphones and smartwatches, leveraging the human skin as the communication medium. A novel frequency modulation technique is introduced to encode and decode bits as vibration pulses which are then sensed via the motion sensors, demonstrating the feasibility of such communication methods. The third work in this dissertation explores the practicality of handwriting inference using motion sensors in wrist-wearable devices, considering various writing forms such as pencil and paper, finger, whiteboard, and air-writing. This research work examines and empirically evaluates existing handwriting inference frameworks with data collected from realistic and natural handwriting settings. Research results from these investigations demonstrate that due to the highly varying nature of handwriting from person to person, wrist motion sensor-based inference attacks are unlikely to pose a substantial threat to users of current consumer-grade smartwatches and fitness bands. The fourth and final research effort in this dissertation investigates the emerging use of motion sensors in smart-headphones and explores the potential risks of keystroke inference using motion and audio data collected from such upcoming smart wearables. Specifically, a novel keystroke inference framework is developed, which utilizes both acoustic and motion data available from modern headphones to infer user keystrokes on external keyboards. The proposed inference framework is then experimentally evaluated for various keyboard types under different realistic environmental conditions using custom-built and commercial headphone hardware. These results highlight the effectiveness and some limitations of the proposed keystroke inference approach in real-world scenarios, demonstrating the potential privacy risks associated with sensor-equipped consumer electronics such as headphones. By addressing these four research directions, this dissertation aims to enhance the security and privacy of wearable devices, mitigate potential side-channel attacks, and pave the way for future research in secure wearable technology applications and communication protocols.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

In recent times, the digital landscape has been reshaped by the emergence and adoption of smart wearables, which include devices such as smartwatches, fitness bands, smart-rings and smart-glasses. These wearable devices have offered users many capabilities which have been only limited to smartphones and more recently these capabilities have further advanced even beyond traditional smartphone features. At the heart of wearable devices' diverse capabilities lies an extensive array of on-board sensors, which can be broadly grouped into four categories: (i) motion and orientation sensors, (ii) health and biometric sensors, (iii) environmental sensors, and (iv) user interaction sensors. The motion and orientation sensor category encompasses accelerometers, which measure linear acceleration; gyroscopes, which gauge angular velocity; and magnetometers, which assess magnetic fields to determine orientation. In the next category, biometric sensors, such as fingerprint or iris detectors, have traditionally been used for authentication in wearables while recent advancements have introduced more sophisticated health-centric sensors, including heart rate monitors, ECG (Electrocardiogram) sensors that capture the heart's electrical activity, and pulse oximeters that determine blood oxygen saturation levels. Environmental sensors, on the other hand, comprise the GPS for location tracking, barometers for atmospheric pressure measurement, and ambient light sensors that evaluate surrounding luminance levels. Lastly, for seamless user-device interaction, wearables are fitted with microphones, cameras, and proximity sensors (detect the proximity of external objects). While all these sensors play a pivotal role in applications ranging from basic phone calls to intricate tasks like on-body authentication, they inadvertently open up potential avenues for privacy breaches. Malicious entities could exploit these sensors as side-channels to glean private information, presenting a critical challenge to user privacy in the wearable technology domain. To address the privacy concerns associated with smart wearables, this dissertation evaluates the potential risks of privacy breaches via sensory side-channels, with a focus on motion sensors, especially under practical and realistic usage scenarios and environmental conditions. The findings of this research, sheds light on the likelihood of private data inference attacks

leveraging motion sensors in smart wearables followed by defense and mitigation techniques to safeguard from such attacks. Furthermore, this dissertation highlights advancements towards establishing secure applications and communication protocols in wearables, made possible by these motion sensors.

The rest of this dissertation is organized as follows. First, an an overview of the work that has been completed as part of this dissertation is provided. The subsequent four chapters outline the detailed findings of the completed research work followed by concluding remarks in the end.

1.1 Practical Applications and Communication Protocols for Wearables: Feasibility of Wrist Motion-Based User Authentication from Handwriting

The popularity of smart wrist wearable technology (e.g., smartwatches) has rejuvenated the exploration of dynamic biometric-based authentication techniques that employ sensor data from these devices. Despite the progress demonstrated by the scientific community, research in this area has not successfully transitioned to practice, and we are yet to see a mainstream user-authentication product based on a dynamic biometric such as handwriting/hand gestures captured using commercial wrist wearables. This work undertakes an investigative analysis to further explore why that is the case. We accomplish this by studying the feasibility and practical deployability of handwriting-based authentication techniques in the literature that utilize motion sensors on-board wrist wearables. We conduct this analysis by replicating four state-of-the-art and representative handwriting-based authentication schemes that employ wrist motion data, in order to test their viability in realistic hand-writing/gesture scenarios. By using data collected from actual human subjects in an unconstrained fashion, we comparatively evaluate the performance of these schemes with well-defined usability and security metrics. Our experimental results show that some of the tested schemes perform considerably well in practice, and are promising. However, our results show that they do suffer from several practical user-dependent and technique-specific challenges that act as roadblocks towards their wide-scale adoption in mainstream applications.

1.2 Practical Applications and Communication Protocols for Wearables: Efficient Vibration-Based Communications Over Human Body Using Motion Sensors

Recent growth in popularity of mobile and wearable devices has re-ignited the need for reliable and stealthy communication side-channels to enable applications such as secret/PIN sharing, co-location proofs and user authentication. Existing short-range wireless radio technology such as Bluetooth/BLE and NFC, although mature and robust, is prone to eavesdropping, jamming and/or interference, and is not very useful as a covert communication side-channel. This work designs and implements *SkinSense*, a vibration-based communication protocol which uses human body/skin as a communication medium to create a low-bandwidth and covert communication channel between user-held mobile and wearable devices. *SkinSense* employs a novel frequency modulation technique for encoding bits as vibration pulses and a spectrogram-based approach to decode the sensed motion data (corresponding to the encoded vibration pulses) to reconstruct the transmitted bits. *SkinSense* is comprehensively evaluated for a variety of operational parameters, hardware setups and communication settings by means of data collected from human subject participants. Results from these empirical evaluations demonstrate that *SkinSense* is able to achieve a stable bandwidth of up to 6.6 *bps*, with bit error rates below 0.1 in our custom hardware setup, and can be employed as a practical communication side-channel.

1.3 Sensory Side-Channel Attacks on Wearables: Handwriting Inference Using Wrist-Based Motion Sensors Revisited

Several recent research efforts have shown that privacy of handwritten information is vulnerable to inference threats that employ motion sensors commonly found on wrist-wearables (e.g., smart watches and fitness bands) as information side-channels. While the adversary model in these earlier efforts have been reasonable and the proposed inference (or threat) frameworks themselves are practical and have technical merit, the related empirical evaluations suffer from several signifi-

cant shortcomings, such as, use of specialized sensor hardware and highly constrained or restrictive experimental procedures, to name a few. As a result, it is hard to estimate the practical feasibility of these threats from existing research results in the literature, and thus, the extent to which end-users must be concerned about the possibility of such attacks in real-life. To answer the above question, this work replicates some of the well-known wrist motion-based handwriting inference frameworks in the literature in order to (re)evaluate their success or accuracy in natural, unrestricted handwriting scenarios and settings by employing commercially available wrist-wearables. The results of these extensive replication and (re)evaluation studies highlight several characteristics in motion data corresponding to natural handwriting scenarios, which were either not observed or ignored by earlier efforts, and contribute to poor inference accuracy of the corresponding frameworks. In summary, this research observes that accurate and practical handwriting inference using motion data from consumer-grade wrist-wearables is difficult primarily due to unique and/or inconsistent handwriting behavior observed in natural writing.

1.4 Sensory Side-Channel Attacks on Wearables: Keystroke Inference Using Multi-Sensor Data from Headphones

Headphones, traditionally used solely for audio playback, have evolved to include advanced sensors such as high-definition microphones and accelerometers to enable a variety of new features and applications. While these sensory enhancements in “smart" headphones significantly improve the overall user-experience, they also expose them to new security vulnerabilities. In this work, we explore the feasibility of a novel private data inference risk, namely, inferring user keystrokes on external keyboards through sensory side-channels on modern headphones. Specifically, we develop a novel keystroke inference framework, called *OverHear*, which utilizes both acoustic and motion data available from modern headphones to infer user keystrokes on external keyboards. The motion (accelerometer) data, while not sufficiently detailed for individual keystroke identification, aids in clustering key presses by hand position. Concurrently, the acoustic data undergoes analysis to extract Mel Frequency Cepstral Coefficients (MFCC), aiding in distinguishing between differ-

ent keystrokes. This information is then applied to machine learning models to predict keystrokes, with accuracy enhancements provided by dictionary-based word prediction methods. We experimentally evaluate the efficacy of *OverHear* for various keyboard types under different realistic environmental conditions using both custom-built and commercial headphone hardware. Our experimental results show that *OverHear* was able to achieve a top-5 key prediction accuracy of around 80% for mechanical keyboards and around 60% for membrane keyboards, with a top-100 word prediction accuracy of over 70% for all keyboard types. These results highlight the effectiveness and some limitations of the proposed keystroke inference approach in real-world scenarios, demonstrating the potential privacy risks associated with sensor-equipped consumer electronics such as headphones.

CHAPTER 2: PRACTICAL APPLICATIONS AND COMMUNICATION PROTOCOLS FOR WEARABLES: FEASIBILITY OF WRIST MOTION-BASED USER AUTHENTICATION FROM HANDWRITING

**The contents of this chapter and the reported experimental results have previously been published in the Proceedings of the 14th Conference on Security and Privacy in Wireless and Mobile Networks (WiSec, 2021), co-authored by Raveen Wijewickrama, Anindya Maiti and Murtuza Jadliwala (in that order). The contents of the published manuscript has been reproduced here with revisions.*

2.1 Introduction

The popularity of wrist wearables such as smartwatches has soared over the past few years. In addition to being a fashionable accessory, a large set of integrated on-board sensors enables wrist wearables to offer a wide-variety of applications besides timekeeping. This includes simple applications such as step counting and sleep time monitoring, to much more complex tasks such as continuous activity recognition [174] and personalized health monitoring [76].

More recently, the research community has been strongly pursuing the idea of employing wrist wearables, and the diverse set of on-board sensors, for (continuous) user identification and authentication tasks [45, 46, 51, 73, 74, 76, 99–101, 111, 143, 165, 176, 201]. In addition to a rich set of available sensor modalities, its position on a user's wrist makes these smart wrist wearables alluring for user authentication type of tasks, especially those based on dynamic biometrics. This is because wrists are actively used in carrying out a variety of day-to-day tasks and the associated wrist movements are unique from person to person due to distinct physiological and kinesiological differences. As a result, an analysis of wrist movements using data from wrist wearable motion sensors (e.g., accelerometer and gyroscope) can provide insight into the various user behavior or activity based authentication modalities. Some recent research efforts in this direction include gait-based authentication [46, 165], authentication based on touch input characteristics [51, 143, 176], gesture-based authentication [100, 101, 201], and handwriting-based authentication [45, 73, 74, 76, 99, 111].

This work specifically focuses on handwriting motion-based authentication mechanisms that are enabled using modern wrist wearable devices equipped with high precision motion sensors. The presence of unique characteristics in a person's handwriting along with the presence of unique wrist movements have made handwriting-based authentication using wrist-wearables, a useful application of wrist-wearable technology to investigate. Besides serving as a convenient primary or secondary (continuous) authentication mechanism for personal use (or for authenticating other devices), such a handwriting motion-based authentication technique could also serve other useful purposes. State-of-the-art research efforts in the literature [45, 73, 74, 99, 111] that employ handwriting-related wrist motion for authentication have not only investigated different practical handwriting scenarios, such as in-air handwriting and writing on a paper using a pen/pencil, but have also demonstrated compelling performance and accuracy results for successful user authentication using this modality. Furthermore, most of these efforts utilized consumer-grade wrist wearables, making them practicable and easily adoptable for real-world applications. Despite these favorable outcomes, research in this area has not successfully transitioned to practice. We are yet to see a successful mainstream mobile/wrist wearable application for handwriting-based user authentication, either as a primary or a secondary/continuous authentication factor. This begs the following questions: *what is preventing these state-of-the-art handwriting-based authentication frameworks from being adopted in mainstream mobile applications and services? Is that because they do not perform well outside of the controlled operating conditions and experimental settings used in their evaluations? And even if they do perform generally well, is their performance at a level required for being successful as a robust user authentication scheme in the wild, similar to other popular (static) biometric-based approaches (e.g., fingerprint)?* Answering these questions is critical for understanding the reasons behind the lack of success and/or adoption of handwriting-related motion as a modality for user authentication in mainstream mobile and wearable applications, and for understanding how/if these challenges can be overcome.

Accordingly, we conduct an investigative analysis in this research work to answer the above feasibility related concerns, and discuss the practicality of handwriting-based authentication using

wrist wearable motion sensor data. We select four state-of-the-art representative handwriting-based user authentication frameworks from the literature, and comprehensively analyze all of them using uncontrolled and unconstrained handwriting data collected from a diverse set of human subject participants in a variety of realistic writing settings. We select these frameworks due to their varying, yet representative approaches both in the design of the respective frameworks as well as the handwriting modalities. Specifically, one of the frameworks utilize carefully engineered features in both time and frequency domains along with traditional machine learning techniques such as SVM [73], while another framework only uses a basic set of temporal domain features along with a Naive Bayes classifier [45]. The third framework does not extract any features, and instead leverages on deep neural networks (DNN) [74]. Lastly, the fourth framework uses frequency domain features with a Logistic Regression model [99]. An additional insight here is that it is not appropriate to directly compare the evaluation results reported by these research efforts, primarily because the data for those experiments were collected from different groups of users (human subjects), under different conditions and setups. For an equitable comparison, we need to evaluate these schemes with data from the same group of subjects under the same experimental conditions, which we accomplish in this work. More specifically, comprehensively investigate the performance of the above mentioned four authentication frameworks against a variety of experimental conditions and parameters, including, authentication window size, training data size, writing settings (e.g., pencil, finger and air writing), robustness under environmental noise, and their ability to perform in true free-form writing.

2.2 Related Work

User authentication has been an extensively researched topic in the literature, with a diverse body of contributions. However, here we only outline research efforts related to unique user movement-based authentication – a form of dynamic biometric – captured by means of motion sensors on a variety of mobile/wearable devices, as it is more relevant to the proposed work. A more comprehensive survey on user authentication, which includes other static and dynamic

modalities and biometrics, can be found in [171, 173].

Authentication techniques that employ motion sensor data from mobile/wearable devices to construct unique user movement related biometrics have been extensively studied in the research literature [45,46,51,73,74,76,99–101,111,143,165,176,201]. These efforts have either employed commercial off-the-shelf (COTS) mobile/wearable devices, including smartphones, smartwatches and smart rings, or other types of specialized devices (e.g., smart pens and hand gloves) equipped with motion sensors such as accelerometers and gyroscopes. Some of the early schemes used unique gestures made by users while holding a smartphone or while wearing a smart wrist wearable for user identification and authentication, where the gesture-related hand/wrist motion was captured by the mobile device's accelerometer and gyroscope sensors [100, 101, 201]. Other forms of contextual body movements such as users' natural gait (walking) based motion [52, 136], motion/orientation corresponding to how users hold their phone (in their hand/s) [58, 176], motion corresponding to how users answer a phone call [146] and fine-grained hand movements such as taps or typing [33,49] on the phone captured using COTS mobile device motion sensors, have also been used to design dynamic biometrics for user authentication.

An advantage of employing such modalities (e.g., tapping, typing, etc.) as a biometric is that it can be used to continuously authenticate users in a real-time fashion. Hand movements observed during handwriting is another suitable modality for such continuous user authentication, and it has also received significant attention. We first outline authentication schemes in the literature that have employed specialized, non-standard motion-capture devices to capture handwriting related wrist/hand motions. For instance, Bashir et al. [26] proposed an authentication scheme by using a smart pen device to capture the accelerometer time-series data corresponding to two different types of handwriting scenarios: (i) writing in the air, and (ii) writing on a paper. In another research, Lu et al. [111] used a custom glove with built-in motion sensors located on the fingertip to capture in-air handwriting movements. In addition to the fact that it required a custom data collection hardware to operate, this scheme only considered writing scenarios in which users wrote a unique passcode/PIN for every authentication attempt. This severely limits its practicality, especially for

continuous authentication.

In the direction of handwriting motion based authentication schemes that employ commercially available wrist wearables, Buriro et al. [34] proposed a framework similar to [111], but by using a consumer-grade smartwatch instead of a custom glove, and the users sign their name in the air to authenticate themselves. Huang et al. [82] proposed an authentication scheme using a smartwatch gyroscope in which they evaluated three gestures written/drawn in the air, namely, a star, a number eight and a triangle. A significant drawback of their approach is that it has been shown to work only for a fixed set of hand signs, and it is not evident whether their framework can be extended or generalized to normal human handwriting or signatures. As before, this limits its applicability in continuous authentication scenarios.

The following four schemes [45, 73, 74, 99] in the literature attempt to overcome the two major shortcomings of earlier handwriting motion based authentication schemes: (i) reliance on specialized motion sensing hardware, and (ii) employing fixed writing patterns or symbols for authentication. Given the diverse set of handwriting scenarios and settings considered by these schemes, and the use of COTS data collection hardware (e.g., smartwatches) instead of specialized hardware, makes them the most promising candidates for adoption by users in a practical and continuous handwriting-based user authentication application. This is also the main reason why we shortlist them as prime candidates for a mainstream continuous authentication application in this category, and perform a rigorous comparative performance analysis of them in realistic usage settings. More detailed descriptions for each of them are outlined later in section 2.4.

2.3 Research Goals

We organize our research goal - studying the practical feasibility of state-of-the-art handwriting-based user authentication schemes that employ motion sensor data from wrist wearables - as a set of four targeted research questions (RQ1 through RQ4). These questions focus on studying the performance of these schemes under varying writing styles and modalities, ambient conditions and written content, and collectively provide a comprehensive performance analysis. We fix the

experimental setup and evaluation parameters (section 2.5), including the employed performance metrics, such that it enables us to obtain insightful answers to these questions.

- *RQ1 – How does handwriting-based user authentication using wrist motion data perform in real, unconstrained writing scenarios?*

Ideally, an effective user authentication scheme should not place any unreasonable constraints on how users should write for the scheme to work effectively. Any undue constraints or changes to the users' usual writing habits or styles will result in low usability and adoption of the scheme. This research question aims to investigate the constraints or restrictions a user authentication scheme (under investigation) places on users' writing style and how does it perform when those constraints are dropped and users are allowed to interact with the scheme using their own everyday writing style. We accomplish this by collecting handwriting-related data in a fully unconstrained setting, where participants are allowed to write in their usual habit or style. We analyze the performance of the schemes under investigation using this unconstrained writing data, and also comparatively evaluate them based on the amount (length) of writing data required for enrollment and authentication.

- *RQ2 – How does handwriting-based user authentication using wrist motion data perform for different writing modalities?*

Depending on a user's context (time, location, etc.) and the writing instrument they are interacting with, the activity of writing can assume different modalities, for example, writing on a paper with a pen, writing on a smart tablet screen using a finger, and gesture writing in the air. Ideally, an effective authentication scheme should work across multiple writing modalities, otherwise, its applicability is restricted to a limited set of user-contexts. This also limits the applicability of the scheme for continuous authentication. Thus, this research question aims to investigate how the schemes (under investigation) perform for a variety of commonly observed writing modalities. We accomplish this by collecting data for a variety of writing modalities in a fully unconstrained

setting, such as, traditional pen/pencil writing on a paper, finger writing on a touchpad screen and gesture writing, and analyzing the performance of the schemes across these modalities.

- *RQ3 – How does handwriting-based user authentication using wrist motion data perform under different types of ambient noise?*

Accelerometers and gyroscopes on modern mobile devices are highly sensitive hardware/software systems. An advantage of high sensitivity is that these sensors can sample small (imperceptible) changes in motion, however, a related disadvantage is that the sampled data from these sensors is easily impacted by ambient noise. An important characteristic of an effective and practical user authentication scheme is robustness against noisy inputs. A robust (against noise) authentication system typically produces very low false negatives (i.e., has high recall), resulting in higher usability and adoption among users. This research question aims to investigate how the user authentication schemes (under investigation) perform when the sampled test motion data (or authentication requests) is noisy. We accomplish this by introducing noise in our collected handwriting related motion data, and analyzing the performance of each of the four authentication schemes on this noisy data. For this analysis, we consider different types and sources of noise commonly encountered during writing, such as writing on a table with a vibrating smartphone and writing while inside a moving vehicle.

- *RQ4 – How does handwriting-based user authentication using wrist motion data perform under user-dependent, free-form writing?*

In RQ1, we analyzed the performance of handwriting motion based user authentication by removing constraints on *how* users should write. In the same vein, in this research question, we investigate if constraints or restrictions on *what* the users write impact the performance of the schemes under investigation. We accomplish this by evaluating these schemes on free-form handwriting data collected from participants in a fully unconstrained setting, i.e., motion data corresponding to handwriting consisting of both upper-case and lower-case letters (and not limited only to specific words with specific lengths or specific letter cases). The main motivation of not restricting users to

write a specific text for authentication is that it limits the applicability of the scheme in continuous authentication scenarios, since continuous authentication is about being able to passively authenticate based on whatever is being written by the user and not asking them to actively authenticate by writing specific keywords. Further, the handwritten letter/word/text cannot not be equated to a set password as nothing conceals handwritten text from onlookers, unlike passwords typed on a PC that are by default concealed on the screen. Ideally, a handwriting-based authentication scheme should be agnostic of the letter/word/text written at any given instance of the authentication attempt.

2.4 Experimental Setup

In this section, we first outline a system model (fig. 2.1) that generically describes the authentication framework of all the four handwriting-based authentication schemes that we plan to comparatively evaluate in this work. Following that, we provide specific modeling and implementation details for each of the four schemes. Then, we provide details of our data collection procedure and the metrics and benchmarks used in our analysis.

2.4.1 System Model

In any handwriting-based authentication scheme that employs motion data from wrist wearables, users would first go through an enrollment phase in which they supply training data to the system by performing handwriting tasks while wearing a wrist wearable device on their writing hand. The motion sensor data collected from the wrist wearable during the enrollment procedure is then used to build a unique profile for the user, which is later tested against when an authentication attempt is made. This "unique profile" for each user typically takes the form of a classification function that is trained using the target user's data. The raw motion data collected during the enrollment phase is typically first pre-processed to remove/reduce noise and then utilized for the feature extraction task. The extracted feature set for the authentic user(s) is then used to train a classification model, sometimes along with labeled non-authentic user data to prevent over-fitting. As such,

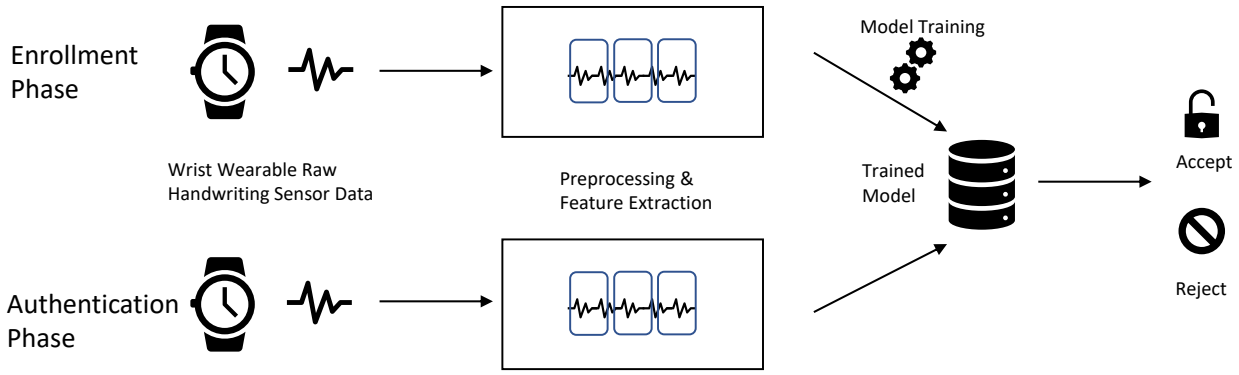


Figure 2.1: A generic system model for handwriting motion based authentication schemes.

Table 2.1: Overview of the four representative authentication frameworks used in our study.

	Description	Device	Sensors & Frequency	Features	Techniques
M01 [73]	Handwriting authentication using paper and pencil as writing tools	Android Smartwatch	Accelerometer, Gyroscope at 100Hz	Both temporal and frequency domain features	SVM, MLP
M02 [45]	Handwriting authentication using paper and pencil as writing tools	LG G Watch	Accelerometer, Gyroscope at 20Hz	Only temporal domain features	NB, MLP
M03 [74]	Handwriting authentication using paper and pencil as writing tools	LG Urbane 2 Smartwatch	Accelerometer at 100Hz	No specific features extracted. Uses raw sensor data.	DNN
M04 [99]	Signature verification using stylus tablet as writing tools	Microsoft Band	Accelerometer, Gyroscope at 62Hz	DTW distance based features in frequency domain	Logistic Regression

the classification model is a binary classifier where it will attempt to classify an authentication attempt as either authentic (valid users who enrolled with the system) or non-authentic (anyone else). In order to authenticate with the system, an enrolled user is required to perform a handwriting task which will then be compared with the user's profile, and an authentication decision is made by the system based on the output of the trained classification model.

2.4.2 Implementations

We implemented the four representative handwriting-based authentication frameworks (summarized in table 2.1) using Python 3.6, while utilizing scikit-learn [142] and TensorFlow [1] with Keras for the corresponding machine learning and deep learning-based models employed by those frameworks. Individual implementation details of each framework's classification model are described next, labeled from M01 to M04. These implementations very closely follow the original works, and reuse their code whenever available.

M01 [73]

In this framework, the raw sensor data is first pre-processed by applying a low pass filter to eliminate outliers. The continuous motion data stream is then divided into smaller windows of data which are then used for feature extraction across four feature categories. First, each of the sensor axes x, y and z, along with their magnitudes, constitute as the first feature category. The second feature category includes Fast Fourier Transform (FFT), Power Spectral Density (PSD), and squared magnitude or power of FFT coefficients. The third feature category includes Discrete Cosine Transform (DCT), Discrete Sine Transform (DST), Real-Valued Fast Fourier Transform (RFFT), eigenvectors and gradients calculated for each of the x, y and z axes. The fourth and final feature category is pitch and roll [193]. Different statistics such as mean, standard deviation, variance, computed on values in these four feature categories are then combined to create an intermediate feature vector, which is further normalized and selectively shortened before being employed in the classification model training/testing tasks, as explained next. The intermediate feature set comprising of 182 features obtained in this fashion is then normalized and ReliefF [96] feature selection scheme is applied to select top-30 features from each of the accelerometer and gyroscope data, resulting in 60 features in the final feature vector. A Support Vector Machine (SVM) binary classifier is then trained for each user using this final feature vector with GridSearch based hyper-parameter tuning. The model is trained using a part of the authentic user's data and data from half the number of other users in the dataset. In the model testing phase, the remaining part of the authentic user's data and the data from the other unseen users are used.

M02 [45]

Instead of a sliding window approach, the M02 approach uses the entire recorded activity signal for feature extraction. In contrast to M02 [45], which employed short and specific transcription tasks during data collection, our common data collection process (outlined in section 2.4.3) involved long handwriting tasks in which users wrote continuously for few minutes. Thus, we utilize a sliding window approach to divide the raw signal (obtained during our data collection)

into multiple smaller windows which are then independently used for feature extraction. For each window, the mean, standard deviation and average absolute difference for all 3 axes are calculated and used as features. The peak positive value and the peak negative value are also extracted from each axis and used as features. The average resultant acceleration, which is the mean of square root of sum of the squared values of x, y and z axes, is also computed and used as a feature. A combination of these features results in a vector of 16 features for each sensor type and a final feature set of 32 features. A Naive Bayes model is then trained and tested in a per-participant fashion, similar to M01.

M03 [74]

The M03 approach employs a Deep Neural Network (DNN) which is created using two consecutive 1D convolutional block layers, followed by a bi-directional LSTM layer for each sensor axis, which are then concatenated and fed into a fully connected layer. Each of the convolutional blocks consists of two 1D convolutional layers with 32 and 64 filters, respectively, with a kernel size of 3 and a "relu" activation followed by a batch normalization layer, a max pooling layer of size 2 and a dropout layer with a 0.5 dropout rate. The output of each convolutional block (per axis) is then sent through a bidirectional LSTM layer with 10 neurons. Each of the output from each axis/convolutional block is then concatenated, sent through another layer of dropout with a rate of 0.5 before finally being input to a dense or fully connected layer with 2 neurons. The dense layer consists of 2 neurons which represents the number of classes in the classification problem along with a softmax activation. The model is fitted using Categorical Cross Entropy loss with Adam [95] optimizer. As DNNs require large amounts of data to train an effective model, M03 [74] uses a data augmentation step in which synthetic data is created using the existing data, in order to increase the training data size. In our experiments, we implement the same data augmentation step where random windows of data are selected to generate new synthetic data, which is repeated till the dataset becomes 4 times the size of the original set [186].

M04 [99]

As the original work in M04 is designed for signature verification, entire motion data corresponding to each signature sample is taken as an input for feature extraction. In order to adapt it to a handwriting authentication scenario, we extract small windows from the data similar to the previously described frameworks. The windows are then normalized and transformed into the frequency domain by Discrete Cosine Transform (DCT). The first 20 DCT coefficients are then extracted and fed as input into the feature extraction phase. In feature extraction, a set of authentic user data windows are selected as templates for the authentic user. The remaining authentic user data was then split into training and testing sets. The training set of an authentic user and half the number of non-authentic users are selected for model training. For each of the sample windows in the training set, the DTW score between each window and the authentic user template windows are computed. The DTW score is calculated individually for each axis of a query sample and the templates in the template set, the lowest score obtained between a template and a query sample is then added to the feature vector (the feature vector consists of 6 features, one for each axis of each of the sensor). The training data created in this fashion is then used to train a Logistic Regression based classification model.

2.4.3 Data Collection

We recruited a demographically diverse set of 21 participants for data collection in three popular writing scenarios, namely *pencil writing*, *finger writing* and *air writing*, utilizing seven participants per scenario. The participants were recruited via advertising flyers around the university and their ages range between 18 to 30 ($\sigma = 4$). The participants were either given a Sony Smartwatch 3 or a LG Watch Urbane to wear on their writing hand (right hand) for the entire experiment. A Samsung GT-N5110 Android tablet was used to display the writing content. An Android Wear app which records accelerometer and gyroscope data at 200 Hz from the smartwatch was developed along with an app for the tablet to display the writing content. In the pencil writing experiment, the participants were provided with a pencil and paper and a table to write on. They were also given

a height-adjustable chair, which they could adjust based on their individual/personalized comfort and writing positions. The finger writing task was carried out in a similar way, except that the participants were given an area on the screen of the android tablet for the writing task. In the air writing setting, the participants were given a chair with no obstructions for the free movement of their writing hand. In all scenarios, the participants were asked to stick to their natural handwriting styles and pace, and no time limit was given to complete the tasks. The participants were asked to write the content displayed on the tablet on the corresponding writing surface. For example, if the tablet displays the word "test", then a pencil writing participant would write the word "test" on the paper given to them using a pencil. In each of the writing scenarios, each participant wrote English alphabets displayed in random order. Each alphabet was written 10 times, totaling to 260 alphabets for lowercase and 260 for uppercase alphabets followed by 40 words. These data collection procedures were approved by our university's Institutional Review Board (IRB).

We also evaluate robustness using a setup where the attacker tries to falsely authenticate by mimicking the handwriting of an authentic user. For this mimicking/impostor attack evaluation, 5 participants (potential victims) were recruited. Their handwritten text of six random Harvard sentences [155] and a video of their writing (which includes their hand/wrist movements and wrist positioning) were recorded.

An attacker was then asked to observe the victims' text and video for practice, i.e. the paper with the handwritten text of the victim or the screen recording of the text written on the tablet surface for pencil writing and finger writing scenarios respectively. In addition, the attacker also carefully watched/observed the video taken of the victim's handwriting before performing the final impostor attack. We only considered the finger writing and pencil writing settings for this analysis because air writing does not produce physically recorded written text that can be reviewed by an attacker for mimicking purposes.

2.4.4 Benchmarks and Evaluation Metrics

We comparatively evaluate the authentication frameworks using the widely accepted metric of Equal Error Rate (EER), which is the point at which the False Rejection Rate (FRR) equals the False Acceptance Rate (FAR) [61]. FAR is the probability an authentication system incorrectly authenticates an unauthorized user or impostor as an authorized user. On the other hand, FRR is the probability that an authentication system incorrectly rejects an authorized user as unauthorized (failed authentication). These metrics depend on the decision threshold of the classification system. A strict decision threshold would imply lower number of false positives, i.e. the probability of an unauthorized user or impostor getting authenticated is lower, but actual users may get rejected for slight anomalies in their writing with higher probability. On the other hand, with a relaxed threshold, there is a higher probability of an unauthorized user or impostor getting authenticated and the probability of authorized users getting rejected for slight anomalies in their writing would be lower [110]. This trade-off between FAR and FRR is application specific. A high security application in which an entry of an unauthorized user is disastrous must require a very low FAR regardless of the possible inconvenience that authorized users may experience [86]. For reporting comparative analysis on the framework performance, we use EER as the primary metric, and employ the FAR and FRR metrics wherever relevant. In general, a lower EER value is a good indicator of a balanced and robust authentication framework.

2.5 Performance Evaluation

All the four authentication frameworks are evaluated per participant, by labeling the target participant's data as authentic and all other participants as non-authentic or impostor when training and testing the models. The overall evaluation results are then averaged across all the target participants. We present EER values for comparison between the handwriting-based authentication schemes, utilizing realistic data collection procedure outlined in section 2.4.3. We conduct our comparative performance evaluation with respect to varying experimental parameters, as categorized next.

2.5.1 The Effect of Authentication Window Size

The window size parameter is intuitively the continuous window of time a user needs to write in order for the authentication framework to effectively perform its functionality. Or in other words, it is the number of continuous motion sensor data samples of handwriting activity used by the authentication framework for the classification task. The window size is a vital experimental parameter because a smaller window may not contain sufficient information for successful authentication, while a larger window size will require a longer time to process and for the authentication task to complete. If an authentication system requires users to write for long periods of time for every authentication attempt, it can result in usability issues as well. In our experiments, we test window sizes ranging from 15 to 60 seconds for both the enrollment and the authentication phases, and we present the results next.

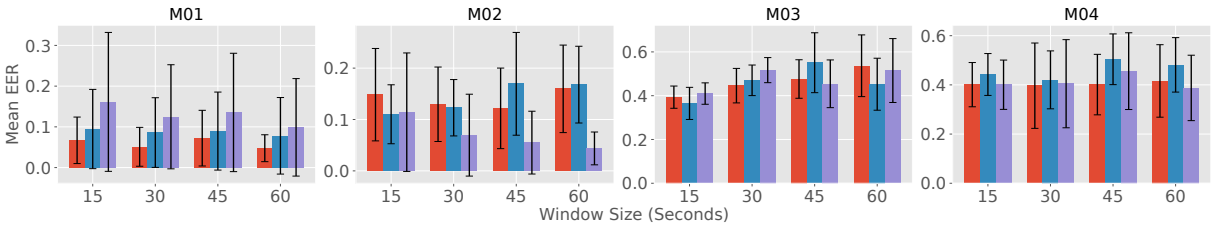
In the M01 window size analysis shown in fig. 2.2a, we see that while all the window sizes (15, 30, 45 and 60 seconds) produce a mean EER of less than 0.10, a window size of 60 seconds performs slightly better (lower EER) for both pencil and finger writing scenarios. In the air writing scenario, the highest EER is recorded for the 15 second window at 0.16 ($\sigma = 0.17$) and the lowest/best EER is recorded again for the 60 second window which is 0.10 ($\sigma = 0.12$). Due to the additional freedom when writing in the air, we observe that users tend to write larger characters which in turn results in lower number of characters per window. Thus, for the air writing scenario, a higher window would capture more characters and more discriminative features across users. Although the difference between the obtained EER values across different window sizes is insignificant in pencil and finger writing scenarios, in the air writing scenario 60 second windows have a clear advantage over the other window sizes with the next lower window size of 45 seconds recording an EER of 0.14. The trends are similar to the results obtained by the authors of M01 at a mean EER of 0.11, especially in the pencil and finger writing scenarios.

In M02 window analysis (fig. 2.2a) for the pencil writing scenario, we observe that the best mean EER is 0.12 ($\sigma = 0.08$) with a window size of 45 seconds. Although the performance difference between window sizes 30 and 45 was low, at 60 second windows the performance

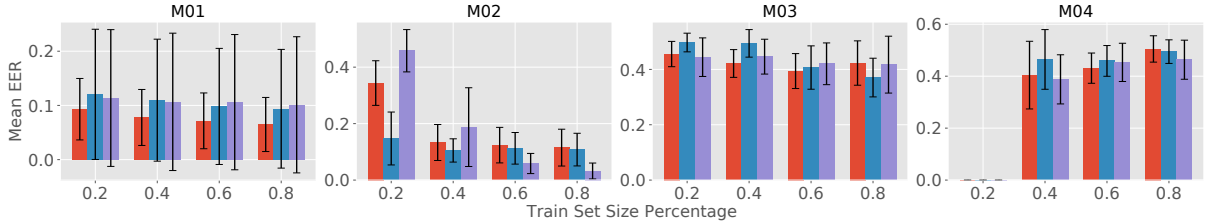
deteriorates to an EER of 0.16. In finger writing scenario, we observe that the best mean EER is 0.11 ($\sigma = 0.05$) at 15 second windows, and it can also be seen that the performance worsens with the size of the window. A possible explanation for the degrading performance at larger window sizes could be that larger window sizes lead to lower number of training examples, which results in an underfitted classification model. In the air writing scenario of M02, we observe that the best mean EER is 0.04 ($\sigma = 0.03$) at 60 second windows and 45 second windows is second best at 0.05 ($\sigma = 0.06$).

As shown in fig. 2.2a, in all 3 writing scenarios M03 produced the lowest EER values when a 15 second window size is used compared to other window sizes. Specifically, the resulting EER values are 0.39 ($\sigma = 0.05$) for pencil writing, 0.36 ($\sigma = 0.07$) for finger writing and 0.4 ($\sigma = 0.04$) for air writing scenarios. In contrast, the experimental results of M03 as reported in the original publication shows that it was able to achieve slightly better results with a mean EER of around 0.3 compared to the results we have achieved. One possible reason why the DNN model of M03 performs better at 15 second windows could be because at a lower window size there would be more training examples which is advantageous when training a DNN model. In the original proposal, a voting based multiple window fusion technique is used after the DNN classifier step to lower the EER to 0.07, but we did not observe significant performance improvement even after such a multi-window voting mechanism. In M04, the mean EERs across all window sizes for pencil writing scenario is around 0.4. In the finger writing scenario for M04, the lowest mean EER for a window size of 30 seconds is around 0.42 ($\sigma = 0.12$). Similar to M01 and M02 air writing scenarios, M04 air writing scenario has the best EER at a window size of 60 seconds which is 0.39 ($\sigma = 0.13$).

In summary, M01 performs the best (EER of less than 0.1) across all schemes in 15 seconds window sizes, while M02 shows the best performance (with an EER of 0.12) for air writing at the same window size. Thus, from a practical perspective, our results show that M01 is more desirable due to its comparatively better performance at lower window sizes, especially for the popular pencil and finger writing scenarios.



(a) Effect of different windows sizes on performance.



(b) Effect of different training set sizes on performance.

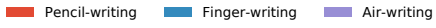


Figure 2.2: Performance based on varying window sizes and train set sizes.

2.5.2 The Effect of Training Set Sizes

We next analyze the effect of training set sizes on the performance of the authentication frameworks M01-M04. To do so, we test each framework with training set sizes between 20% to 80% of the total available user (participant) data. However, the total amount of time and training examples for each user varies due to varying writing speeds. One user may have more characters or words written during a particular window of time compared to another. This also highlights that these authentication frameworks may require varying amounts of data during the user enrollment phase, depending on the targeted accuracy or error-rate thresholds. Requiring larger amounts of training data to achieve a better performing framework is not very convenient from the end-user perspective, thus making the mainstream adoption of such schemes difficult. As can be seen from our experimental results in fig. 2.2b, M01 does not have any significant performance improvement above the training set size threshold of 40% for any of the writing scenarios. Similarly, M02 (fig. 2.2b) also does not have any significant performance improvement above the training set size threshold of 40% for the pencil writing and finger writing scenarios. However, in the air writing scenario, a considerable performance improvement (for M02) is observed when the training set size is set above 40% of the total user data. Specifically, the EER at 40% training set size is 0.19

($\sigma = 0.07$), and when the training set size is set to 60% it dropped down significantly to 0.06 ($\sigma = 0.04$), which further drops to 0.03 ($\sigma = 0.03$) when using 80% of the data for training. This again indicates that due to the extra freedom and the higher time (larger characters) that occurs during air writing, the framework hugely benefits from a having higher amount of training data allowing it to generalize well on the test data.

In the evaluation of M03 and M04 (fig. 2.2b), we observe that both these frameworks still produce mean EERs of over 0.35 across all writing scenarios even with higher training set sizes (The training set size of 20% is not evaluated for M04 as it had insufficient examples for constructing both the required template and training sets). Although M03 with its DNN model could benefit from a larger set of training data, the performance increases observed were minimal, again indicating the need for high training data volumes. The overall poor performance of M04 on the other hand could be indicative of the fact that the specific features that were used by the scheme did not generalize well for the continuous handwriting based authentication task.

In summary, framework M01 and M02 perform considerably better at even lower train set sizes indicating that these frameworks do not require large amounts of training data for successfully operating as an authentication scheme.

2.5.3 Different Writing Settings

Next, we present a detailed evaluation for each authentication framework in each of the three writing scenarios (table 2.2).

Pencil writing

In the pencil writing scenario, our experiments with M01, which uses a SVM binary classifier, demonstrated a mean EER of 0.05 across all participants ($\sigma = 0.03$). The lowest EER we observe for a single participant is 0.01 and the worst EER for a participant is 0.14. Furthermore, 6 out of 7 participants show a mean EER of less than 0.1. M02, with a Naive Bayes classifier, has a mean EER of 0.15 ($\sigma = 0.10$) across all participants for the pencil writing scenario. We also observe

that for 5 out of the 7 participants, the EER is below 0.20. The best M02 EER for a participant is 0.02, while the worst is 0.32. For M03, the mean EER across all participants is 0.39 ($\sigma = 0.05$). The lowest recorded EER for a participant is 0.34 and the highest is 0.47. The best mean EER of framework M04 is 0.40 ($\sigma = 0.09$) for a 30 second window. The lowest EER recorded for a participant in M04 is only 0.34 and the highest is as high as 0.52 in M04. Moreover, only one participant has an EER below 0.40 in M04. In summary, for the pencil writing scenario, we observe that M01 produced the lowest EER compared to all other frameworks.

Finger writing

In the finger writing scenario, M01 has the best EER (compared to all other schemes) of 0.08 ($\sigma = 0.09$), with 6 out of the 7 participants having less than 0.12. The framework M02 shows the second best performance with a mean EER of 0.11 ($\sigma = 0.06$) and a best EER of 0.03. M03 has a slightly higher mean EER of 0.36 for finger writing, in which two of the participants have a mean EER of 0.26 while all other participants' EERs are over 0.40. Lastly, M04 has the worst performance for finger writing with a mean EER of 0.42 ($\sigma = 0.12$) and with only one participant showing an EER below 0.3. Similar to the pencil writing scenario, M01 produces the best performance (lowest EER) for finger writing scenario.

Air writing

In the air writing scenario, M01 has a mean EER of 0.10 ($\sigma = 0.12$) with only one out of the 7 participants recording an EER higher than 0.10. The M02 framework demonstrates the best performance overall out of all the frameworks for air writing with a mean EER of 0.08 ($\sigma = 0.05$), with all except one participant showing an EER of over 0.06. The M03 framework has a mean EER of 0.4 ($\sigma = 0.08$) with only one participant showing an EER of below 0.3, while M04 shows similar performance with a mean EER of 0.39 ($\sigma = 0.06$). In summary, while M01 demonstrates the best performance (lowest EER) for pencil and finger writing scenarios, M02 has the lowest EER for the case of air writing.

2.5.4 The Effect of Sampling Frequency

Even though modern wrist wearables feature highly precise motion sensors, sampling these sensors at a high rate is an energy intensive operation, which significantly impacts the device's battery life, and thus, treated as a critical design factor. Lu et al. [109] demonstrate that higher sampling frequencies of motion sensor data results in considerably higher battery power consumption. Specifically, they show that a rate of 200 Hz consumes 6.3% (per hour) of the battery power of the device on average (tested on a Sony Smartwatch 3) and battery power consumption reduces when the sampling rate is lowered with only 4.4% (per hour) at a 100 Hz sampling rate and 2.0% (per hour) at a 50 Hz sampling rate. In other words, handwriting authentication frameworks that require fine-grained motion data (i.e., sampled at high frequencies) for performing well could adversely impact the device's battery charge (i.e., drains it faster) and requiring frequent battery recharges. This ultimately will adversely impact the usability and adoption of such schemes by end-users. Thus, in this set of experiments we evaluate how the four authentication frameworks M01-M04 perform under motion sensor data sampled at different frequencies. Schemes that perform reasonably well at lower frequencies would obviously be much more energy (battery) efficient, and preferable by end-users.

Our experiments with M03 and M04 show (fig. 2.3) that lower sampling rates produce comparable performance across all the writing scenarios, i.e., their performance does not vary much with change in sampling frequencies. But as seen in fig. 2.3, frameworks M01 and M02 produce significantly worse EERs at lower sampling rates, compared to the original sampling rate of 200 Hz. In summary, none of the analyzed frameworks produce reasonable levels of EERs at lower sampling frequencies. This highlights a significant challenge, especially, towards use of these frameworks in continuous authentication scenarios as it would require periodic sampling of motion sensors, which will in turn impact battery longevity.

Table 2.2: Performance summary (measured in EER).

	M01	**M02**	**M03**	**M04**
Pencil Writing	0.05	0.15	0.39	0.40
Finger Writing	0.08	0.11	0.36	0.42
Air Writing	0.10	0.08	0.40	0.39

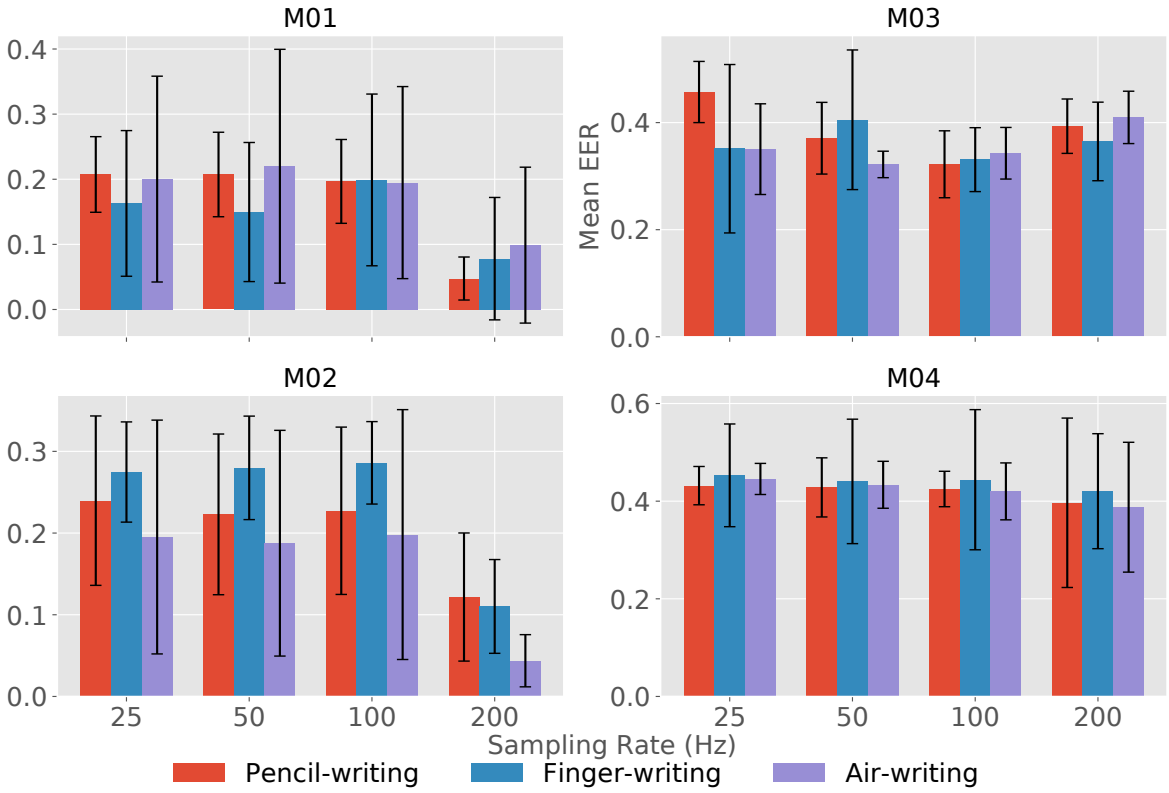


Figure 2.3: Effect of sensor sampling rates on performance.

2.5.5 The Effect of Environmental Noise

Next, we evaluate the impact of environmental noise on the performance of the four authentication frameworks M01-M04. For this, we separately record a few types of background noises that users could encounter during each of the writing scenario. We then superimpose this pre-recorded motion sensor noise over the raw handwriting related motion sensor data obtained from our study participants, prior to using the data for training and testing of the four authentication frameworks. For finger writing and pencil writing scenarios, the motion noise is emulated by placing a vibrating phone on the table (writing surface) at close proximity. For the air writing scenario, the accelerometer noise emulation is obtained from a moving vehicle. After re-performing our experiments with

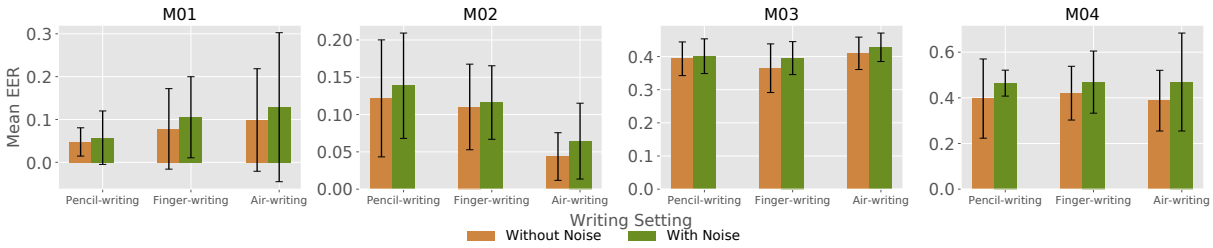


Figure 2.4: Effect of environmental noise on performance.

the noisy data, we observe (fig. 2.4) that all frameworks demonstrate a considerable degradation of performance. We believe that our results are still relatively optimistic as we only introduced one type of noise for each of the writing scenarios. In practice, there could be a combination of various additional environmental noises polluting the device's motion sensor data which will further worsen the performance of these frameworks.

2.5.6 Convenience vs. Security

We also analyze how adaptable each of these frameworks are in terms of their potential target application, i.e. whether they are more suited for a high security application or for a high user-convenience application. High security applications such as access control to government intelligence, military applications or other highly sensitive data could tolerate high FRR, but FAR needs to be kept at a minimum level. Similarly, usability focused applications such as consumer-grade smartphone unlocking which prioritize user convenience, could allow a slightly higher FAR as a trade off in achieving a minimal FRR.

From our experiments with M01 (fig. 2.5a) we see that in the case of pencil writing the EER occurs approximately at 0.5 decision threshold. However, a slight increase in the decision threshold would result in a sudden increase in the FRR (over 0.2), while the FAR only reduces slightly and converges around 0.8. This suggests that if a stricter (higher) decision threshold is chosen with high security in mind, the rate at which unauthorized users may mistakenly gain access decreases only slightly. However, actual users would suffer considerably as they are likely to fail authentication approximately 2 or more out of 5 attempts for decision thresholds above 0.5. Although the low

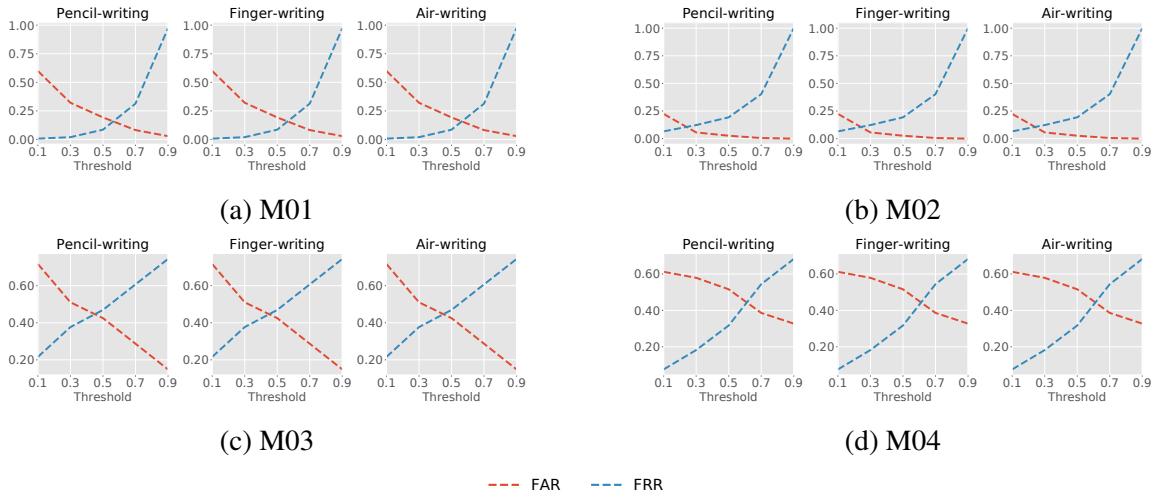


Figure 2.5: False acceptance rates vs. false rejection rates at different thresholds.

EER of 0.05 suggests that M01 is generally a suitable framework for balancing convenience and security, it lacks flexibility towards adjusting the decision threshold. The M01 framework in finger and air writing scenarios show similar characteristics when it comes to the adjustability of the decision threshold. Moreover, as the EER for these writing scenarios lie at approximately 0.10, the room for adjustability is further reduced when compared to the pencil writing scenario. As seen in fig. 2.5b, for all three writing scenarios in M02, the ERR occurs at a decision threshold below 0.3. When adjusting the decision threshold in the pencil writing scenario (EER=0.15), an increase of the threshold from 0.3 to 0.5 drops the FAR from 0.09 to 0.04. However, at the same time, FRR almost doubles from 0.15 to 0.29. A similar pattern was also observed in the finger writing scenario. Taking a closer look at the air writing scenario in M02, we observe that the threshold could be made stricter (up to a 0.5 threshold) in trying to achieve a much more secure system by keeping the FRR under 0.2 and reducing the FAR to less than 0.03. For the M03 framework (fig. 2.5c), adjusting the decision threshold either for security or for convenience would significantly increase FRR and FAR values resulting in an EER over 0.5. Very similar patterns were also observed for the M04 framework (fig. 2.5d). This further demonstrates that M03 and M04 are rather unsuitable for both security and usability oriented applications. In terms of adjustability of the threshold depending on the use case (security or usability), M02 demonstrates better versatility across all frameworks and all writing scenarios.

2.5.7 Comparison with Other Modalities

We next compare the handwriting-based authentication frameworks against other modalities of biometric authentication. The best mean EER among the four handwriting-based frameworks is for M01 in pencil writing scenario (0.05) followed by finger writing (0.08). Framework M02 came in second place with an EER of 0.15 for pencil writing and 0.11 for finger writing. M02 performed the best among all frameworks in air writing scenario with an EER of 0.08, with M01 following behind at 0.1. Evidently, these EER values are much higher compared to the mainstream fingerprint based authentication methods (table 2.3) which can be as low as 0.000022 [54, 202]. On the other hand, the handwriting-based authentication schemes' EER values are equitable to that of other motion sensor based authentication frameworks. Specifically, frameworks based on gait [121] and gesture [201] have EERs of 0.07 and 0.04, respectively. This is not surprising because, while modalities such as fingerprint and iris have matured to become reliable first factor authentication mechanisms, motion based authentication frameworks (including handwriting-based) are primarily being explored as a means for second factor or continuous authentication.

Table 2.3: EER attained by other authentication modalities.

Biometric	EER	Used at Mainstream Consumer Level
Fingerprint [54, 202]	0.00022	Yes
Iris [208]	0.006	Yes
Gait Based [121]	0.07	No
Gesture Based [201]	0.04	No

2.5.8 Mimicking Attack

To evaluate the mimicking attack, we look at the False Acceptance Rate (FAR), because a higher FAR is indicative of a more successful attack. FAR is calculated at the same decision threshold where the EER occurs. We observe an overall increase in FAR in M01 for both pencil and finger writing scenarios from < 0.1 to over 0.2 during the attack. For M02 we observe a slight increase in FAR in pencil writing from 0.09 to 0.11, followed by finger writing from 0.8 to 0.15. For M03 and M04, in pencil writing scenario, we observe an increase in FAR from 0.33 to 0.40

and from 0.26 to 0.49, respectively. But for finger writing scenario, both frameworks showed a decrease in FAR from 0.43 to 0.35 and from 0.66 to 0.37, respectively. In summary, M01 is more vulnerable to a mimicking attack in both finger and pencil writing scenarios followed by M02, which shows a relatively lower increase in FAR. While M03 and M04 have higher FARs even in non-mimicking attack scenarios, we observe that only the pencil writing scenario is affected by the attack.

2.6 Factors Impacting Performance

In this section, we comprehensively investigate additional factors that could potentially impact the performance of the authentication frameworks evaluated in this work. We focus on framework-specific factors, namely, feature selection and its impact on the learning-based classification models employed by all the four frameworks followed by participant-specific factors, namely, how the diversity in handwriting styles and techniques impact performance.

2.6.1 Feature Analysis

As all the authentication frameworks that we study in this work employ some type of a supervised learning-based classification function, our first objective is to further investigate which features (computed from the training data) have the most impact on the framework performance, and if the performance varies significantly with a change in the feature set. We first evaluate the M01 framework, which employs both temporal and frequency domain features and ideally uses the top-30 features out of a total of 182 features calculated for each of the accelerometer and gyroscope raw data stream. In other words, it uses a total of 60 features (30 computed from the accelerometer data and 30 from the gyroscope data). We re-tested the M01 framework by reducing (choosing top-15 instead of top-30 features for each sensor stream) and increasing (choosing top-60 instead of top-30 features for each sensor stream) the size of the employed feature sets and observing its effect on the overall performance of the scheme under different writing scenarios.

Our experimental results for this analysis, outlined in fig. 2.6, show that for the pencil and

air writing scenarios, the mean EERs obtained for the reduced feature set case (top-15 features for each sensor stream) is quite comparable to the regular case (top-30 features for each sensor stream), indicating that the framework performs fairly well even when using a lower number of features. Finger writing scenario was an exception here, where the performance for the reduced feature set case was slightly worse (mean EER around 0.11) compared to the regular case (mean EER just under 0.08). However, we observed that, as the size of the feature set increases, the performance of M01 framework worsens for all three writing scenarios. This indicates that the number of features originally selected by the M01 framework for the final feature set (i.e., top-30 from each sensor stream) provides the optimal performance.

Among these top-30 features per sensor stream, we observed that close to 50% of the features were related to general statistics, such as mean, standard deviation and variance of both time and frequency domain features. However, for all the three writing scenarios for M01, we observed (across all participants) that only a very few frequency domain features were selected (in the top-30) during the feature selection step compared to the time domain features. Also, having a larger intermediate feature set (182 features) allows the model training pipeline to select the best features for a given participant. We believe that this is also one of the main reasons why M01 performed well across all writing scenarios and all participants, compared to other schemes.

In contrast to M01, the M02 framework does not include a feature selection step in the processing pipeline, and computes and uses only time domain features for model training and verification. Thus, for M02, we further investigate for each writing scenario the best set of features and study their impact on its performance. To this end, we select, top 8, 16 and 24 features out of the total 32 feature set. Our analysis shows that a top-8 feature selection step provides minor performance improvements (lower EER) across all writing scenarios. We further observed that for the pencil writing scenario, 6 out of the top-8 features are accelerometer features. This does reflect on the actual writing scenario since during pencil writing there would only be very minute angular accelerations. In finger writing, we see an equal number of features from both accelerometer and gyroscope sensors since more wrist movements could be observed during finger writing. Air writ-

ing shows similar behavior to finger writing with roughly an equal number of features from both the sensors in the top-8. Even though the M02 framework was designed for the pencil/pen on paper handwriting scenario, we observed in our evaluations that it performed rather well for the air writing scenario, compared to the pencil and finger writing scenarios. We believe that this may be because the features computed in M02 are much more responsive to significant movements of the wrist (including the arm), which is the case during air writing. To perform better for the pencil and finger writing scenarios, M02 may need to include a much more carefully computed set of features that precisely captures the subtle movements of a user's wrist.

The DNN based classification model for the M03 framework employed the raw accelerometer data stream, and thus no feature extractions were required/performed. As a result, we were not able to perform an equitable feature level analysis for M03, similar to what we did for M01 and M02. However, we believe that a complex DNN-based classification function, such as the one used in M03, would require large amounts of training data to achieve acceptable levels of performance, comparable to other frameworks analyzed in this work. This requirement of a collection of a large training dataset is not feasible during a relatively short enrollment period in practice, and is the biggest shortcoming of M03.

To recollect, framework M04 was originally proposed for signature verification. Our goal was to investigate if such a scheme could be adapted for a free-form handwriting based authentication. Our experimental results and analysis, as presented in section 2.5, show that M04 features are not well-suited for authentication using motion data corresponding to free-form writing with it producing EERs around 0.4 across all the writing scenarios. We believe that one of the main reasons for this is because the written text corresponding to a signature has very little variability, compared to free-form writing which has a lot of variability (for example, the same letter can be written in different ways by the same person). Thus, features in M04 which are computed based on DTW (or similarity) scores between the query or test sample and a set of template (or training) samples works well for signature based authentication, but do not generalize well for authentication based on free-form writing.

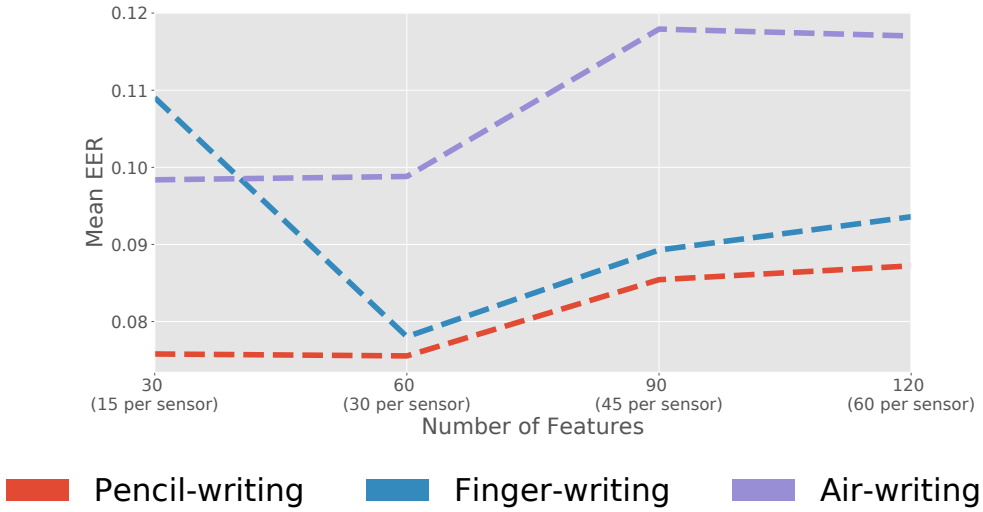


Figure 2.6: M01 feature analysis.

2.6.2 Participant Handwriting Specific Factors

As users have different (often, unique) handwriting styles and traits, our second objective is to further investigate which participant-dependent handwriting factors significantly impact the performance of the frameworks under consideration. To this end, we first carefully analyze the inconsistencies or irregularities that could be observed in users' handwriting traits and styles. We observe (during our data collection experiments) that there are significant irregularities in handwriting by the same participant. One key factor that contributed to this irregularity is the number of strokes a user or participant employs when writing certain characters. For example, most users write the English uppercase letter 'B' with 2 strokes, but at times the same user may write it using just one stroke (for example, when in haste or hurry) resulting in a completely different wrist motion. To further characterize this irregularity in users' writing styles/behaviors, we analyze the finger writing data which we collected using a smart tablet as the writing surface. From this data, we observe that several users employ varying number of strokes for at least 3 out of the 26 lowercase alphabets and for at least 5 out of the 26 uppercase alphabets. These character-level inconsistencies or irregularities easily propagate to words and sentences, and thus to any models trained on these prolonged writing constructs. In summary, handwriting irregularities may adversely impact the performance of authentication models trained using handwriting-related motion data, and such models may not

generalize well to the different writing styles by the same user in different contexts.

Another factor that could adversely impact the performance of authentication frameworks employing handwriting related wrist motions is the positioning of the wrist while writing. During the authentication enrollment phase, a user may have positioned his/her wrist at a certain stance with respect to the writing surface, however, during the authentication (or verification) phase, that stance may be different or may have changed. The stance or positioning of the wrist while writing, together with the angle of the wrist, could significantly affect the way a user's wrist moves while writing, which inadvertently affects the performance of these authentication schemes. Additionally, the tendency of the wrist wearable device striking or coming into contact with the writing surface while writing (especially, during pencil and finger writing scenarios) could also introduce significant noise in the captured wrist motion data and affect the authentication performance. These factors and issues were very commonly observed during our handwriting data collection experiments which were done in a realistic and completely unconstrained setting.

Achieving high performance (or accuracy) when building a classifier for hand-writing motion based user authentication will require taking into account all these inconsistencies during the model training phase. But the main question that arises to this end is: *how easy or practical it is to replicate all such instances of inconsistencies (which are highly dependent on external and, at times, unpredictable factors) during the enrollment/training phase.* As an example, in a traditional fingerprint-based (static) biometric authentication system, users would simply be required to touch the fingerprint reader with different pre-defined angles/portions of the finger during the enrollment phase. But, to train a handwriting-based (dynamic) authentication system in a similar fashion, it will be non-trivial to enumerate all the pre-defined set of scenarios that users must write in during the enrollment phase. Furthermore, from a user convenience point of view, users may be reluctant to spend too much time in the enrollment phase. However, in a use case where handwriting-based authentication is used as a continuous authentication method, the classification models can be continually updated/improved over time with more user data passively collected during authentication events. In this way, the difficulty in training the system due to various participant-specific and

Table 2.4: Summary of results.

RQs	Insights Gained
RQ1	Frameworks such as M01 and M02, performed reasonably in terms of usability/practicality under constrained writing, but required significant proportions of training data which may hinder their adoption as a mainstream authentication scheme.
RQ2	Certain frameworks (especially M01) demonstrated good potential for adaptation across different writing modalities.
RQ3	A degradation of performance was observed in the presence of ambient noise, which suggests that in real-life usage performance could further degrade raising practicality issues for mainstream adoption.
RQ4	M01 and M02 evaluations suggest that free-form handwriting-based user authentication is achievable with trade-offs between convenience and security showing potential for mainstream adoption.

environmental factors discussed above can be overcome to a certain degree.

2.7 Discussion & Conclusion

The widespread use of smart devices in activities of everyday life, along with various apps handling users' private information, has made authentication of these devices/applications essential. Thus, the need for secure, yet convenient mechanisms for user authentication have become imperative. In this work, we evaluated four state-of-the-art handwriting-based authentication schemes with the goal of understanding the true potential and practicality of these schemes, followed by an extensive analysis of possible technical challenges faced by such authentication schemes. We comparatively analyzed these schemes against vital parameters, such as the writing sample window sizes, training data sizes, and performance at different sampling rates. Findings related to our specific research goals are summarized in 2.4. We further discussed how each of these schemes perform in terms of convenience and security, which is often a trade-off when it comes to authentication mechanisms.

When considering different writing settings, while air writing has shown comparatively better performance with an EER below 0.05 specifically with scheme M02, the practicality of using air writing for a continuous authentication scheme is questionable. It is unlikely that air writing being used as a writing mode for extensive writing tasks, since prolonged air writing could be tiring for the arm, making it unsuitable as a continuous authentication scheme. Both pencil writing and finger writing based authentication seem reasonably practical in real life, but finger writing has a slight advantage since it does not require any other tools such a pencil and paper.

Prior to concluding, we would like to highlight some additional shortcomings of the handwrit-

ing motion-based authentication frameworks studied in this work. All the authentication frameworks evaluated in this work employ a binary classification function that needs to be trained using both authentic and non-authentic user data. Such a trained model cannot be developed purely on the user-end (e.g., a user's device), but it has to be developed on some service provider end who has access to data from multiple users. In other words, when new users want to enroll in such an authentication framework, they may have to share their personal motion data (corresponding to their handwriting) with a service provider for building a personalized authentication model for themselves. This raises significant privacy concerns for the users. Moreover, training the model on the service provider end is not efficient as any (or all) model updates (e.g., as required in the case of continuous authentication) would need to be communicated to the provider resulting in significant communication cost and latency.

Our concluding perspective on handwriting-based authentication is that while its immediate adoption is uncertain, it can show major improvements in the future as smart wearables come equipped with more precise and efficient sensors. When that occurs, handwriting-based authentication can potentially become another mainstream mechanism for user authentication.

Acknowledgment

Research reported in this publication was supported by the Division of Computer and Network Systems (CNS) of the National Science Foundation (NSF) under award numbers 1828071 and 1943351.

CHAPTER 3: PRACTICAL APPLICATIONS AND COMMUNICATION PROTOCOLS FOR WEARABLES: EFFICIENT VIBRATION-BASED COMMUNICATIONS OVER HUMAN BODY USING MOTION SENSORS

**The contents of this chapter and the reported experimental results have previously been published in the journal, Internet of Things, Volume 23, 2023, co-authored by Raveen Wijewickrama, Sameer Anis Dohadwalla, Anindya Maiti, Murtuza Jadliwala and Sashank Narain (in that order). The contents of the published manuscript has been reproduced here with revisions.*

3.1 Introduction

The ubiquity of smart wearables has rapidly increased in the past few years, wherein inexpensive devices such as smartwatches, fitness bands, smart-glasses, and smart-shoes provide users with a variety of additional capabilities beyond that of a smartphone. Apart from a small subset of these smart wearables which can operate autonomously, the vast majority of smart wearables are used in conjunction with a paired user-held mobile device such as a smartphone. Currently, this communication between the mobile device (smartphone) and paired smart wearables is primarily accomplished using short-range wireless radio technology such as Bluetooth/BLE, NFC, and Wi-Fi. Although these technologies are very mature and enable robust communication between mobile devices, wireless radio signals are prone to eavesdropping, jamming and/or interference, resulting in loss of communication or privacy or both. There is a genuine need for reliable and stealthy non-radio communication side-channels in several use-cases and applications involving these devices. For instance, a low bandwidth, non-radio based protocol could be used to swiftly and securely send a short secret code or a PIN from a smartphone to a smart wrist wearable to automatically initiate and establish a high-bandwidth and secure radio channel, such as Bluetooth or WiFi. Besides secret sharing, the availability of such an out-of-band communication side-channel could prove to be convenient and practical for several other security applications such as proof of device co-location and user authentication. The increasing number of body worn smart devices

which includes both recreational (e.g., smartwatches, smart-headsets etc.) and medical (e.g., continuous glucose meters, wearable blood pressure monitors etc.) use cases, the need for such device co-location proofs becomes further heightened. Especially, since all these devices share a common platform, i.e., the skin it makes a compelling argument to use the skin itself to transmit signals among these devices. Some recent proposals on alternate body-area communication channels have suggested the use of optical [42, 43] and acoustic signals [62, 97] to enable low-bandwidth message transfers between mobile devices. Such communication channels, however, are not robust and often perform poorly in the presence of noise or non-ideal ambient conditions, and are not very covert (i.e., easily detectable) [158].

Other proposals in this direction have employed vibration motors, often embedded in mobiles devices to provide haptic feedback to users, as a transmitter [84, 114, 156, 157]. In such techniques, fine-grained motion sensors such as accelerometers and gyroscopes (also found on most modern mobile devices) have been adopted as receivers [126]. Hwan et al. [84] were one of the first to propose the pairing of a vibration motor and an accelerometer to establish a protocol for low-bandwidth communication between two smartphones kept on the same (hard) surface. Roy et al. [157] followed up by proposing a scheme, called Ripple, which provided improved bandwidth and added security. Kim et al. [94] proposed the use of a smartphone as a vibration transmitter for a custom medical device with an accelerometer implanted under the skin acting as the receiver. Later, Sen and Kotz [167] proposed the use of a smart ring as a vibration transmitter to communicate with Internet-of-Things (IoT) devices equipped with accelerometers, however for their scheme to work, the transmitter (smart ring) needed to be in direct contact with the receiver (IoT device) for reliable communication.

In this work, we take inspiration from these earlier research efforts to explore the feasibility of employing vibration motors and accelerometers to build a reliable communication protocol between two user-held mobile devices (e.g., a smartphone and a wrist wearable), by using the human body/skin as the underlying physical medium for vibration signal propagation. Given that modern mobile devices such as smartphones and wearables are always in contact with the human body/skin,

and that they come pre-equipped with vibration motors and motion sensors, such a communication technique could be naturally deployed without requiring any special setup/hardware. Also, as the user's own body/skin acts as a communication channel, the scheme would be naturally covert and robust against trivial eavesdropping. As mentioned earlier, we believe that such a vibration-based communication scheme could be easily extended to a body-area network where any two devices, in or on a person's body, could employ it as a covert side-channel for other use-cases such as proof of co-location (on the person's body). In addition to being employed for on-body communications, the same principle could also be used for effecting communication between an on-body device and an external device, for example, by briefly touching the external device on the body to transmit data (as vibrations via the skin) to the on-body device(s).

Although vibration-based communication techniques have been proposed before, additional research and experimentation is required to build a robust (against noise and movements) and easy-to-deploy communication system that employs vibrations traveling through human skin/body, which is what this work attempts to accomplish. We attempt to utilize existing vibration based modulation techniques and adapt them to the transmission via human skin/body which is where one of our main contributions occur. To this end, the main challenges include identifying how vibrations can be modulated so that they can effectively propagate via human skin/body, and efficiently using an accelerometer to sense these vibrations in the presence of volatile body movements. Moreover, inherent anatomical and biomechanical differences between individuals can also affect the efficiency of the communication channel. Consequently, the main requirements as we design such a communication technique are: (a) **robustness** to vibration noise and anatomical differences that are inherently present in human body/skin, (b) a sufficient level of **data rate** suitable for low-bandwidth communications, and (c) reliable communication under **mobility** such as while traveling in a vehicle.

Our key contribution in this research work is the design, implementation and evaluation of a novel vibration-based communication protocol, called *SkinSense*, which utilizes human body/skin as a communication medium in order to create a low-bandwidth and covert communication channel

between user-held mobile and wearable devices. We demonstrate the feasibility and performance of *SkinSense* by employing both a custom hardware setup, which enables the flexibility of being able to manipulate the different hardware and protocol parameters for a comprehensive evaluation, and consumer-level devices. By collecting data from a diverse set of human subject participants, we comprehensively evaluate *SkinSense* under different/varying operational parameters and communication settings, including, distance between transmitter and receiver, device orientation, ambient noise, sampling rate, communication direction and motion sensing hardware (accelerometer versus gyroscope). In addition to this, we also conduct an in-depth analysis of the impact of the various *SkinSense* design parameters on communication performance and error rates, empirically evaluate its vulnerability to acoustic or Sound of Vibration (SoV) attacks, and study its energy requirements. Lastly, we systematically compare the performance of *SkinSense* with other efforts in the literature and conduct a small user-study to gauge the perception and preferences of end users using such a communication side-channel.

3.2 Background

We now briefly introduce some background information relevant to our proposed communication protocol.

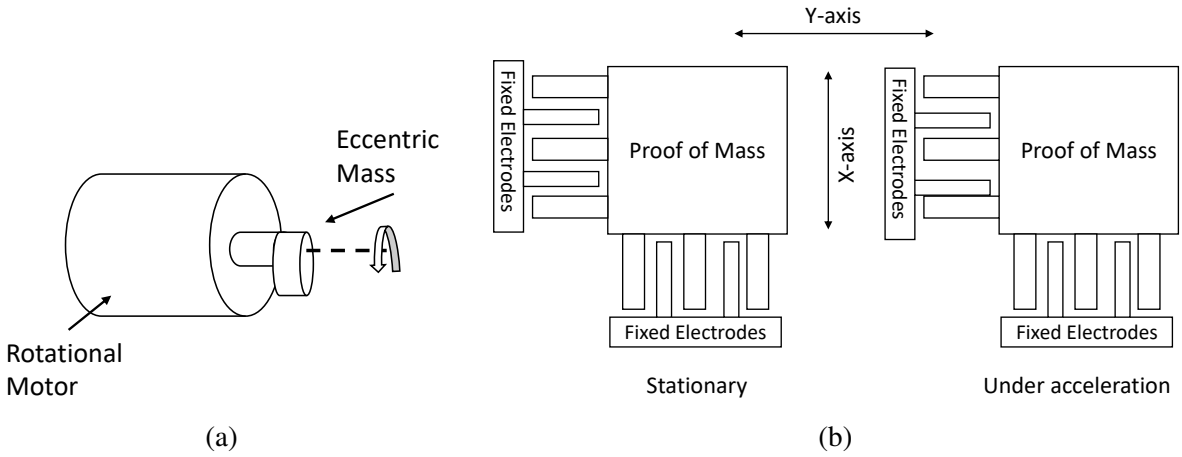


Figure 3.1: Architecture of (a) ERM motor and (b) capacitive accelerometer.

3.2.1 Vibration Motor

A vibration motor is a mechanical device that uses motion from an unbalanced metallic mass to produce oscillations or vibrations [64]. They are usually found inside most modern mobile devices such as smartphones and smartwatches to provide haptic feedback or notifications to users. There are two main types of vibration motors: (i) *Eccentric Rotating Mass (ERM)* motors, and (ii) *Linear Resonant Actuator (LRA)* motors. ERM motors are powered by direct current (DC) and employ an asymmetric mass (fig. 3.1a) which moves eccentrically when rotated [166]. The amplitude and frequency of the vibrations can be controlled by varying the input DC voltage, and they both increase with higher voltages [5, 157]. LRA motors, on the other hand, consist of a magnetic coil that pushes a mass up and down to create vibrations, enhanced by a spring. LRA is driven by a precise amount of AC current so as to achieve resonant frequency of the spring, which limits the vibration motor's amplitude and frequency within a very narrow band. Huang et al. [83] showed that ERM motors are better suited for applications requiring complex encoding of the vibration signal, compared to LRA motors which work well only for simple binary encoding. Therefore, we employ an ERM-based DC coreless motor in this work.

Motor Control. *Pulse Width modulation (or PWM)*, a technique to reduce the average power delivered to a load by effectively breaking the input electrical signal into discrete parts, is often used to control intertial loads such as vibration motors found in most mobile devices [25, 113]. The average value of voltage (and current) supplied to a motor in PWM is regulated by turning the switch between the power supply and the motor ON and OFF at a fast and variable rate. In other words, PWM can control the operation of a motor by providing it with a series of electrical pulses or "ON-OFF" signals. The power to the motor is controlled by varying the width of these pulses, which in turn, varies the average DC voltage applied to the motor. Ultimately, PWM enables greater control over the motor without altering the voltage level of the supply voltage, which is ideal for encoding signals over motor vibrations. We utilize a selected range of PWMs to control the motor based on the feedback observed via vibration signals captured using motion sensors, as discussed later in section 3.3.3.

3.2.2 Accelerometer

Accelerometers are used to measure acceleration or speed changes of a device with respect to the surrounding environment. Most modern mobile and wearable devices are equipped with *micro-electromagnetic (or MEMS)* [56, 187] accelerometers, which enable applications ranging from simple step counting to complex activity recognition [174]. There are two main types of MEMS accelerometers in use today: (i) *piezoresistive* accelerometers, and *capacitive* accelerometers [63]. Piezoresistive accelerometers consist of a mass attached to a piezoelectric crystal. When vibrations or movements occur on the accelerometer, while the mass itself remains unchanged, it makes the crystal either compress or stretch depending on the direction and magnitude of acceleration. Capacitive accelerometers (fig. 3.1b) consist of a proof-of-mass suspended between two plates, one which is fixed and other which is free to move inside the accelerometer housing. When vibrations/movements occur on the accelerometer, the distance between the plates change proportionally to the acceleration, resulting in a change of capacitance [188]. Most modern devices are equipped with capacitive accelerometers due to their smaller size and ability to measure low-frequency motion [56]. Accelerometers produce raw measurements along three axes, with gravity components applied to whichever axis is pointing to the ground. Another type of motion sensor commonly found in consumer mobile devices is the gyroscope, which in contrast to accelerometers, measures a device's angular velocity. Later in section 3.4, we comparatively evaluate our proposed communication protocol using both the accelerometer and gyroscope sensors.

3.2.3 Mechanics of Vibrations

Vibrations are a form of mechanical oscillations, defined as repetitive back and forth movement of an object between two states or positions about an equilibrium point [150]. There are 2 kinds of vibrations, *free* and *forced*. Free vibrations occur as a result of a brief energy transfer from an external force on to an object (e.g., one time picking of a guitar string) [150]. In contrast, forced vibration is a continuous periodic external force applied to an object (e.g., repeatedly pushing a playground swing). Vibrations can be fully described using two parameters, namely, *frequency*

and *amplitude*. Frequency of vibration describes how fast the vibrating object is moving, while amplitude is the maximum displacement of the vibrating object in motion, i.e., the strength of the vibrations. When an object vibrates, its particles are disturbed by the vibrational energy propagating from one point of the object to another, also called the *vibration wave* (fig. 3.2). How well these waves propagate through the object (or medium) depends on its molecular structure (such as density). If an object has tightly packed molecules, it is known to have a higher rigidity (e.g., wood or steel). Such objects or mediums are better at transmitting vibration waves making them travel farther, faster and longer compared to less rigid mediums (e.g., a cushion pillow) [31]. Vibration waves have a characteristic wavelength (γ), which is the distance between two adjacent crests (highest points) of the wave. When a wave travels from one medium to another, although the wavelength changes, the frequency remains unchanged [150]. This is defined as $v = f\gamma$, where v is the velocity and f is the frequency of the wave. Thus, the velocity and wavelength of the vibration waves originating from some source and transferring to an object/medium could change depending on the rigidity of the object.

In this work, we are attempting to leverage vibrations originating from a handheld device such as a smartphone to design a communication mechanism, where the vibration waves propagate to the receiver through the skin of the hand (holding the handheld). Previous works on vibration-based communication schemes have employed objects such as wooden tables as the transmission medium [157]. These efforts have been able to achieve very high data rates and accuracy, primarily because wood is a good propagation medium for vibration waves due to its high rigidity. In contrast, human skin absorbs more vibrations due to its lower rigidity, making it less desirable as a transmission medium for vibration waves [5]. Consequently, designing a vibration-based communication scheme relying on the propagation of the vibration waves through the human skin is a much more challenging endeavor. *Prior studies have shown that the propagation of vibration waves via the human skin is correlated to the vibration frequency of the source, and that the vibration decay is quicker at low and high frequencies (i.e., vibrations cannot propagate for longer distances) compared to intermediate frequencies of the source [120, 169].* Since our work consid-

ers a communication scenario of a handheld device and a wrist wearable, which naturally results in a certain amount of distance (with distances ranging from 15 *cm* to 2 *cm* depending on the motor placement on the handheld device) between the two devices when worn on the hand, understanding and analyzing this vibration decay and the effect of vibration frequency was important in designing an effective communication framework. In order to identify these intermediate frequencies which would help in designing *SkinSense*, we do a *sine sweep test* as explained further in section 3.3.2.

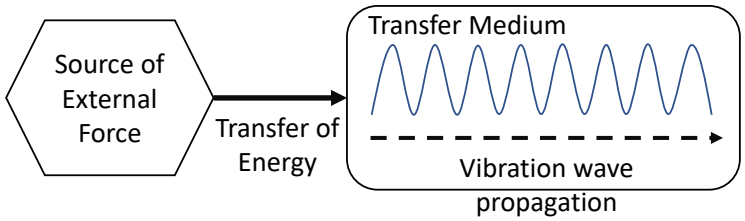


Figure 3.2: Vibrational energy propagation from a source to a medium.

Based on the high-level discussion so far on the mechanics of vibrations, we now formally describe how vibrations created by motors (e.g., ERM motors) can be used to send vibration waves (or signals) through human skin as a medium. The centrifugal force generated by an ERM motor is given by:

$$F = mr\omega^2$$

where F is the force (in $Newtons$), m is the mass of the eccentric mass (kg), r is the radius of the eccentric mass ($meters$) and ω is the angular velocity (speed of the motor) in $rads/sec$ [150]. As m and r are physical properties of the motor that cannot be changed, the centrifugal force generated by the motor can only be changed by manipulating the angular velocity, ω. The vibration frequency and amplitude in an ERM motor cannot be changed independently and they both increase linearly based on the voltage provided. ERM motor speeds are proportional to the applied voltage, therefore amplitude/frequency changes can be done by manipulating the voltage via pulse width modulation (PWM) as discussed above in section 3.2.1. As clarified earlier, identifying a set of intermediate frequencies that propagates well from a handheld device to a wrist wearable via human skin as the medium is the first step in designing an effective communication scheme via skin. To modulate a

set of frequencies, we rely on a PWM based technique and then use these different frequencies to encode data in *SkinSense* as described in detail in section 3.3.

3.3 System

We now present the design and other technical details of *SkinSense* by first providing an overview of the system architecture, followed by design details of the communication protocol. We then present the technical details of the encoding (modulation) and decoding (demodulation) algorithms followed by an outline of different hardware setups that we employ in the implementation of *SkinSense*. Finally, we present details of the human subject data and performance metrics used in its evaluation.

3.3.1 System Overview

Figure 3.3 shows a high-level overview of *SkinSense* enabling vibration-based communication between a handheld device and a wrist wearable by using the user's hand (specifically, the skin tissues) as the communication channel. We consider a half-duplex communication channel, i.e., the handheld device and wrist wearable can both act as a transmitter and receiver, however information can flow in only one direction at a time. Moreover, we consider that the communicating devices are located on the same hand, but they do not need to be in physical contact with each other. The message that needs to be transmitted from the handheld device to the wrist wearable (or vice versa) is first encoded into a stream of vibration signals by using an encoding (or modulation) algorithm described in Section 3.3.3. The vibration motor on the transmitting device then emits these vibration signals, which is carried via the skin tissues of the hand to the receiving device. The receiving device's motion sensor records these vibrations and decodes the encoded message using a decoding algorithm outlined in Section 3.3.3.

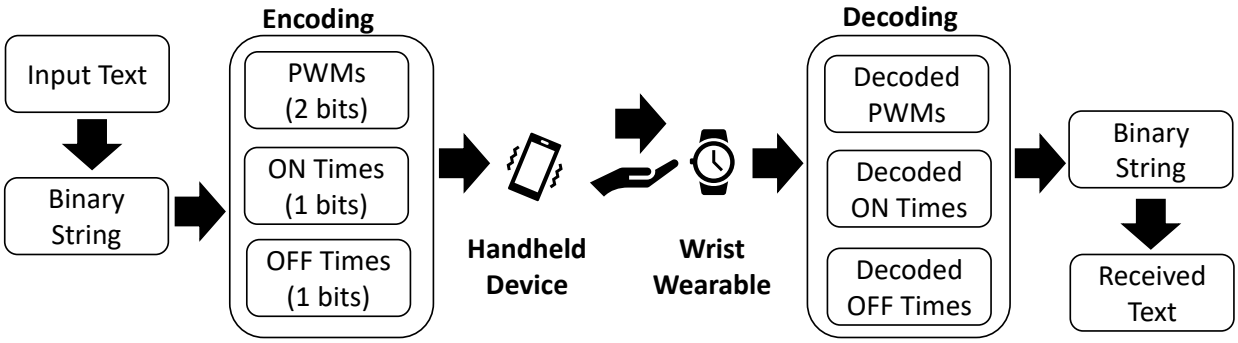


Figure 3.3: *SkinSense* communication protocol.

3.3.2 Transmitter Design

Our final transmission algorithm was arrived at after many rounds of iterations and preliminary investigations where we scrutinized on how vibrations travel through the human skin and its ability to be captured by an accelerometer. As a first step, our goal was to examine the possibility of identifying/differentiating multiple vibration frequencies for data encoding, which would effectively allow us to significantly increase the bandwidth of our communication protocol (compared to using only a single frequency encoding scheme). Specifically, we analyzed how vibrations of different frequencies, after being generated by the motor and traveling via the skin, gets captured on an accelerometer. Once specific operable frequencies with clear separations were identified, our next goal was to minimize the vibration times, i.e. the time of a single vibration pulse (`ON` times) and the interval between two vibration pulses (`OFF` times). At the same time, to further increase the bandwidth and data rate, we also found that time domain data encoding could also be used in conjunction to frequency-based encoding by reliably determining vibration `ON` times and `OFF` times. The specific details of selecting operating frequencies and time modulation are presented next.

Identifying the operable frequencies

In order to identify and characterize the range of frequency bands to operate the transmitter motor in, such that the receiver accelerometer would have a distinct response to the vibration

signal, we did a *sine sweep test* [157] by test operating the motor in the range of 20 (low) to 240 (high) PWM values and analyzed the corresponding accelerometer signal at the receiver.

We observed that the frequency response captured by the accelerometer peaks around PWM 60 and fades away starting around PWM 100 (refer fig. 3.4). Based on this observation, we determined that the motor reaches its resonant frequency at around PWM 60. While a simple ON or OFF binary encoding mechanism would only require one operational frequency for the motor, this will limit the amount of data that can be encoded using vibrations. After identifying the operable frequencies to be in the range of PWM 20-100, we then closely analyzed which specific PWM frequencies to work with. For the frequency analysis, we closely observed the frequency response of PWMs from 20 up to 100 in steps of 10. We subsequently found PWMs 20, 30, 60 and 100 to be the ones with minimal interference with neighboring PWMs, and thus employ them as the final set of PWM parameters in *SkinSense* implementation.

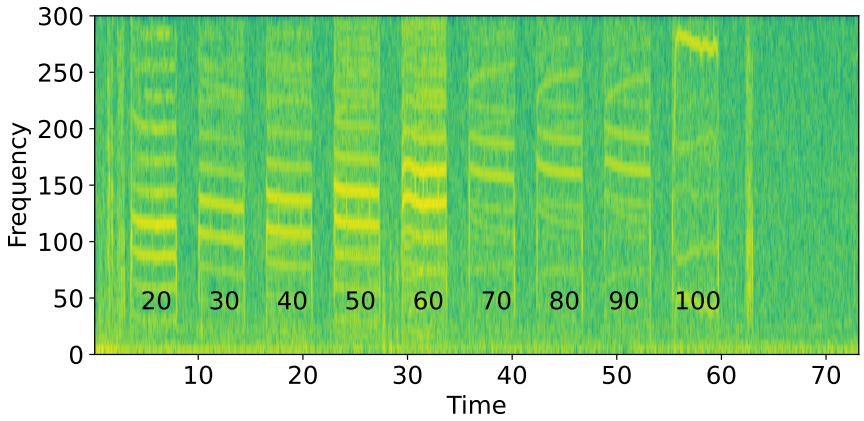


Figure 3.4: Spectrogram of PWMs ranging from 20 to 100.

Time based modulation

We next studied the possibility of minimizing the ON time and OFF time of the vibrations to further increase the data rate. For that, we analyzed how accurately we can infer the ON time and OFF time windows of vibrations using the accelerometer signal for a range of ON and OFF time values. These ON and OFF times are affected by a phenomenon called the *ringing effect* [157], where the vibration may remain in the medium for sometime before completely dying down when

the voltage is cut off to the motor. When choosing OFF times, we need to specifically select values that accommodate the remnants of the previous vibration pulse such that, the next vibration pulse is not affected by it to prevent inaccuracies during demodulation. We tested OFF times in the range of 150-1000 *ms* and were able to clearly distinguish between OFF time values of 250 *ms*, 500 *ms*, 750 *ms* and 1000 *ms*. Any values lower than 150 ms proved to be too short, leading to interference between consecutive vibration pulses. Therefore, without affecting the overall accuracy, we select 150 *ms* and 300 *ms* as the OFF time window values in our algorithm implementation. After finalizing appropriate OFF time values, we determine ON time window values, i.e. the amount of time the vibration motor stays on in a single window. Similar to the OFF time analysis, we tested ON times in the range of 250-1000 *ms*, followed by ON time values below 250 *ms*, however values below 250 *ms* proved to be too small to be accurately identified in the accelerometer signal. Thus, without affecting the overall accuracy, we fixed 250 *ms* and 500 *ms* as the ON time window values for our implementation. The full hardware setup used for this analysis is detailed in section 3.3.4. An independent analysis may be required to determine the optimal parameters for each motor-accelerometer pair since different motor types may have varying resonant frequencies, and different accelerometer types may have varying sensitivities.

3.3.3 Communication Algorithms

SkinSense employs a PWM based frequency modulation technique for encoding bits as vibration pulses and a spectrogram based approach to decode the sensed motion data (corresponding to the vibration pulses) to reconstruct the transmitted bits, as outlined next.

Encoding Algorithm

The transmission algorithm takes as input a sequence of bits to be transmitted and outputs a sequence of parameters (PWMs, ON times and OFF times) to be passed on to the vibration motor. The PWMs are a measure of the voltage given to the motor, which in turn controls its frequency. The ON times signify the time in milliseconds (*ms*) for which the vibration motor is switched on,

while the OFF times signify the time in milliseconds between two consecutive vibration pulses. By modulating these three parameters independently, we are able to transmit the bit string as a series of vibrations of differing times and frequencies.

SkinSense currently uses four values of PWMs (20, 30, 60, 100), two values of ON times (250 *ms*, 500 *ms*) and two values of OFF times (150 *ms*, 300 *ms*) for modulation, as described earlier in section 3.3.2. As a result, we are able to encode $log_2 4 = 2$ bits using the PWM values and $log_2 2 = 1$ bit using the ON time and OFF time values, respectively. In order to transmit a bit sequence using the above setup and set of parameters, we need to partition it into 4-bit long words. Then, we encode each 4-bit word as follows: The first two bits of the word are encoded using one of the four PWM values, the third bit using one of the two ON time values and the fourth bit using one of the two OFF time values. At the end of transmitting message, we append a pilot sequence to improve accuracy while decoding.

Decoding Algorithm

The decoding algorithm takes as input the raw accelerometer values sensed at the receiver and outputs the decoded bit sequence. The sequence of steps involved in the decoding algorithm are as follows:

(1) Spectrogram computation: Raw signal filtering and normalization.

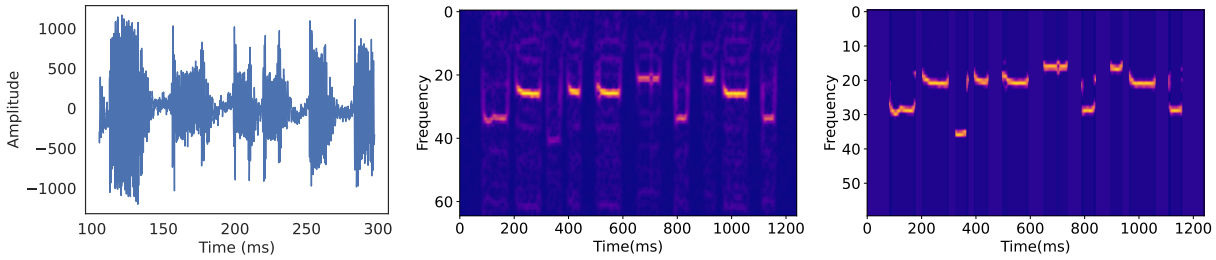
(2) Peak detection: Deriving the encoded parameters.

(3) Symbol separation: Symbol separation to identify the encoded PWM values, ON time values and OFF time values.

(4) Pilot sequence based mapping and decoding: Decode the message by generating the corresponding bits associated with each PWM values, ON time values and OFF time values.

Spectrogram. The spectrogram of raw accelerometer time-series data samples are first computed in the decoding process. Essentially, a spectrogram is a matrix which depicts the strength of the accelerometer signal over time at different frequencies (a spectrum of frequencies), i.e., each

position of the matrix corresponds to a point in frequency and time. In other words, the time-series data is converted from the time domain to the frequency domain (using FFT with a window size of 128 samples and an overlap of 124 samples). In order to reduce the effect of noise, we filter out all frequencies below 40 *Hz* from the spectrogram. These frequencies fall below our band of operation and correspond mainly to low frequency noise which may be present in the human hand. Figure 3.5 shows the frequency spectrum before and after removing this noise. We then perform mean and variance normalization on the resulting spectrogram in order to eliminate any constant biases that may be present.



(a) (b) (c)

Figure 3.5: (a) Raw accelerometer signal, spectrogram of the accelerometer signal (b) before filtering, (c) after filtering to remove noise.

Peak Detection. In order to determine the `PWM` values used during encoding, we rely on the fact that the frequency of the vibrations are linked to the PWM. By accurately determining these frequencies, we can estimate the `PWM` value. It was observed that for most `PWM` values, there are two frequencies where there is a significant amount of energy present. This can be observed in the Figure 3.4, the two darkest yellow bands for each `PWM` represent these two frequencies. They represent the two most prominent *overtones* of the vibrations passing through the hand (overtones are any frequencies higher than the lowest frequency present in the signal [32]). Due to the presence of two prominent overtones, in order to derive the frequency of the vibration at any point in time, our algorithm detects the two most prominent peaks in the spectrogram and computes the mean of the frequency of these two overtones to obtain a close estimation of the transmitted frequency. However, at higher `PWMs` (e.g., 100), it was noticed that only one overtone contains a majority of the energy. In order to handle these cases, the algorithm compares the frequencies of two most

prominent peaks, and only if the second most prominent peak is of the same order of magnitude as the most prominent one, the mean of the two frequencies is computed. Otherwise, it simply considers the frequency of the most prominent peak for the frequency estimation.

Algorithm 3.1 Symbol separation algorithm.

Input: $y[]$
Output: $ONTimes[], OFFTimes[], PWMS[]$
Initialization: $ONTimes \leftarrow [], OFFTimes \leftarrow [], PWMS \leftarrow [], startTimes \leftarrow [], endTimes \leftarrow [], i \leftarrow 0$
for $i <=$length(y) **do**
  if $(y[i] >= 0$ and $y[i+1], y[i+2], y[i+3] = 0)$ **then**
    $endTimes.append(i)$
  if $(y[i] >= 0$ and $y[i-1], y[i-2], y[i-3] = 0)$ **then**
    $startTimes.append(i)$
$ONTimes[] \leftarrow endTimes[] - startTimes[]$
$j \leftarrow 0$
for $j <=$length($ONTimes$) **do**
  if $(ONTimes[j] < 200)$ **then**
    $ONTimes.remove(j)$
$OFFTimes[] \leftarrow startTimes[1 : (length(startTimes) - 1)] - endTimes[0 : (length(endTimes) - 1)]$
$k \leftarrow 0$
for $j <=$length(ONTimes) **do**
  $PWMs[j] = mean(y[startTimes[j] : endTimes[j]])$

Symbol Separation. From the previous peak detection process, we obtain a vector (denoted as y) of the detected frequency for each time window. In time windows without any vibration, the frequency is set to 0. We then use the following heuristic to separate out the transmitted symbols, in the form of `ON` times, `OFF` times and `PWMs`. First, the first and last time windows of each vibration are computed by using the *symbol separation algorithm* (algorithm 3.1). Then, the `ON` times, `PWMs` and `OFF` times are computed by using this information. In order to negate any effects of short-duration impulsive noise, we discard any symbols whose `ON` times are smaller than 200 *ms*. We do this filtering because the shortest possible length of a transmitted symbol is 250 *ms* as per the `ON` time values we have chosen for transmission, which implies that `ON` times smaller than 200 *ms* are noise.

Pilot Sequence based Mapping and Decoding. As mentioned earlier, a pilot sequence is appended to each message to improve accuracy at the time of detection. Due to a variety of reasons, such as the frequency response of the user's hand, the precise orientation, tightness of the watch, and the time taken by the motor to ramp up and ramp down, there could be considerable variance in the received parameters with every use of the communication system. Hence, we use this pilot signal during decoding to measure any offsets in the transmitted parameters and adjust them accordingly. The pilot signals consists of the following sequence of `ON` times, `OFF` times and `PWMs`, respectively: `ON` time values: [250,500,250], `OFF` times values: [150,300], `PWM` values: [20,30,60].

These encompass most of the parameters used to transmit information in our system. At the receiver, this pilot sequence first undergoes the spectrogram based filtering, peak detection based frequency estimation followed by symbol separation to identify the parameters used in the pilot sequence. These values obtained are then used as a mapping to decode the actual message. We start with the pilot sequence, where each value p_i^{pilot} in the estimated parameter set PWM^e of the pilot sequence e is mapped to its corresponding value q_i from the original parameter set $PWM^o = \{20, 30, 60\}$. By using this information, the output of the symbol separation algorithm is then mapped to their corresponding parameters as follows: For each estimated parameter p_i^{msg} in the estimated parameter set PWM^{msg} for the transmitted message msg, if $p_i^{msg} < mean(p_j^{pilot}, p_{j+1}^{pilot})$, p_i^{msg} is mapped to p_j^{pilot}, else, p_i^{msg} is mapped to p_{j+1}^{pilot}. For example, for `PWM` values of the pilot sequence, the symbol separation algorithm may output the following values, 24, 35, 66. Now since we know that pilot sequence can only have `PWM` values 20, 30 and 60 and the order of their occurrence, we map these estimated values to each corresponding `PWM` value. Then, for the actual message, if an estimated `PWM` value is 27, it is mapped to the pilot sequence estimated value of 24 (via the above described method) which corresponds to `PWM` 20 of the original encoding parameter value. The same procedure is followed for other parameter sets, ON^{msg} and OFF^{msg}. Finally, the message is decoded by reconstructing 4-bit long words for each `PWM`, `ON` and `OFF` combination of values obtained from the mapped symbol separation output, i.e., the first 2 bits of a word are

derived based on the PWM value, while the third and fourth bits are derived using the ON time and OFF time value, respectively.

A sample PIN transmission. Figure 3.6 instantiates the aforementioned algorithms of *SkinSense* with a real example, wherein a 4-digit PIN is first encoded into a vibration pattern (consisting of ON, OFF, and PWM parameters) and then transmitted. A spectrogram composed from the received accelerometer signal is then used to detect the relevant ON, OFF, and PWM parameters based on the vibration pulses, which are then used to successfully decode and output the original PIN.

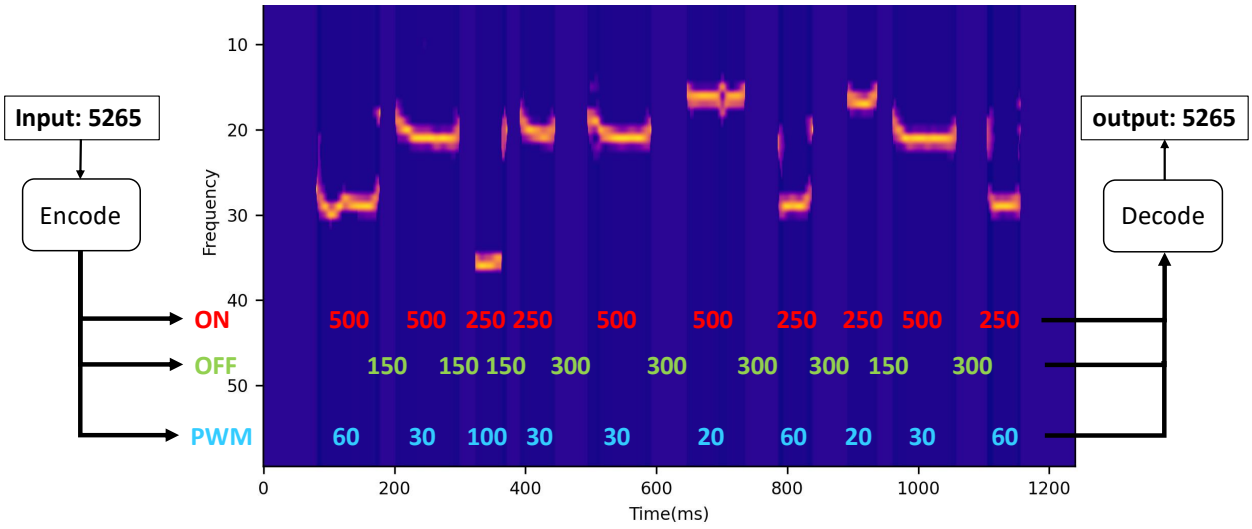


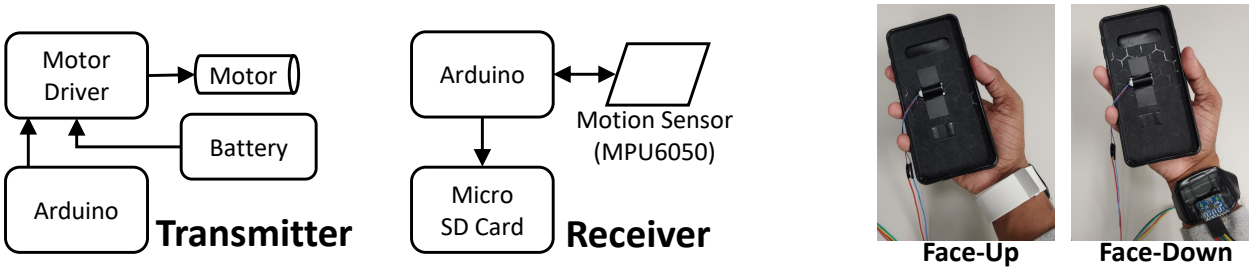
Figure 3.6: Encoding and decoding of a PIN.

3.3.4 Implementation Hardware

To comprehensively evaluate *SkinSense*, we implement two different hardware setups. To enable an exhaustive investigation of the performance for a variety of fine-grained transmitter and receiver parameters, we first implement and evaluate *SkinSense* in a custom hardware setup (Section 3.3.4). Then to evaluate *SkinSense* in a realistic setting, we implement a consumer hardware setup (Section 3.3.4) comprising of commercially available mobile and wrist wearable devices.

Custom Hardware Setup

We implement this setup by building custom devices closely resembling transmitters and receivers on commercial handheld devices (e.g., smartphones) and wrist wearables (e.g., smartwatches). In this setup, we use Arduino Uno and Nano boards to control the transmitter and the receiver, respectively. For the transmitting vibration motor, we use an ERM motor (16000 *RPM*) connected to the Arduino Uno board via a L298 motor driver [48], powered using a 12*V* (5 *Amp-Hours* capacity) Lead Acid battery. We use a 12V battery to power the motor to achieve longer operating times for our experiments. However, as the motor operates in the range of 1.5V to 3V it can also easily be powered using two (1.5V) AAA batteries. We power the Arduino Nano board (the receiver device) via a 5*V* USB input. To build the custom handheld device prototype, we use a consumer level smartphone case (fig. 3.7b) on which we mount the vibration motor. For the custom wrist wearable prototype, we use a Sony Smartwatch 3 Band (the smartwatch portion removed, see fig. 3.7b) to mount the motion sensor. For both the custom devices, we use a MPU6050 GY-521 MEMS motion sensor containing a three-axis accelerometer and a three-axis gyroscope. The motion sensor is also connected with a Micro-SD card using an Arduino Micro-SD card adapter (via SPI mode) for data recording purposes. As explained earlier (section 3.2), we use a `PWM` technique to control the operation of vibration motor. The sampling rate of the motion sensor achieved when writing to the Micro-SD card is 700 *Hz*. To evaluate communication in the reverse direction (i.e. wrist wearable to handheld device), we swap the motor and motion sensor between the two custom devices.



(a) (b)

Figure 3.7: (a) Custom hardware architecture, (b) wrist wearable wearing orientations.

Consumer Hardware Setup

For this setup, we use a Nokia 6 (2017) smartphone as the handheld device and a Sony Smartwatch 3 as the wrist wearable. The Sony Smartwatch can only achieve a maximum sampling rate of 200 *Hz*, in contrast to the sampling rate of 700 *Hz* that we were able to achieve using the custom hardware setup. Moreover, in these devices the Android API only provides control of the motor's vibration amplitude, but not of the frequency. Similar to Roy et al. [157], we looked into the possibility of modulating the amplitude to encode data. However as mentioned earlier, due to sampling rate limitations in consumer level smartwatch accelerometers, we were not able to accurately differentiate between different amplitudes just by using the accelerometer data. This effectively limits the full implementation of the *SkinSense* communication protocol on these devices as we rely on multiple frequencies to encode data.

3.3.5 Human Subject Data Collection

We perform a comprehensive empirical evaluation of *SkinSense* using data collected from human subject participants in realisitic settings and environments. For our data collection study, we recruited 13 participants in the age group of 18-29 from our university campus. Participants in our study wore the custom wrist wearable device on the wrist of one hand (preferred by the participant) and held the custom handheld device (fig. 3.7b) in the palm of the same hand. In this setup, binary data was communicated from the handheld device to the wrist wearable, and vice versa, using our proposed vibration-based approach under a variety of different ambient/device settings and algorithm parameters.

In our first experimental setup, we evaluate the effect of two different orientations of the wrist wearable device on the performance of the communication scheme. The first orientation is the case when the wrist wearable is worn facing upwards along the top of the forearm (or wrist), while the second orientation is when the wrist wearable is facing downwards along the bottom of the wrist (see fig. 3.7b). As different users may prefer to wear their wrist wearables in different orientations and, depending on the orientation of the wrist wearable the position of the vibration motor

and accelerometer on the wrist/forearm may change impacting the performance of *SkinSense*, it is important to further analyze these setups. In the second experimental setup, we test the effect of distance between the transmitter (vibration motor) and receiver (accelerometer) devices on *SkinSense*'s performance. Accordingly, we mount the motor at three different positions on the custom handheld device, top (15 *cm*), middle (7.5 *cm*) and bottom (2 *cm*). In the third data collection setting, participants were asked to walk while holding the custom handheld device and wearing the wrist wearable while the devices were communicating with each other. The purpose of this setting was to study how noise introduced (in the accelerometer sensor readings) due to physical activities such as walking impacts the performance of *SkinSense*. In the fourth experimental setup, we tested the reverse direction, in which the wrist wearable (mounted with a motor) acts as the transmitter and the handheld device (mounted with an accelerometer) act as the receiver. This setup would evaluate the half-duplex property of *SkinSense*. Finally, the different orientation and participant walking setups are repeated for the reverse communication direction as the fifth and sixth experimental setup. In addition to these experiments, we also conducted an experiment to test *SkinSense* using consumer level devices where a set of participants wore a Sony Smartwatch 3 on their wrist while holding a Nokia 6 (2017) smartphone on the palm of the same hand. Similar to the above experiments, binary data was communicated from the handheld device to the wrist wearable. In each of these experiments, four 128-bit randomly generated binary strings are used as test communication (transmission). All data collection was done after obtaining consent from the participants, and our data collection and analysis procedures were approved by our university's Institutional Review Board (IRB).

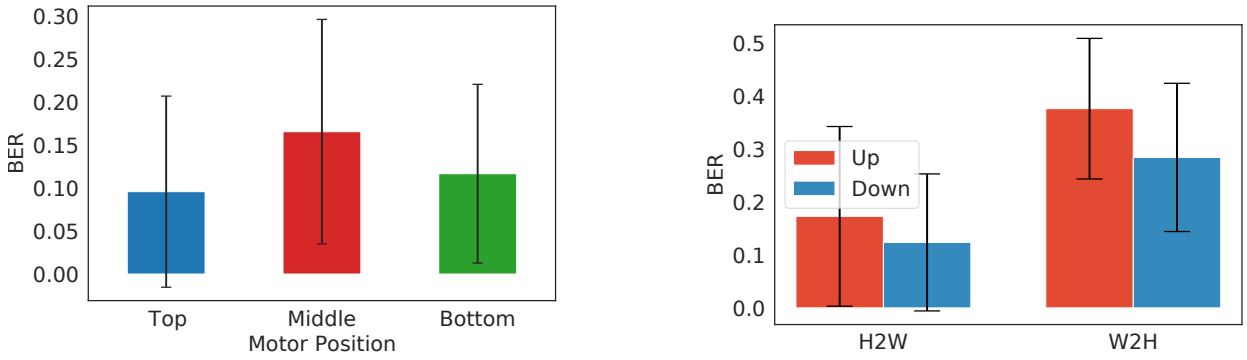
3.4 Evaluation

In this section, we present a comprehensive empirical evaluation of *SkinSense* under a variety of operational settings, algorithm parameters and hardware setups. We evaluate the accuracy and efficiency of *SkinSense* using the following standard metrics: (i) Bit Error Rate (*BER*): *BER* is the ratio of the number of incorrectly interpreted bits, (ii) Bit Rate (*BR*): *BR* is the transmission speed,

i.e. the number of bits transmitted per unit time (seconds).

3.4.1 Effect of Distance Between Transmitter (Motor) and Receiver (Motion Sensor)

In order to observe the effect of distance between the transmitter (vibration motor) and receiver (accelerometer sensor) on system performance, we mount the motor on the phone case at three different positions. Specifically, the motor is mounted at the top, middle and bottom positions of the handheld device case, which results in a distance of approximately 15 *cm*, 7.5 *cm*, and 2 *cm* to the wrist wearable, respectively. We observe that when the motor is at the middle position, the performance slightly degrades (see fig. 3.8a), while for top and bottom positions the resulting mean *BERs* are comparable. We further observe that in the middle motor position, for half the participants the resulting *(*BER) values are lower than 0.15 (see fig. 3.9a) with five of them having *BERs* lower than 0.05. For the top (farthest from the receiver) motor position, we observe the best mean *BER* of 0.10 ($\sigma = 0.13$) and that half the number of participants have *BERs* lower than 0.05. In the bottom position, we have similar performance with a mean *BER* of 0.12 ($\sigma = 0.10$) and 50% of the participants with a *BER* less than 0.10. We observed that when the motor is positioned at the top of the handheld device, although farthest from the receiver (accelerometer), the vibrations makes the whole handheld device slightly more agitated. This, in turn, results in a stronger vibration signal reaching the accelerometer at the receiver. A similar effect is observed when the motor is mounted on the bottom of the handheld device. When the motor is mounted on the middle of the device, the vibrations are slightly dampened, accompanied by the fact that human palm is curved in the middle and thus the direct surface area that may be coming in contact with the hand when vibrating is smaller, which is reflected in the slightly higher *BER* values. From fig. 3.9d, which depicts the cumulative probability distribution of *BER* for each transmitter-receiver distance, we can see that for nearly 70% of the cases in the top and bottom positions, the achievable *BER* is less than 0.15.



(a) (b)

Figure 3.8: Mean BERs: (a) distance between the motor and the motion sensor, (b) wrist wearable orientation.

3.4.2 Effect of Wrist Wearable Orientation

Next we evaluate the effect of the wrist wearable's orientation on system performance, with the communication direction from the handheld device to the wrist wearable (fig. 3.7b). Here, we see (fig. 3.8b) slightly lower *BERs* when the wrist wearable is worn facing downwards (accelerometer located at the bottom of the wrist) compared to when facing upwards (accelerometer located on the top of the wrist), with *BER* values of 0.12 ($\sigma = 0.13$) and 0.16 ($\sigma = 0.13$), respectively. fig. 3.9b shows that half the participants had *BERs* less than 0.10 for the face down orientation. We believe that this is because when the vibration motor and motion sensors are aligned on the same side of the hand, the vibrations have a more direct path to travel. Further, from fig. 3.9e, we see that for the face down orientation, close to 70% of the time *BER* will be lower than 0.2, as opposed to 60% probability in the face up orientation.

3.4.3 Performance Under Noise

Next we evaluate *SkinSense* under two different types of movement noise that impact motion sensor readings. The first type of noise occurs when the user is relatively stationary while traveling inside a moving vehicle (with vehicle speeds between 16 kmh to 64 kmh). Due to practical limitations, we simulated this experiment by superimposing prerecorded motion sensor data recorded from the handheld device while inside a moving vehicle, to the motion data collected from partici-

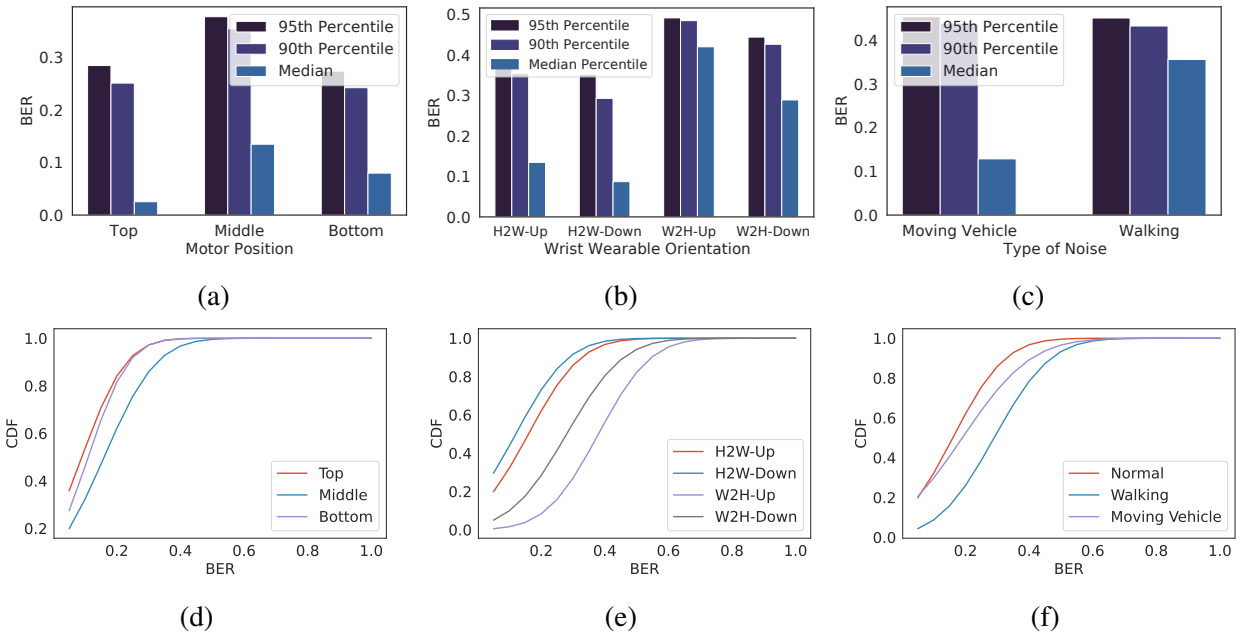


Figure 3.9: BERs of participants at percentiles 95^{th}, 90^{th}, and median: (a) distance between the motor and the motion sensor, (b) wrist wearable orientation, (c) effect of different types of noise. CDF of BER: (d) distance between the motor and the motion sensor, (e) wrist wearable orientation, (f) effect of different types of noise.

pants in a lab setting. The second type of noise we experimented with occurs due to user movement, where participants were asked to walk with the handheld device and wrist wearable on the same hand while *SkinSense* is executing. The first noise scenario (moving vehicle) shows slightly better performance with a mean *BER* of 0.19 ($\sigma = 0.17$). Further, 7 out of the 13 participants showed *BERs* less than 0.10 (fig. 3.9b), with only 3 participants showing significant degradation of *BERs* of over 0.3. This indicates that *SkinSense* is fairly robust against movement noise resulting from *plane motion*, such as traveling inside a vehicle. For the second type of noise scenario (walking), the *BER* considerably degrades with an observed mean *BER* of 0.29 ($\sigma = 0.28$). For both the above scenarios, the communication direction was from the handheld to the wrist wearable. Moreover, with built-in APIs [69] about user movements on modern mobile operating systems, such as Android and iOS/watchOS, it is also possible to contextually turn off *SkinSense* when the user is not stationary.

3.4.4 Effect of Sampling Rates

To evaluate the impact of reduced motion sensor sampling rates on protocol performance, we test *SkinSense* with the accelerometer (at the receiver) sampled at 200 *Hz*, compared to the 700 *Hz* sampling rate which was used previously. We choose a sampling rate of 200 *Hz* because it is the maximum sampling rate that is achievable on most modern consumer level mobile and wearable devices. Based on the results obtained in this setting, we observe that the *BER* of the communication scheme degrades to around 0.4 ($\sigma = 0.06$) when a lower sampling rate is used. This indicates that such a low accelerometer sampling rate at the receiver is probably not sufficient for capturing the range of vibration frequencies that we are utilizing to encode data in our transmission algorithm. With the limitation of not being able to use multiple frequencies, achieving a better *BER* would only be possible at the cost of a reduced bit rate.

3.4.5 Half-Duplex Communications

In the experimental settings discussed so far, we have only considered communication in the direction from the handheld device to the wrist wearable. However, as modern wearable and handheld devices come equipped with both a vibration motor and motion sensors, it would be extremely useful to evaluate communication in the reverse direction to what we have evaluated so far (i.e., from the wrist wearable to the handheld, in our case). If feasible, it would grant a half-duplex property to the communication channel, which would enable a whole set of applications that require communication in both directions. As our proposed communication protocol and hardware setup is amenable to such an evaluation, we also test communications in the reverse direction, i.e. using the wrist wearable as a transmitter and the handheld device as a receiver. Overall, the achieved performance results are significantly lower compared to the earlier case, as seen in the fig. 3.8b and fig. 3.9b with overall *BERs* dropping to 0.38 ($\sigma = 0.13$) for when wrist wearable is facing upwards and 0.28 ($\sigma = 0.14$) for when the wrist wearable is facing downwards. Similar to the wrist wearable orientation experiment (section 3.4.2), we observe that when the wrist wearable is facing downwards a lower *BER* can be achieved. We also conduct the same

noise-related experiments (similar to section 3.4.3) for this setting and observe a similar pattern in the performance results where the user movement (walking) experiment degraded the *BER* to 0.43 ($\sigma = 0.02$) while the vehicle movement experiment reduced it to 0.42 ($\sigma = 0.13$). These low performance results can be attributed to the fact that the wrist wearable surface area in contact with the wrist/hand is much smaller compared to the handheld device. This may result in reduced perception of the generated vibrations by the the skin/hand, thus resulting in an overall reduced performance.

3.4.6 Consumer-grade Hardware Setup

Next we discuss the performance of *SkinSense* when using commercial consumer-grade hardware (as summarized in section 3.3.4). With the highly constrained access to the vibration motor and motion sensors on commercial mobile and wearable devices, *SkinSense* when executed on these devices could only achieve low overall *BERs* of around 0.4, for both the watch wearing orientations. We believe that one of the main constraints is the software-restricted sampling rates of motion sensors in commercial smartphones/smartwatches, which limits the maximum allowable sampling rate to only 200 *Hz*. Further, the Android API also restricts the frequency modulation of the vibration motor on these devices, which prevents us from using the embedded motor at its full operating capacity. These factors restrict us from using the 4 PWM frequency bands (of the motor) for communication which we used in the custom hardware case. As a result, we are not able to modulate in the PWM frequency band in the commercial device case, which effectively brings down the achievable bit rate from 6.6 to 3.3 *bps*. Although the reduced performance of *SkinSense* can be primarily attributed to the software and hardware limitations of existing commercially-available mobile device hardware, we believe that better motion sensors (with high sensitivity and sampling rates) and vibration motors in future devices will result in a slightly more favorable outcome for such vibration-based communication systems.

Table 3.1: Mean BERs for accelerometer and gyroscope.

Direction	Accelerometer	Gyroscope
Handheld to wrist wearable	0.16 ($\sigma = 0.13$)	0.39 ($\sigma = 0.09$)
Wrist wearable to handheld	0.38 ($\sigma = 0.13$)	0.41 ($\sigma = 0.10$)

3.4.7 Accelerometer vs. Gyroscope

Our preliminary experiments involving both the custom and consumer-grade devices demonstrated that accelerometers produced better feedback than gyroscopes in our setup. To comprehensively compare the impact on performance when a gyroscope is used as a receiver as opposed to an accelerometer, we perform some additional experiments using our custom hardware setup. Based on the observed results (see table 3.1), we can conclude that our protocol produces better performance (lower *BER*) using the accelerometer. The achievable *BER* drops from 0.16 to 0.39 when a gyroscope is used as a receiver (when transmitting from the handheld device). In contrast to accelerometers, gyroscopes measure a device's angular velocity and it is likely that surface vibrations, which are already dampened down as they travel through the human skin/body, do not produce a significant amount of angular motion to be discernible on a gyroscope.

3.4.8 Failed Transmission Detection

After further scrutinizing the transmitted messages with higher *BERs*, we observed that these inaccuracies are mostly caused due to missing bits, which may be due to voluntary or involuntary hand movements occurring during data transmission. To overcome this, we propose a message length based verification at the receiver. For this, if we assume that the receiver knows the length of the incoming message, or that the message length is fixed. Incorrectly or erroneously received messages can be easily identified (and flagged) based on bit length mismatches. In case of such a mismatch detection, the receiver can request a re-transmission. We observe that using such a heuristic significantly improves the *BERs* in our custom hardware setup, where *BERs* dropped below 0.04 when erroneously received messages are correctly identified for re-transmission.

3.4.9 In-Depth Analysis

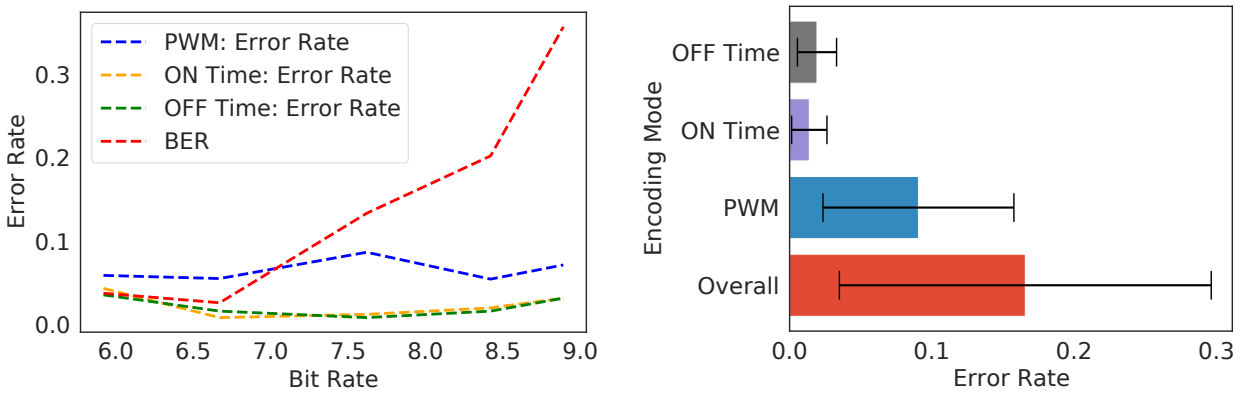
Bit Rate vs Error Rate

As discussed before, to gain higher transmission speeds we can reduce the `ON` times of the encoding algorithm, but this directly affects the accuracy which can be seen in fig. 3.10a. At higher bit rates, starting around 7 *bps*, *BER* steadily increases. However, we believe that with the use of more advanced, highly sensitive motion sensors that could be operated at higher sampling rates, *SkinSense* could achieve higher transmission speeds while maintaining a low error rate.

Figure 3.11a shows the confusion matrix for PWM-based symbols, and we observe that most errors occur between adjacent PWMs (e.g. 20 and 30, or 60 and 100). As previously discussed in section 3.3.2, we chose the PWMs to be 20, 30, 60, and 100 after observing that they have minimal confusion with each other when decoded via the accelerometer signal. However, when testing under realistic settings with possible variations in bone structure of hands of different users, along with variations in the way they hold a handheld device in their palm, we observed that some of these PWM-based frequency vibrations could get picked up by an accelerometer differently. Essentially, if we are able to use additional PWMs (i.e. vibration frequencies) as carriers to encode data, we would be able to achieve a higher bit rate. But due to the fact that accelerometer sensors are not able to distinctly identify some of the adjacent frequencies, it limits us from achieving a higher bit rate. In other words, higher bit rates would come at the cost of higher error rates (*BERs*). This is further confirmed by Roy et al. [157], who also highlighted in their work that high energy vibrations, occurring in the resonant frequency band, could potentially interfere with neighboring frequencies, which also limit the number of usable frequency bands in such vibration-based communication schemes.

Analysis of Individual Encoding Parameters

We also analyze each of the individual parameter modes in our algorithm which we use to encode data, i.e. PWMs, `ON` times and `OFF` times. We analyze the error rates of these modes



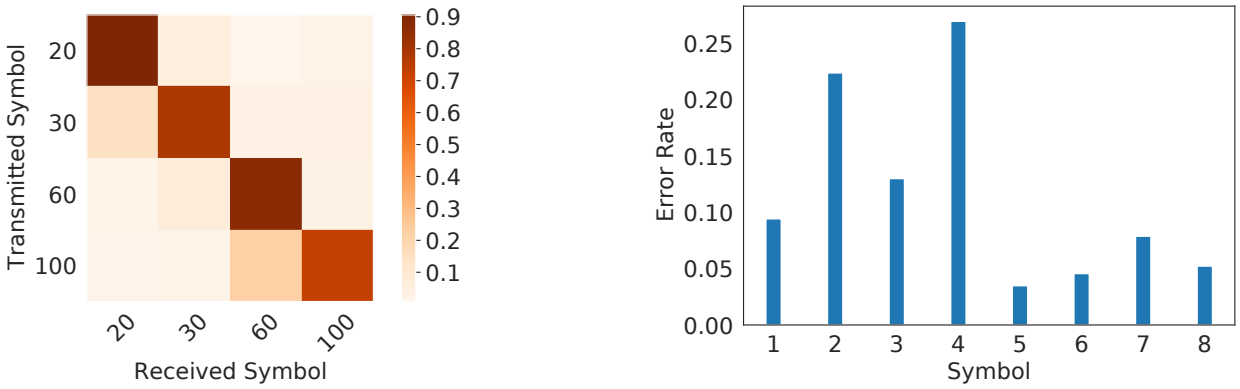
(a) (b)

Figure 3.10: Individual encoding parameters (a) Error rates vs bit rate (bit/s), (b) Error rate comparison.

individually to understand which modes work the best, in terms of bit rates or bit error rates. fig. 3.10b shows individual error rates for each of these modes. We see that the time-based modes (`ON` and `OFF`) perform the best with overall error rates less than 0.02, while the frequency-based mode (PWMs) show a relatively higher error (0.10). Although the time-based modes work more reliably than the PWM, the amount of modulation that can be done using time-based modes would be limited due to them being directly affecting the bit rates/transmission speeds. Figure 3.11b indicates the error rates for each individual symbol where numbers 1-4 are PWMs, 5 and 6 are `ON` times and 7 and 8 `OFF` times. This further shows that time-based symbols (5-8) perform the best as opposed to the PWM-based symbols. The observations made in fig. 3.11a is further clarified here as it can be seen that PWMs 30 and 100 show the highest error rates due to them being misidentified as 20 and 60, respectively.

3.4.10 Acoustic Side-Channel

Roy et al. [157] recognized the possibility of information leakage via the noise produced during vibrations, and proposed a mechanism to jam the acoustic side-channel by emitting a noise from the transmitter to suppress the sound produced by vibrations. They further studied the *sound of vibrations (SoV)* for different surfaces that the transmitter may be placed upon, and observed that glass surfaces cause the highest side-channel leaks, i.e. produces the loudest noise. To understand



(a) (b)

Figure 3.11: (a) Confusion matrix of transmitted and received symbols. (b) Per symbol error rate.

Table 3.2: Ratio of sound intensity of vibration signals to ambient noise.

Setup	**Lab**	**Apartment**	**Vehicle**	**Supermarket**
Custom hardware setup	2.11	2.20	0.92	0.80
Consumer level smartphone	1.46	1.52	0.63	0.58

the severity of such a vulnerability in our proposed system, we similarly measure the SoV when a hand is used as the communication medium. Table 3.2 provides the ratio of SoV of our custom and consumer-grade device setup (recorded 2 *ft* away from the transmitter) to the ambient noise at various locations including, a school laboratory, an apartment, inside a moving vehicle and in a supermarket. The lower the ratio (closer to zero), the quieter the SoV is relative to the ambient noise, thus, reducing the risk of acoustic leakage. We observe the SoV to ambient noise ratio to be high in quieter locations such as an apartment or a school lab, while the ratio falls below 1 for louder locations such as a vehicle or a supermarket. This is in contrast to Roy et al.'s [157] results, where the ratio was around 1.5 for all these locations. Compared to the consumer-grade smartphone setup, our custom setup (as seen in table 3.2) produces much louder sound ratios, which is likely because of the vibration motor being mounted on the custom handheld device without an enclosure. The acoustic leakage from *SkinSense* is significantly lower when consumer-grade mobile devices are used due to the enclosed nature of the vibration motor in such devices. We believe that this threat can be minimized by employing an acoustic jamming mechanism similar to the one proposed by Roy et al. [157] by utilizing a speaker near the transmitting device.

3.4.11 Practical Significance of the Bit Rate

To understand the practical usefulness of the bitrate afforded by *SkinSense*, we further analyze the transfer time for practical device-to-device secrets such as 4-digit PINs and 8-character passwords. For this analysis, we consider the data transfer scenario from the hand-held device to the wrist-wearable at a distance of 7.5 cm. We observe that transfer of a 4-digit PIN takes a little more than 5 seconds, while an 8-character password takes approximately 10 seconds. Although these transfer speeds are considerably slower compared to short-range radio technologies such as Bluetooth and NFC, it must be noted that *SkinSense* is envisioned to be only used as a potential secure side-channel to supplement traditional radio-based communication channels. Specifically as observed above, *SkinSense* can enable usage scenarios such as securely proving device co-location or secure authentication in the absence of reliable radio channels by enabling sharing of short secrets at reasonable side-channel speeds.

3.4.12 Energy Requirements

To evaluate the energy requirements of *SkinSense*, we conducted an experiment to measure the amount of battery energy consumed over time. For the transmitter, we tested on two smartphones by sending 30 messages over a period of 1 hour. For the receiver, we tested two smartwatches by running the motion sensor data collection for 30 messages over a period of 1 hour. As seen in table 3.3, energy consumption is only around 150 mAh for Moto G7 Plus (2019) which is a newer device with Android 10, compared to the 300 mAh for an older smartphone, Nokia 6 (2016), with Android 9. The receiver smartwatches show a similar energy consumption pattern with the newer Ticwatch with only 12 mAh consumption compared to 14 mAh in the older Sony W3.

Table 3.3: Energy consumption.

Device	**Battery Capacity**	**Energy Consumed**
Sony W3	420 mAh	14 mAh
Mobovi Ticwatch E	300 mAh	12 mAh
Nokia 6	3000 mAh	300 mAh
Moto G7 Plus	3000 mAh	150 mAh

Table 3.4: Comparison with related works.

Related Work	Setting	Sensors	Technique Used	Data Rate	Performance
VibeRing	Ring on finger to device	Accelerometer	Machine Learning based	12.5 bps	0.05 (BER)
Ripple	2 devices on a solid surface	Accelerometer	Frequency domain based	200 bps	0.017 (BER)
Vib-Connect	2 devices in direct contact (mobile phone and laptop)	Accelerometer	Time domain thresholding based	Not mentioned	100% (Accuracy)
SYNCVIBE	2 devices in direct contact	Accelerometer	Time domain based	20 bps	0.005 (BER)
SecureVibe	Device on top of a implanted medical device	Accelerometer	Time domain based	20 bps	Not mentioned
Skin Sense (this work)	Between handheld device to wrist-wearable via skin	Accelerometer	Frequency time domain domain based	6.6 bps	0.10 (BER)

3.4.13 Comparison with Previous Works

Sen and Kotz [167] proposed a vibration-based communication scheme using a smart ring, where the smart ring act as the vibration transmitter to communicate with Internet-of-Things (IoT) devices embedded with accelerometers. They were able to achieve a *BER* of 0.05 with a bandwidth of 12.5 bps. Similar to many of the above mentioned works, their smart ring transmitter was required to be *in direct contact* with the receiver IoT device for reliable communication. The closest work to ours, in terms of using the human body/skin as the communication medium, is by Ma et al. [114]. They proposed a Multiple-Input-Multiple-Output (MIMO) communication scheme using two vibration motors and two piezo transducers. Due to limitations of vibration motors such as ramping time and the volatile nature of human skin as a channel, they use two motion sensors (acceleromter+gyroscope) embedded at the transmitter side, in order to acquire and utilize additional Channel State Information (CSI) by employing deep learning. Their MIMO scheme was able to achieve a MIMO capacity of about 5 bps/Hz, which is more than twice the capacity that could be achieved using a comparable Single-Input-Single-Output setup. However, there are several drawbacks in their proposal. First, their scheme is not very practical because it relies on extremely customized hardware and setup, often not found in commercial mobile and IoT devices. Moreover, their scheme employs deep learning algorithms that typically require large amounts of training data for acceptable performance, which may not be trivial to obtain for all communication settings, conditions, and individual users. Lastly, performance evaluation of their scheme was done using data

from only two human subject participants, and so it is unclear how generalizable their results are. Another similar work (SecureVibe) which uses skin as a medium was proposed by Kim et al. [94] using a smartphone vibration motor and a custom made implantable medical device equipped with an accelerometer. Although, their work achieves bit rates around 20 bps, their setup requires the smartphone to be directly on top of the implanted device under the skin with only 1 cm distance between the motion sensor and the smartphone. We demonstrate that *SkinSense* framework could be effective up to a distance of 15 cm between the handheld smartphone (vibration motor) and the wrist wearable (motion sensor). Further, their work was only evaluated using an emulated human body model and not on actual human subject participants. In contrast, *SkinSense* is proposed for two externally held/worn devices, and the communication protocol was tested with human subject participants. Shah et al. specifically studied the vibration propagation via skin of human arm and reported that even at a mere distance of 4 cm [169], the vibration intensity drops around 70-80% and continues to drop over 90% at distances over 8 cm. In contrast, in a rigid medium such as a wooden board, the authors of *Ripple* [157], observe that vibration intensity gradually increases up to 15 cm before attenuating. This allows them to use 10 different vibration amplitudes to transmit data and still be able to accurately distinguish them during the demodulation to achieve a relatively higher data rate of 200 bps. It should be noted that on a medium with low rigidity such as the human skin, propagation of the vibration signal attenuates much more quickly at longer distances. This makes *SkinSense*'s design challenge a rather non-trivial one and with the same token makes adaptation of techniques employed by protocols such as *Ripple* [157] in this scenario infeasible.

3.5 Discussion

User's Perception and Preferences. As vibrations carried by the human skin/body is perceptible to the end-user, we deployed a short survey to our 13 participants to gauge their feelings about the *SkinSense* protocol. Based on the received survey responses, 38% of the participants noted that they were not bothered by *SkinSense*'s operation, while 54% were bothered slightly and only 8% were highly bothered. Although users can take advantage of *SkinSense* by easily

switching (or transfering) the hand-held device to the wrist-wearable device hand, we also studied users' preferences regarding *SkinSense* usage. When asked about which hand they use to hold a handheld device/smartphone, around 62% answered that they use the same hand as the wrist wearable wearing hand followed by 23% who use both the hands. The remaining 15% said that they use their non-smartphone holding hand to wear the wrist wearable. Findings of this short survey does indicate that a vibration-based communication protocol such as *SkinSense* is usable (from an end-user perspective) in practice, although more extensive usability and user-satisfaction surveys are needed.

Other Side-channels. In addition to the acoustic side-channel threat, there is a possibility of an attacker physically attaching an eavesdropping motion sensor to the communication surface [157]. We believe that such a type of threat is unlikely in our proposed communication setup/protocol because we use the human body/skin as the communication channel. An adversary is unlikely to be able to directly attach an eavesdropping device to a victim user's body/skin without their cognizance. The transmitted messages could also be encrypted to further minimize the possibility of a contact based attack. Further from the security perspective, as *SkinSense*'s main goal is to provide a secure channel against external eavesdropping devices, we assume that both the transmitter and receiver devices are fully trusted, executing only trusted (or non-malicious) apps.

3.6 Related Works

Vibration-based communication techniques that employ vibration motors and motion sensors (esp. accelerometers) have been previously studied for various forms of underlying communication mediums (or channels), such as hard surfaces, direct device-to-device contact, and also via human skin. Yonezawa et al. [203, 204] proposed a mechanism to send information from a smartphone to a laptop computer and achieved a data rate up to 10 bps. Their proposal encodes information in a vibration signal emitted by the smartphone, which needs to be kept in *physical contact* of the laptop, and the laptop detects the vibrations (with the encoded information) via an embedded or on-board accelerometer. Lee et al. [98] proposed a similar communication framework (SYNCVIBE)

where the 2 devices require *physical contact* using a smartphone as a vibration transmitter and an external accelerometer device as a receiver attached to it and were able to achieve data rates around 20 bps with a BER of 0.005. Hwang et al. [84] proposed a similar communication mechanism, but between two smartphones placed on a solid surface, such as a wooden table, metal or plastic shelf, a stack of paper, and a cushioned chair. Their scheme achieved over 90% accuracy for all the surfaces when the devices are placed *roughly 25* cm from each other. However, the accuracy drops significantly when the two devices are too close (10 cm) or too far apart.

Roy et al. [157] proposed a similar communication framework, named *Ripple*, which achieved a data rate of up to 200 bps by using custom off-the-shelf vibration motors and accelerometer chips, and up to 80 bps by using consumer smartphones. However, like Hwang et al. [84], they employed *solid surfaces* such as wooden and glass tables as the communication medium (or channel) in their scheme. In a follow-up effort, Roy et al. [156] proposed another vibration-based communication technique, but this time by using a microphone instead of an accelerometer as the receiver. Evaluation of their follow-up proposal showed that a smart ring with a vibration motor can achieve a bandwidth or data rate of around 7 $kbps$, while for a smartwatch the bandwidth drops to 2 $kbps$. However, an important requirement of their scheme was that the transmitting device had to be in *close proximity* to the microphone-based receiver to achieve these bandwidths.

In addition to these wearable device based related works, several other works have explored cyberphysical systems such as UAVs to propose the use of acoustics of their motors for communication and fingerprinting [24, 148].

When considering non-radio based body area networks, several other technologies have been proposed in the literature. One such area is the use of bioacoustics [175, 206]. Zhang et al. [206] proposed a method for communication by using a bone transducer as a transmitter and an accelerometer as a receiver. However, acoustic based methods are often too sensitive to background noises and are susceptible to eavesdropping attacks. Apart from acoustics, more recent works have explored the use of optical based wireless communication [41, 53]. A major drawback in optical techniques is the line of sight requirement where each device in the body should be in the view of

the other in order for the communication to succeed.

3.7 Conclusion

In this work we explored a novel form of communication between a handheld device and a wrist wearable by using a vibration motor transmitter and accelerometer-based receiver with human skin/hand being used as the communication medium. Since the human hand could have various anatomical and biomechanical differences among different people, we tested our proposed scheme under multiple realistic settings with 13 human subjects. Our proposed scheme was able to achieve a sustainable bandwidth of 6.66 *bps* while keeping the BER below 0.10. Although, the current consumer level smartphones and wrist wearables have limitations, resulting in our scheme not being able to perform optimally, we believe that our work opens up further research in the area related to vibrations and human body-area communication channels.

Acknowledgment

Research reported in this publication was supported the National Science Foundation (NSF) under award number 1943351.

CHAPTER 4: SENSORY SIDE-CHANNEL ATTACKS ON WEARABLES: HANDWRITING INFERENCE USING WRIST-BASED MOTION SENSORS REVISITED.

**The contents of this chapter and the reported experimental results have previously been published in the Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks (WiSec, 2019), co-authored by Raveen Wijewickrama, Anindya Maiti and Murtuza Jadliwala (in that order). The contents of the published manuscript has been reproduced here with revisions.*

4.1 Introduction

Inference of sensitive user data by employing on-board sensors as information side-channels has been a significant privacy concern ever since the inception of commercial, consumer-grade wearable devices such as smart watches and fitness bands. Several proposals in the research literature have already demonstrated how data from zero-permission wrist-wearable sensors can be abused to infer keystrokes, user-activities, and behavior [90, 104, 115, 117, 118, 132, 159, 174, 189, 191, 196]. In the same vein, multiple research efforts have also demonstrated the feasibility of inferring handwritten text using motion sensors (such as accelerometers and gyroscopes) present onboard these wrist-wearables. Some of the initial efforts in this direction showed the feasibility of inferring larger handwriting gestures, such as, writing on a whiteboard [19] or using hand/finger movements to write in the air [8, 9, 199]. More recent efforts have focused on inferring smaller and more natural handwriting gestures, such as, writing on a paper with pen/pencil [198]. Some of these works were presented with an adversary in mind, whereas others were presented merely as a mobile/wearable application or service. In this work, we focus on the problem of inferring handwritten text primarily from an adversarial point of view.

While these earlier research efforts concluded that their inference/classification frameworks were able to infer handwritten English letters and words from wrist-wearable motion data in an accurate and feasible manner, we observed that several of the assumptions made by them (implicitly or explicitly) were either not realistic or impossible to include in an adversarial setting. For exam-

ple, Amma et al. [9] used specialized motion sensors and custom wrist-wearable hardware in their inference framework, which could sample at more than $800Hz$. However in practice, most common commercially-available smartwatch and fitness-band motion sensors have maximum (peak) sustainable sampling rates of around $200Hz$. The availability of such specialized sensors and hardware, and the extremely fine-grained motion data generated by it, may result in an accurate inference of handwriting, but would be difficult to assume in an adversarial setting where the target user is probably just wearing a consumer-grade wrist-wearable with sensors that have limited capabilities. Other limitations of some of the previous works include testing primarily in a personalized setting (training and testing data collected from the same participant), vague definition of segmentation techniques used to separate individual sentences and words, and disregard for varying writing styles of the same target user. The absence of these factors in their evaluation also gave us the impression that their data may have been collected in a tightly controlled fashion, which is not reflective of participants' natural handwriting and/or writing in a natural setting.

Motivated by these shortcomings of existing research efforts, in this work we attempt to validate if the current empirical results on handwritten text inference using wrist-wearable motion sensors are generalizable and applicable under more practical adversarial settings and handwriting scenarios. Broadly, our goal in this work is to evaluate if existing handwriting inference frameworks are a genuine and realistic threat to a wrist-wearable device user's privacy and security. In order to accomplish this goal in a structured fashion, we first closely replicate the four most notable inference frameworks in this direction, specifically, the ones defined by Xu et al. [199], Arduser et al. [19], Amma et al. [9], and Xia et al. [198]. Next, by means of contemporary, consumer-grade wrist-wearables, we collect natural handwriting related motion data from a large number of human subject participants in an unconstrained and non-restrictive setting for a variety of different writing scenarios. In order to showcase why it is much more difficult to infer natural handwriting, we then perform a detailed comparative analysis of our results with those obtained by previous efforts, across different writing scenarios. Our final contribution is an in-depth discussion on the factors that affect the success of handwriting inference attacks in real-life, supported by data and results

obtained from our experimentation.

4.2 Adversary Model and Background

Before outlining details of our replication and validation experiments, let us first provide an unambiguous description of the adversarial setting and capabilities assumed in this work, followed by a brief technical background of handwritten text inference. We also provide a detailed review of the four notable wrist motion-based handwriting inference schemes in the literature and summarize the key research gaps that we attempt to fill in this work.

4.2.1 Adversary Model

The use of hands (and gestures) to compose lingual texts continues to be one of the most prevalent methods of communication and many of these handwritten or hand-gestured text may contain sensitive information, such as personal identifiers and financial credentials. Our adversary in this work is as a malicious actor or entity whose goal is to infer such private handwritten text by employing some form of an information side-channel. While there exists several types of information side-channels, our focus in this research work is on zero-permission motion sensors (such as accelerometers and gyroscopes) found on most modern wrist-wearable devices such as fitness-trackers and smartwatches. Our adversary can gain access to this motion sensor information by tricking the victim user into installing on its wearable device a Trojan or malicious application that is masquerading as some useful application (for example, a game or a fitness application). Once this Trojan application is installed on the victim's wearable, it can sample and exfiltrate the motion sensor data back to the adversary using the device's network connection. The fact that popular wearable operating systems like Android Wear and watchOS allow third-party applications unrestricted access to back-end sensors such as gyroscope and accelerometer (thus, the term zero-permission), enables the malicious Trojan application to sense and exfiltrate this data without raising any flags or violating any system security policies. By masquerading as an application that would normally require access to these (motion) sensors for its operation, the malicious Trojan

application can achieve further stealth.

Once the adversary is able to remotely archive this exfiltrated sensor data, say, on its computation server, he can analyze it in an offline fashion to infer the actual written information from the data with as high accuracy as attainable. Such an adversary model is practical and has also been commonly employed in the literature for studying similar privacy threats due to zero permission mobile and wearable device sensors [104, 115, 117, 119, 159, 190, 191].

In this work, we limit ourselves to the problem of inferring handwritten text in the English language only. This enables us to have a comprehensive and equitable comparison with other research efforts that have also inferred only English language written text. Additionally, we also assume that our adversary only employs the victim's hand movement data while writing, as perceptible on the victim's wrist-wearable motion sensors, for the inference attack. The adversary does not employ additional information, such as contextual dictionaries on the topic of writing and victim's language abilities, in order to improve the accuracy of the inference attacks. This is done to keep the adversary model practical and to achieve an equitable comparison with the inference frameworks being evaluated in this work. That being said, the adversary is free to use a generic language dictionary and well-known spelling correction techniques for improving handwriting inference.

4.2.2 Inferring Handwritten Text from Wrist Movements

Any framework for inferring handwritten text from wrist movements (or wrist motion sensor data) would ideally comprise of the following two key phases: (i) identifying the writing and non-writing parts in the sensor data stream in order to segment strokes, alphabets, words, and sentences, and (ii) using the segmented sensor data to perform character, word, or sentence recognition/classification. However, before we outline a concrete technical framework for handwriting inference from motion sensor data, let us first characterize the different writing styles and writing elements that may vary from person to person, and sometimes also for the same person. Writing in English language can be broadly categorized into the following two styles: lower versus upper case writing, and cursive versus non-cursive writing [30]. Irrespective of the style, writing in

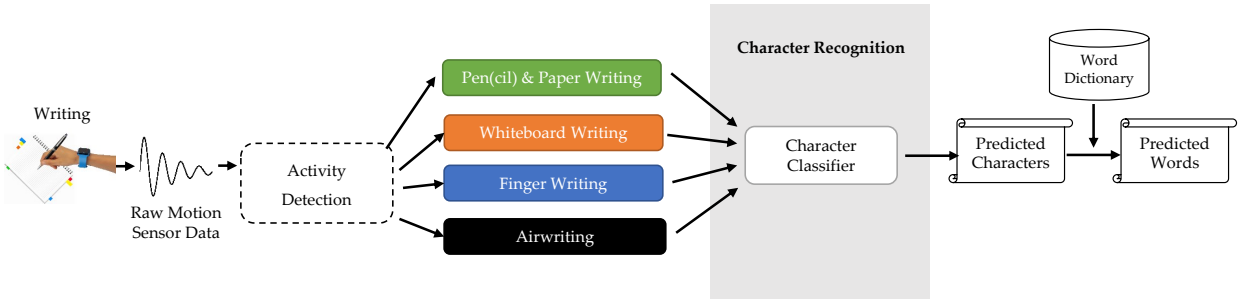


Figure 4.1: Generalized attack framework.

English (or any other) language comprises of drawing a series of alphabets, where each alphabet is nothing but a series of one or more strokes made with a writing apparatus such as a pen. A *stroke* can be defined as a continuous line drawn in one go, starting with a "pen-down" action and ending with a "pen-up" action. The same concept applies to all forms of writing, except that the pen can be replaced with another writing apparatus. Each handwritten alphabet can be uniquely described by the **number**, **direction** and **order** of strokes. It is possible to write the same alphabet in different cases with different number, direction and order of strokes. Even in the same case, the same alphabet can be written using a different number, direction and order of strokes. As a matter of fact, a written alphabet comprising of n strokes can have $n!$ different stroke order and directions depending on the writer. Thus, it is easy to imagine that handwriting related wrist or hand movements could differ significantly even for the same alphabet. For example, written in the same case, but with different number, direction, and/or order of strokes. These differences in movement are reflected on the motion sensors located on writer's wrist, resulting in different sensor data streams for the same alphabet. This is a significant challenge that must be overcome by any motion-based handwriting inference framework. It is worth pointing out that wrist movement-based handwriting inference is significantly different, and more challenging, than traditional image or pixel-based handwriting recognition [133, 144], primarily because image or pixel data does not include/consider information about the number, order and direction of strokes.

4.2.3 Previous Work and Motivation

As noted earlier, we are not the first to investigate handwriting inference threats using wrist-wearable motion sensors as an attack vector. Multiple previous research papers have demonstrated the feasibility of inferring different forms of handwritten information from motion data collected by means of wrist-wearables. We are particularly interested in the following four forms of handwriting scenarios that were evaluated earlier, primarily because they are the most commonly observed in real-life situations: (i) pen(cil) and paper writing [198], (ii) whiteboard writing [19], (iii) finger writing [199], and (iv) air writing [7–9]. In this section, we describe the main strengths and shortcomings of these earlier research efforts, and outline our primary motivation for revisiting the problem of handwriting inference using wrist-wearables.

Pen(cil) and Paper Writing: Xia et al. [198] proposed an eavesdropping attack on the classical pen and paper based writing scenario using motion data recorded from a smartwatch worn on the writing hand. The threat was accomplished using wrist motion data that was sampled from the smartwatch's accelerometer and gyroscope at $200Hz$. Most modern smartwatches and fitness tracker motion sensors do support this sampling rate. The authors' employed a thresholding based word-wise segmentation on the continuous accelerometer data followed by an alphabet-wise segmentation using the gyroscope signal before classifying individual alphabets. The authors' did study a generalized setting for their classification algorithm, where training and testing data from different participants was used, making it realistic because it is generally difficult (or impossible) for an adversary to collect labeled training data from the victim or target. Given the above strengths, this work also has several shortcomings. First, the proposed inference framework is limited to non-cursive handwriting, and only lowercase alphabets were evaluated with the argument that, it can be easily extended for uppercase alphabets. Moreover, the framework is also difficult to replicate and generalize to other writing scenarios and settings due to the use of fixed thresholds (during segmentation) and specific features (during alphabet inference).

Whiteboard Writing: Arduser et. al [19] proposed an inference framework that employs accelerometer and gyroscope data from a target victim's smartwatch to infer text written on a white-

board. Besides the standard learning-based alphabet classification routine, the inference framework comprised of a pre-processing routine that first converted the motion data from device coordinates to whiteboard coordinates, which eliminated the effect of watch orientation when writing on different (top or bottom) areas of the whiteboard. Moreover, the motion data was sampled from standard consumer-grade smartwatches, which shows that such threats can be executed in the wild without using any specialized hardware. This research effort also suffers from several significant shortcomings. First, the proposed inference framework was used to evaluate only uppercase alphabets. Moreover, it is unclear whether the proposed framework can be easily replicated and generalized to other writing scenarios and settings as critical parameters (and description) related to the coordinate conversion routine and employed sensor sampling rates are unavailable. Lastly, it is unclear whether the provided empirical results are for a personalized (training and testing data from the same participant) or a generalized (training and testing data from different participants) classification setting.

Finger Writing: Xu et al. [199] investigated the problem of alphabet recognition (with each character approximately $2.5'' \times 2.5''$ in size) when writing using the index finger on a surface by means of a Shimmer [172] device worn on the wrist of the writing hand. One of the most significant outcome of this research effort was that the authors were able to obtain very high (more than 90%) inference accuracy for their proposed inference framework. At the same time, one of the biggest drawback of their work was the use of sophisticated Shimmer devices in the inference framework, which are not as ubiquitous and popular as commercially available smartwatches and fitness-bands. In addition to this, the proposed inference framework was used to evaluate only writing of uppercase alphabet letters. This raises serious concerns about the broad applicability and generalizability of the proposed framework to other wrist wearable hardware and writing scenarios/settings. Also, the paper neither mentions the total number of unique participants for which the proposed inference framework was evaluated, making it hard to understand the statistical significance of the obtained results, nor does it provide details on whether a personalized or a generalized setting was used for the classifier evaluation.

Air Writing: Amma et. al. [7–9] proposed an input mechanism (named *airwriting*) which detects and classifies hand writing gestures in the air using the motion sensor data collected from a specialized sensor-integrated hand glove. By comprehensively evaluating their inference frameworks in both personalized and generalized settings, the authors show that it is possible to obtain a reasonably high inference accuracy in this writing scenario/setting. However, while the use of a custom-designed hand glove capable of sampling motion sensors at $819.2Hz$ is viable for enabling novel HCI applications, such kind specialized hardware is not very popular and ubiquitous. This significantly limits the applicability of the above inference framework and the related results in an adversarial setting. Moreover, the proposed inference framework was used to evaluate only uppercase words from a dictionary as it did not include appropriate segmentation algorithms for separating out the alphabets within each word. This reliance on a dictionary for executing the inference algorithm significantly limits the type of information that can be inferred and is not very practical or realistic in an adversarial setting.

As evident from the strengths and shortcomings of each of the above research efforts, it was difficult for us to reasonably estimate whether these threats to users' handwritten information from current consumer-grade wrist-wearable sensors is practically feasible or not. And if it is, how would such an attack perform across a diverse group of users with different and unique handwriting styles? And, is there a way to develop a unified inference framework that will not only work against diverse handwriting styles, but also different forms and cases of handwritten text? In order to fully understand the extent to which end-users must be concerned about the possibility of such attacks in real-life, it is paramount for us to answer these questions. As outlined earlier, we were unable to find these answers in the current research literature. In this work, we attempt to seek these answers by closely replicating the implementation of the above four inference frameworks and re-evaluating them in realistic adversarial settings. In the next section we give details of the replicated experiments, and in section 4.4 we present our evaluation results obtained using these replicated experiments.

4.3 Experimental Setup

Due to the unavailability of publicly-available code, we replicated the inference frameworks of the four research efforts discussed above, i.e., pencil and paper writing [198], whiteboard writing [19], finger writing [199], and air writing [8,9], as closely as possible based on the information available in the corresponding papers. Below, we provide details of the experiments that we conducted using these replicated implementations of the inference frameworks.

4.3.1 Participants and Data Collection

To account for participant diversity in our experiments (with these frameworks), we recruited 28 participants aged between 18 and 30 ($\sigma = 4$) years, seven participants for each form of writing. 13 out of the 28 participants were male, and remaining 15 participants were female. All 28 participants were recruited using fliers posted around our University campus, and as a result they were from diverse demographic backgrounds. In order to test generalized inference models as an adversary would, and in-line with some of the previous works, only right-handed participants were used for the study (i.e., writing with the right hand). For the same reason, we also enforced non-cursive writing. In order to minimize bias in the collected motion sensor data, no other restrictions were imposed on the participants. Participants were not given any time limit to complete their writing tasks, and were encouraged to write using their normal or accustomed handwriting style. The entire experiment was also approved by our University's institutional review board (IRB).

4.3.2 Writing Scenarios

Participants in our experiments were asked to wear a smartwatch (Sony Smartwatch 3 or LG Watch Urbane, depending on the experimental scenario as detailed later) on their writing hand and perform the assigned writing tasks (as described below). Both accelerometer and gyroscope data were recorded at $200Hz$ from the smartwatch while the participants were undertaking their writing tasks. Depending on the writing scenario and setting, participants were provided with appropriate writing apparatus and environment. For example, in the pencil writing scenario participants were

(a) Pencil & Paper Writing (b) Whiteboard Writing (c) Finger Writing (d) Air Writing

Figure 4.2: Writing scenarios considered in our experiments.

provided with a pencil, a chair to sit on, and positioned near a table with a sheet of paper on top of it (used as the writing surface). In the whiteboard writing scenario, participants were provided with a marker pen to write on a nearby whiteboard mounted on the wall. In the finger writing scenario, participants were provided with a chair to sit on, and positioned near a table with a touchscreen tablet computer on top of it ($8''$ Samsung GT-N5110 Android tablet, used as the writing surface). To match with [199], the writing area on the tablet screen was designed to be $2.5'' \times 2.5''$. Lastly, for the air writing scenario, participants were provided with a chair to sit on, and ample free space around them to allow free movement of their arm. In all scenarios, the alphabets/words/sentence to be written were displayed on a nearby tablet screen, except in case of finger writing where the same tablet computer was used for both displaying the writing task and as the writing surface. The tablet was also used to record the ground truth alphabets/words, and additional ground truth spatial data in the finger writing scenario for in-depth empirical analysis of writing characteristics. Figure 4.2 show the setup of all the four writing scenarios described above.

4.3.3 Writing Tasks

In all of the four writing scenarios outlined above and depicted in fig. 4.2, participants performed the *same* set of writing tasks, where some of the subtasks were randomized in order to minimize bias in the writing activity. The design of our writing tasks was carefully undertaken so as to enable us to perform an equitable comparison of our results with the ones obtained earlier, while at the same time helping us gain more insight on the impact of different writing characteristics, settings, scenarios, etc. on the resulting inference accuracy. The writing subtasks were as

follows:

- **Alphabets.** Participants wrote individual alphabets one at a time, covering all 26 alphabets in random order, in both upper and lower cases. Each alphabet was written 10 times, for a total of 260 alphabets written by each participant.
- **Words.** Participants wrote 4-8 alphabet words, one at a time, selected random from a vocabulary [68]. Each participant wrote 20 words, in both upper and lower cases.
- **Sentence.** Participants wrote a sentence covering all alphabets of the English language, "the five boxing wizards jump quickly", in both upper and lower cases.

In addition to the in-lab writing tasks, we also collected data from 2 participants to evaluate writing activity recognition among other daily activities. The participants wore a smartwatch for a day, and were asked to perform writing scenarios belonging to each of the above scenarios at random times during the day.

4.3.4 Inference Frameworks

We implemented the targeted inference frameworks [9, 19, 198, 199] primarily using Python 3.7, making use of the machine learning library *scikit-learn* wherever applicable. The specific implementation details for each of the writing scenarios are presented below.

Pencil Writing: The raw sensor data was pre-processed by using Pauta Criterion [207] to eliminate outliers in the data, followed by a low pass filter with a range $1Hz$ to $25Hz$. As, our data collection already contains ground truth, no boundary detection was needed when training individual letter recognizer models. Then, for each letter sample in the preprocessed gyroscope data, entropy was calculated for each axis after a Fast Fourier Transform (FFT). For each axis, amplitudes for each frequency from 1 to $25Hz$ were computed. The remaining processing included finding peaks and valleys in the data and then computing features related to first peak and valley, last peak and the maximum peak. The resulting feature vector from this computation included a total of 115 features, which were then used to train a Random Forest classifier. Hyper-parameter

tuning was done by using random search followed by a grid search to generate the best set of hyper-parameters. In the word detection phase, the top-5 letter predictions for each letter of a word were used to generate a list of letter sequences, which were then processed by a spell correcter module based on a comprehensive (100,000 words) dictionary, in order to obtain a list of corrected words. From the list of corrected words, the most frequently occurring word is selected as the final predicted word.

Whiteboard Writing: The pre-processing done by Arduser et al. [19] involves converting the data from device coordinates to whiteboard coordinates. As sufficient information about this coordinate conversion was unavailable, we were unable to accurately replicate it. To compensate for this limitation and still have an equitable analysis, our whiteboard writing participants were asked to maintain a constant height when writing. The individual letters were then directly used in a Dynamic Time Warping (DTW) algorithm for character recognition. DTW has been traditionally used for time series alignment and calculation of a similarity distance between two time series [168]. A part of the collected wrist motion data (of users' handwriting) was used as templates for each letter in the alphabet, and these templates were then aligned with the query test sequences to compute the similarity distance score, where lower score would imply a better match. The results presented by Arduser et al. considered the presence of a written character in the top-3 predictions of the corresponding character as a successful inference. To compare against these results, we used the top-3 predictions for each character from DTW, i.e. the three lowest DTW scores. In the word detection phase, the audio collected alongside the motion data was used to segment letters within a word. A vector was constructed by summing the absolute amplitudes between 5 to $10KHz$ in the time series. Then, consecutive values greater than a threshold in this vector are combined, where the first value above the threshold is marked as the starting point and the possible end points are marked within a 2.5 $second$ window from this start point. The corresponding accelerometer data between start and end point sequences is then used in the letter recognition, and used as constituents for word prediction.

Finger Writing: Although no pre-processing step for raw data is mentioned in [199], certain

Table 4.1: Comparison of alphabet inference accuracies. Empty fields imply that the original work did not test those setting.

	Personalized				Generalized			
	Lowercase		Uppercase		Lowercase		Uppercase	
	Original Work	Our Replication	Original Work	Our Replication	Original Work	Our Replication	Original Work	Our Replication
Pencil Writing (Xia et al.)	–	11%	–	10%	50%	6%	–	5%
Finger Writing (Xu et al.)	–	8%	91%	17%	–	5%	–	7%
Whiteboard Writing (Arduser et al.)	–	51%	94%	56%	–	20%	–	27%
Air Writing (Amma et al.)	–	14%	95%	14%	–	5%	82%	9%

individual features (from the data) were required to be computed after passing the data through a low-pass and band-pass filter. A set of features relating to motion energy, shape, posture were computed for both accelerometer and gyroscope data over all the axes. A detailed description of these features can be found in [130]. This resulted in a feature vector comprising of 46 features for each character window. These feature vectors were then used to train Naive Bayes, Logistic Regression and Decision Tree based classifier models for the character recognition tasks.

Air Writing: In this writing scenario, the raw sensor data was first normalized and then used to extract the average amplitude for each axis (from both the accelerometer and gyroscope data stream) resulting in a feature vector comprising of six features. These features were then used to build and train a separate Hidden Markov Model (HMM) classifier [147] for each letter in the alphabet, resulting in a total of 26 models. These HMMs were implemented with left to right topology and 30 states. A Gaussian Mixture Model with six components was used to obtain observation probabilities for each state of the HMM.

4.4 Inference Accuracy Results

In this section, we present evaluation results from our evaluation experiments with the replicated inference frameworks discussed above. First, we present inference accuracy results that were obtained in a generalized setting, followed by results obtained in a personalized setting. A summary of the inference accuracies obtained in our experiments compared with those obtained in the original papers of the four handwriting inference schemes can be found in Table table 4.1.

4.4.1 Generalized Inference Accuracy

The generalized models were tested by using a Leave-One-Out-Cross-Validation (LOOCV) mechanism, where data from a single participant is used as the test set while the remaining participants' data is used as the training set for the corresponding classification model.

Pen(cil) Writing: The extracted feature sets for each individual alphabets, as described in the previous section, were used in training a Random Forest classifier. Classification using the trained Random Forest classifier in the generalized setting yielded an average accuracy for lowercase alphabets of about 6% ($\sigma = 1\%$) with alphabet *"l"* having the highest accuracy at 19% and all other alphabets had accuracies below 15%. The uppercase alphabets also resulted in similarly poor average accuracy of only 5% ($\sigma = 2.0\%$) with alphabets *"E"* and *"L"* having the highest accuracy at 17% and 16%, respectively, and all other alphabets had accuracies below 10%. In comparison, authors of the original work [198] were able to obtain a mean accuracy of 50% ($\sigma = 17\%$) for the lowercase writing scenario. Our poor alphabet-level accuracies were reflected in word prediction as well ($< 1\%$). In comparison, authors in [198] obtained a word accuracy of about 33%.

Finger Writing: The finger writing inference is done using three classifiers: Decision Tree, Naive Bayes and Logistic Regression. The classification accuracies of all the three classifiers turned out to be poor, with only 5% average accuracy ($\sigma = 1\%$) for lowercase alphabets. Similar accuracies were also observed for the uppercase alphabets with 7% average accuracy ($\sigma = 3\%$) for all the three classifiers. Word prediction was mostly unsuccessful ($< 1\%$ accuracy) due to the low alphabet-level inference accuracy.

Whiteboard Writing: When each participant's dataset was tested against the templates taken from all other participants, an average accuracy of 20% ($\sigma = 4\%$) for lowercase alphabets was observed. Alphabets *"l"* and *"z"* showed the highest accuracies at 55% and 47%, respectively, while all other alphabet accuracies were below 30%. We were able to obtain an average accuracy of around 27% ($\sigma = 8\%$) for uppercase alphabets, where alphabets *A*, *H*, *L*, *M*, *N*, *W*, *Z* showed over 40% accuracy. We obtained an average accuracy of 39% ($\sigma = 6\%$) for lowercase alphabets when top-3 predictions were considered, with alphabet *"o"* having an accuracy of 87%, followed by *"l"*

having 75% accuracy. For uppercase alphabets, an accuracy of 44% ($\sigma = 8\%$) was obtained with *"L"* having the highest accuracy at 72% followed by *M*, *N*, *H*, *W*, *V* having accuracies above 60%. It is unclear whether the results presented by Arduser et. al [19] were obtained in a generalized setting or a personalized one, but they were able to report a very high average inference accuracy of around 94%.

Air Writing: In the generalized setting, our average accuracy for the uppercase alphabets was only 9% ($\sigma = 1\%$). The mean lowercase alphabet accuracy was a mere 5% ($\sigma = 3\%$). The trained HMM-based character models were then concatenated and tested to infer words. The poor individual character inference accuracies were again reflected in word-level inferences, with less than 1% word inference accuracy.

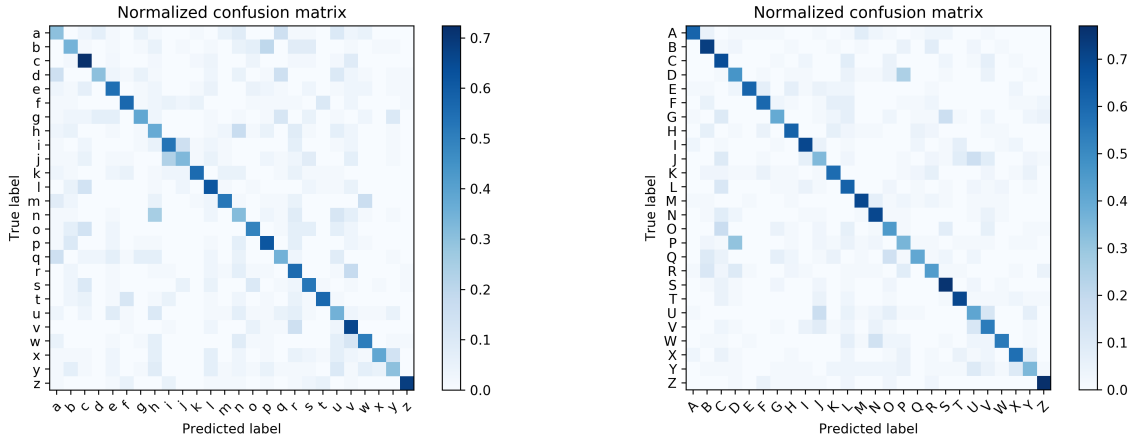
4.4.2 Personalized Inference Accuracy

In the personalized setting, the classification models were evaluated by splitting the (motion) dataset of a participant into a training set and a testing set, and cross-validated wherever applicable.

Pen(cil) Writing: The dataset of each participant was split into training and testing data using a 60:40 ratio. The alphabet inferences were poor even in the personalized setting, with only an 10% average accuracy for uppercase and 11% accuracy for lowercase alphabets. The authors in [198] do not provide any results for a personalized scenario mainly because their scheme was proposed as an attack. The poor alphabet inference accuracy was insufficient for a word inference, even with the help of a dictionary to recognize words.

Whiteboard Writing: The whiteboard writing scenario was analyzed in a personalized setting by using 50% of the alphabet samples per participant as a training set (or set of templates for the DTW algorithm), while testing was performed using the remaining alphabet samples. This resulted in a 51% mean alphabet accuracy ($\sigma = 0.13$) for the lowercase alphabets, with alphabets *c*, *p*, *v*, and *z* having accuracies over 60%. We were able to obtain a 56% mean accuracy for uppercase alphabets ($\sigma = 12\%$), with alphabets *B*, *I*, *M*, *N*, *S*, *Z* having accuracies over 70%. Figure 4.3a shows that alphabets *"n"* was often misclassified with *"h"* (and vice-versa). Similarly, letters *"i"*

and *"j"* were also often misclassified with each other due to the high similarity between their strokes. Figure 4.3b shows that in the uppercase alphabets, *"P"* and *"D"* were often misclassified with each other along with *"U"* and *"V"*. We also tested the prediction accuracy by considering the top-3 guesses. This resulted in 73% accuracy ($\sigma = 10\%$) for uppercase alphabets. The alphabets *C*, *M*, *S*, *Z* had over 80% accuracy. For lowercase alphabets, 69% accuracy ($\sigma = 10\%$) was obtained, and only the alphabets *c*, *f*, *k*, *p*, *z* had accuracies over 75%. In comparison, [19] presented results only for uppercase alphabets and had a 99% accuracy within 3 guesses.



(a) (b)

Figure 4.3: Confusion matrix for whiteboard writing alphabets (a) lowercase, (b) uppercase.

Finger Writing: For this scenario, all the three classifiers were evaluated with a 60:40 train:test ratio. We were able to observe an average accuracy of around 8% for all the three classifiers ($\sigma = 2\%$) for inferring lowercase alphabets. Inference of uppercase alphabets produced a slightly higher average accuracy of around 17% ($\sigma = 10\%$). In comparison, Xu et. al [199] had over 85% accuracy for all the three classifiers for uppercase alphabets. Word prediction was mostly unsuccessful ($< 1\%$ accuracy) in our test due to the low alphabet-level inference accuracies.

Air Writing: For this scenario, the data set for each participant was split using a 65:35 training:testing ratio. We observed an uppercase alphabet inference accuracy of around 14% ($\sigma = 3\%$). Inference of lowercase alphabets also resulted in an average accuracy of around 14% ($\sigma = 5\%$). The individual character HMM models were then concatenated to build word models for every

word in a vocabulary of approximately 1000 words [68]. The word-level data collected per participant was then tested against these word models to predict words. The low individual character accuracies were reflected in the word-level inferences and we obtained an average word-level inference accuracy of less than $< 1\%$.

4.4.3 Writing Activity Detection

Among the four handwriting schemes considered in this work, only [9] (air writing) and [199] (finger writing) evaluated the problem of detecting writing related gestures/activities among other activities. In real-life deployment, this step is equally essential for both HCI applications and for an adversary trying to infer private handwritten data. Xu et al. [199] considered gestures relating to the index finger, the hand and the arm, and classified them using the same set of features used for alphabet inference. They obtained true positive rates of over 90%. Amma et al. [9] used a binary SVM classifier to identify the air writing motion in a continuous motion stream and achieved a recall of 99% and a precision of 26%. As both of these prior works used specialized devices, while we only considered wrist motion data available from the smart watches (used in our experiments), it was challenging to perform an equitable comparative evaluation. We replicated the handwriting activity detection model used by Xu et al. [199], but tweaked it so that it can be used to identify any of the four writing scenarios (i.e., pencil writing, whiteboard writing, finger writing and air writing). In a personalized setting with a user's labeled data included in the training, our activity detection model achieved around 56% recall (and 57% precision) for air and finger writing scenarios while pencil writing achieved 39% recall (47% precision). Whiteboard writing resulted in the lowest recall value at only 23% (34% precision). When considering each writing scenario against all the other writing and non-writing activities, whiteboard and finger writing resulted in over 90% recall with under 40% precision, and air writing and pencil writing resulted in recall of 78%. A generalized testing of our activity detection model achieved around 35-40% recall for air writing, whiteboard writing and pencil writing, whereas finger writing resulted in the lowest recall at just 8%. The whiteboard writing achieved highest precision at 65%, while the other three

writing scenarios had a lower precision in the range of 20-40%.

4.5 Factors Affecting Inference Accuracy

As evident from section 4.4, our replicated experiments did not perform as well as some of the previous works. In this section, we analyze the factors that we believe were the main causes of the poor inference accuracies, which indirectly determines the practicality of such handwriting inference attacks in real-life settings.

4.5.1 Number of Strokes

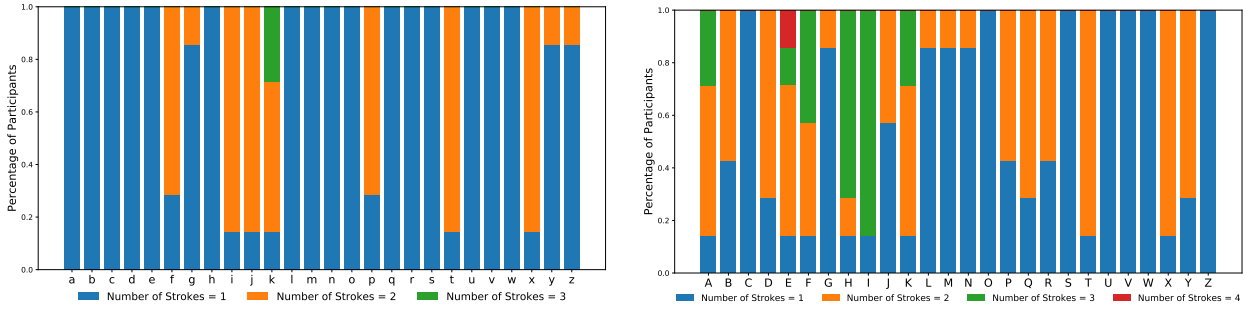
One of the main factors that influences a person's handwriting (and thus, its inference by the described frameworks) is the number of strokes the person uses to write each alphabet of a language. During our experiments, we observed varying writing styles among different participants which effectively resulted in varying number of strokes (across participants) for writing the same alphabet. Figure 4.5a and fig. 4.5b shows that the same alphabet is written using different number of strokes by different participants. For lowercase alphabets, we observed that *a*, *b*, *d*, *e*, *f*, *g*, *h*, *k*, *p*, *q*, *w*, *x*, *y* and *z* have varying number of strokes. For alphabet *k*, we observed some participants used just one stroke and some other participants used up to three strokes. Uppercase writing shows more variation in number of strokes compared to lowercase writing, where except for alphabets *C*, *H*, *O*, *S*, *U* and *W*, all other alphabets show variations. Notably, the alphabet *E* was written using number of strokes ranging from three to four.

We also observed that even the same participant sometimes use varying number of strokes for the same alphabet. Figure 4.4 shows the mean of variances for the number of strokes calculated per participants. It is observed that except for alphabets *c*, *o*, *q* and *s*, all other lowercase alphabets show some variance in the number of strokes used. In uppercase alphabets, only *C*, *L*, *O*, *S*, *U*, *V*, *W* and *X* were consistent when it comes to the number of strokes used. The high variance of *k* is possibly due to the multiple ways that alphabet can be written, in which number of strokes ranging from one to three can be used to write it. Similarly, for alphabet *E*, possible methods of writing

includes using number of strokes ranging from two to four (considering a stroke to be the writing segment from one pen-down to the next pen-up).

(a) Lowercase (b) Uppercase

Figure 4.4: Variance in number of strokes per alphabet per participant, averaged for all participants.



(a) Lowercase (b) Uppercase

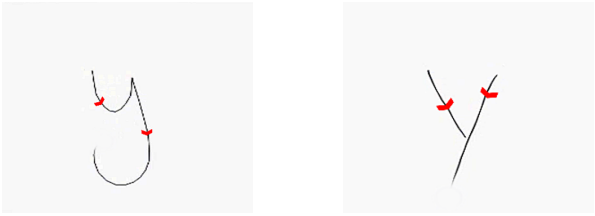
Figure 4.5: Number of strokes for the same letter for different participants.

4.5.2 Order of Strokes

Additionally, we also observed that alphabet letters written using more than one stroke introduces another element of confusion, which is the order in which the strokes are written. A simple example would be writing uppercase alphabet *T*, in which we predominantly observe two strokes. These two strokes can be either written as a horizontal stroke followed by a vertical stroke, or vice versa. Such variations in order of strokes is likely to cause high degree of misclassification during inference.

(a) Writing Style 1 (b) Writing Style 2

Figure 4.6: Different ways of writing alphabet uppercase N.



(a) Writing Style 1 (b) Writing Style 2

Figure 4.7: Different ways of writing alphabet lowercase y.

4.5.3 Direction of Strokes

Another important factor, especially considering that we are using motion signals to infer handwriting, is the direction of the strokes used to write an alphabet. One common way of writing *f* is starting from the curved top and writing the vertical stroke, followed by the horizontal stroke. But, depending on a person's writing habit, one could also write *f* with a vertical stroke from the bottom and curve it on the top. Figure 4.6 shows how two different participants have written the alphabet *N*. One participant started from the bottom of the first vertical stroke and continued the same stroke to end of alphabet. The other participant started the first vertical stroke from the top and then went up through the same stroke to complete the alphabet. This is a clear depiction of how even the direction of strokes differ among various writers. Further, the overall shape of the alphabets also could differ across participants. Figure 4.7 shows two different styles of writing lowercase *y* with a curved style strokes and non-curved strokes. Such variations in direction of strokes is likely to cause high degree of misclassification during inference.

4.5.4 Training Data Relevance

Even the same person can write differently based on the time, location, or some other context. The variations of number of strokes can be assumed as a possible indicator of such context-depended writing characteristics. For example, while in haste a writer may choose to use lesser number of strokes than usual for writing an alphabet letter or write certain alphabets differently than usual, whereas in more leisurely settings, the same writer may be more careful and consistent in his/her writing. Also, the previous alphabet could affect how the next alphabet is started when writing in a natural setting. This specific transition motion that occur between alphabets, could vary mainly based on how the previous alphabet was written, i.e. the number and/or the order of the strokes. And these transition motions intrinsically could affect alphabet classification, because the writing motion of an alphabet preceded or followed by the transition motion could easily be mistaken for a totally different alphabet. To this end, the authors of air writing [9] specified that they used a separate HMM model for such transitions, but do not provide details on how the data to train this model was obtained. And as pointed out above, these factors vary based on the writer and also the writing conditions (time, location, writing surface, pen/pencil used, position, etc.). Therefore, training such a model would be a complicated task due to the highly irregular nature of these transitions.

4.5.5 Uppercase vs Lowercase

Most of handwriting inference or recognition related works in the literature consider only one of the alphabet cases, i.e., lowercase or uppercase, along with claims that their framework can be easily extended to perform inference in the other case. However, we observed that frameworks which rely on feature-based parametric classifiers do not extend well into the other case. This was observed for finger writing and air writing in which the original frameworks only considered uppercase, and our results indicate slightly better accuracy for uppercase alphabets over lowercase alphabets. Authors of pencil writing [198] utilized highly specific features designed to assist in the classification of alphabets normally containing exactly two strokes, such as *f*, *i*, *j*, *p*, *t*, and *x*. As a

result, it is unclear how such features would perform for uppercase alphabets which can have more (or less) than two strokes.

4.5.6 Wearables Not on the Writing Wrist

According to an ongoing online study [4] taken part by more than 5667 participants, only about 39.93% users prefer to wear their watch on their dominant hand. This indicates that a majority of users would wear their (smart)watch or fitness band on their non-dominant hand, or in our context the non-writing hand. Needless to say, the outlined handwriting inference attacks using wrist-wearable motion sensors will fail if the eavesdropping smartwatch or fitness-band is not worn on the writing hand.

4.5.7 Specialized Devices

As mentioned earlier, the original air writing [9] and finger writing [199] works used specialized hardware, such as a glove with a much higher sampling rate (more than $800Hz$) than that supported by consumer-grade smartwatches. Such specialized hardware is advantageous to an attacker since it allows capturing more sensitive and comprehensive information on hand movements. While the requirement of specialized hardware currently limits the scope of handwriting inference attacks, recently researchers have shown that consumer-grade smartwatches may have more potential than previously known. For example, Laput et. al. [97] show that consumer-grade smartwatch OS impose an artificial limit on the sensor sampling rate, which can be bypassed by modifying the OS kernel. With a modified kernel, Laput et. al. [97] were able to record accelerometer data at $4KHz$ in order to detect hand gestures and detect objects grasped by the hand. Therefore, it is possible that in future, sensor sampling rate becomes much higher in smartwatches, which would allow adversaries to capture more sensitive hand motion data capable of more accurate handwriting inference.

4.6 Discussion

4.6.1 Limitations

Our main objective in this work was to investigate whether state-of-the-art wrist motion based handwriting inference techniques do actually work "as advertised" in realistic (uncontrolled and unconstrained) writing settings and scenarios. Our overarching goal was to determine if these schemes pose a significant privacy threat and can be deployed as a feasible adversarial tool to infer sensitive handwritten text. Although we demonstrate that existing wrist motion-based handwriting inference techniques do not perform well in realistic writing scenarios using modern consumer-grade wrist wearable devices and would not be very feasible adversarial tools, our work stops at that point. In this work, we do not make any attempt to propose novel inference frameworks that outperform the existing ones considered in the earlier research efforts. However, the lessons learned from this research effort will definitely be useful in such endeavors in the future.

Despite our best efforts to collect participant handwriting data in a natural and unconstrained setting, we were obviously not able to capture all possible writing situations. Our data collection was still in a conventional writing setting and we did not include/evaluate non-conventional scenarios such as writing too quickly (due to one being in a haste) or writing too slowly. Moreover, our experiments only considered a set of standard and popularly used writing apparatus and surfaces, and we did not evaluate these existing inference mechanisms for a variety of other alternate writing tools (such as stylus, marker, chalk, etc.) and surfaces (such as curved or angled writing surfaces). It should also be noted that while participants were given complete freedom (and recommended) to write in a natural, unconstrained fashion (with the only limitation being non-cursive), we were unable to control environmental factors such as preferred ambient light and temperature which can also potentially impact a person's natural writing ability or style. Furthermore, all the wrist motion based handwriting inference schemes analyzed in this work consider only non-cursive writing scenarios primarily because of the inherent complexity of inferring inter-connected letters within a word in cursive writing. In addition to this, these schemes only collected data from right-handed

writers for consistency reasons. In order to accomplish an equitable comparison of the obtained inference results, we also carried out our data collection experiments only for non-cursive handwriting and for right-handed writers. A more comprehensive analysis in this direction should also include data from left-handed and cursive writers.

4.6.2 Replicability

When trying to replicate the results of previous handwriting inference works, a significant amount of our efforts went in to re-implementing the inference frameworks and re-collecting data in realistic unconstrained writing settings. Our research would have been less demanding if authors of these earlier works would have made their research reproducible. Unfortunately, this is not surprising as replicability has been a significant issue in the security and computer systems community [47, 88]. To make our research effort more useful to the community, we have made all our data and source code publicly available. A web link to these artifacts can be found at the end of this section. Researchers working in the same domain will now be able to comparatively analyze their proposals to the existing ones in the literature. **Artifacts:** https://sprite.utsa.edu/art/dewristified

4.7 Conclusion

In this research work, we investigated how frameworks on wrist-wearable motion sensor based handwriting inference attacks perform in realistic day-to-day writing situations. We carefully analyzed the major factors that bring complexity to wrist motion based handwriting recognition by highlighting specific ambiguities we observed in the order of the strokes, number of strokes, and direction of strokes when writing a character, followed by the overall shape of a character. In addition to these writing characteristics being different among different users, we also observed inconsistencies within the same user's handwriting. Our investigation depicts that due to highly varying nature of handwriting from person to person, wrist motion sensor based inference attacks are unlikely to pose a substantial threat to users of current consume-grade smartwatches and fitness bands.

Acknowledgment

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

CHAPTER 5: SENSORY SIDE-CHANNEL ATTACKS ON WEARABLES: KEYSTROKE INFERENCE USING MULTI-SENSOR DATA FROM HEADPHONES

**The contents of this chapter and the reported experimental results are currently under review at The Network and Distributed System Security (NDSS) Symposium 2025, co-authored by Raveen Wijewickrama, Maryam Abbasihafshejani, Anindya Maiti and Murtuza Jadliwala (in that order). The contents of the manuscript has been reproduced here with revisions.*

5.1 Introduction

A headphone (or earphone) has become a ubiquitous computer peripheral which enables a more private and personal experience for users in a variety of online communication and entertainment applications. Similar to other mobile device hardware, modern headphones also come equipped with a multitude of sensors that play a pivotal role in enhancing the overall user-experience and functionality (of the hardware). Among these sensors onboard modern headphones, two of them stand out: (i) *accelerometers* for their ability to detect motion, and (ii) high-definition *microphones* that enable high fidelity voice sensing and recording.

Most modern headphones incorporate multiple microphones and accelerometer sensors to enable advanced features and applications such as wearing detection [13], gesture controls [39, 57], adaptive noise cancellation [183], spatial audio [122], and fitness tracking [145, 151]. However, integration of advanced sensors on a device such as a headphone also presents potential security and privacy challenges. Due to their proximity to the user and surrounding devices/peripherals (e.g., keyboards and mouse), combined with their multi-sensing capabilities, headphones provide a unique vantage point for surreptitiously capturing sensitive user data such as inputs and keystrokes. Despite the possibility that headphones could present a novel and attractive attack surface for eavesdropping sensitive information (e.g., keystrokes), the feasibility of such threats has not received much, if any, attention in the research literature.

Keystroke inference through acoustic eavesdropping has been investigated before. For instance,

Lie et al. [103] harnessed sound waves produced by typing on a traditional keyboard using a smartphone's microphone. In contrast, Narain et al. [131] and Lu et al.'s KeyListener [112] focused on touchscreen keyboards, with the latter using an adjacent adversarial phone. Similarly, Bai et al. [22] applied stereo audio recording from a mobile phone near a physical keyboard for keystroke inference. These prior works mainly utilized fixed audio recording devices, such as mobile phones or specific computer microphones controlled by an adversary, to gather the acoustic signals generated from keystrokes and then use them for the inference task. In practical situations, it's unlikely that an individual would overlook unknown devices in their vicinity. Though an intruder could consider leveraging the victim's mobile phone (via a malicious app), the unpredictable and user-dependent placement of the phone relative to a keyboard often diminishes the feasibility of such acoustic-only inference attacks.

Past research has also explored the use of motion sensors for keystroke inference, often utilizing mobile phones or wearables such as smartwatches [36, 105, 116, 192]. Marquardt et al. [123] notably employed a smartphone's accelerometer to detect vibrations from nearby keyboard keystrokes. However, such methods are limited by the need for close and/or fixed proximity to the keyboard or require the victim to wear or hold the infiltrated wearable device on their hand, limiting their practicality (as in the earlier case).

Diverging from these traditional keystroke inference attack scenarios, which typically involve unpredictable and non-practical placement of explicit recording devices near the target user, we investigate the feasibility of a more realistic scenario involving a recording device (i.e., a headphone) that is ubiquitously present on the target user in a predictable location. Many modern headphones often come equipped with multiple microphones on both ears [13, 70], which serve dual purposes: capturing audio and facilitating noise cancellation. Such a design makes modern headphones an enticing vector for a keystroke inference attack. Further, prominent Bluetooth-enabled headphone manufacturers [12, 72, 161] provide smartphone/desktop apps to pair with their headphones. We speculate that such apps, which may have access to the headphone sensor data, could behave maliciously and exploit this connection to discreetly record and transmit sensor data to carry out

keystroke inference on unsuspecting users. A probable scenario is where a manufacturer itself is adversarial and uses its native app to record sensor data to infer sensitive information about the users.

Even if an adversary is surreptitiously able to record sensor (audio+motion) data from the target headphone, there are several other non-trivial challenges (in such a keystroke inference attack scenario) which will need to be overcome. One challenge is that the relative position between the headphones and the keyboard can vary considerably based on user traits such as height, arm length, and the distance from the chair to the table. This is compounded by both voluntary and involuntary head movements during typing. Prior research often utilized dual microphones, heavily relying on a technique called Time Difference of Arrival (TDoA) which uses time difference that results when sound travels to the individual microphones, yielding stable results primarily because the data collection device was either on the same surface as the keyboard or was the input device itself (e.g., mobile phone qwerty keypad keystroke inference) [205]. Further, these prior works on acoustic keystroke inference primarily relied on recording devices that were stationary. In our setting, due to the motion of the headphone induced by users' head movements, the recording devices can no longer be considered to be stationary and may record much more noisier/inconsistent data.

In light of the above challenges, in this paper we propose *OverHear*, a novel framework to accurately infer keystrokes using acoustic and motion sensor data collected from headphones. While the accelerometer data alone lacks the granularity to distinguish individual keystrokes, it proves to be very useful in clustering keys corresponding to each hand. Traditional acoustic-based sound source localization techniques may not work well for this due to the potentially unsteady movements of the victims. From the acoustic data, we extract Mel-Frequency Cepstral Coefficients (MFCC) features, and train and test different machine learning models to predict individual keystrokes. This prediction is then further refined using a dictionary-based spell-correction approach to improve the overall inference success of the typed word. We evaluated our framework with the help of 17 participants by using a custom headphone hardware setup and with 5 participants using commercially available Apple AirPods. Our comprehensive experimental evaluation using data collected from

these participants shows that *OverHear* is able to attain a top-5 key prediction accuracy (i.e., the correct key is among the top five predicted keys) of approximately 80% for mechanical keyboards and about 60% for membrane keyboards. Furthermore, our framework demonstrated a top-50 word prediction (i.e., the correct word is among the top fifty predicted words) accuracy nearing 50%, and surpassed 70% in top-100 word prediction accuracy across all keyboard types.

Our main contributions can be summarized as follows:

- **Development of a new keystroke inference framework** which employs data from headphone microphones and accelerometers to accurately infer user keystrokes. The framework also includes a keyboard type identification module to identify the keyboard type used by the victim (e.g., mechanical or membrane).
- **Design of an enhanced word prediction mechanism** based on spell-correction to further improve the efficacy and prediction performance of the proposed framework in the presence of missing or extraneous keystrokes.
- **Comprehensive empirical evaluation of the proposed keystroke inference framework** using data from human participants under realistic/unconstrained settings, spanning across various environmental/ambient noise scenarios.

5.2 Related Work and Motivation

Analysis of acoustic signals for the purpose of inferring user input on a variety of mobile and computing devices has been the subject of several studies in recent years, each employing unique methodologies and achieving varying degrees of accuracy. Asonov and Agarwal [20] in 2004 employed acoustic data collected using a dedicated PC microphone with a trained neural network that uses frequency domain features to show that individual key presses can be recognized with an accuracy of 79%. They demonstrated that their inference framework works across multiple different computer keyboards as well as telephone and ATM key pads. In a similar line of research, Liu et al. [103] proposed a novel approach for inferring keystrokes on a mobile device by using the resulting audio signals. They developed a system that utilizes the built-in stereo microphones of

a smartphone positioned adjacent to keyboard to record sound waves produced by a user's typing. By means of a purely non-ML based approach such as Time-Difference-of-Arrival (TDoA), they were able to identify unique patterns in these recorded acoustic waves and match them to specific keys (being typed). Their framework was able to produce key inference accuracies close to 85%. While both the above two research efforts are significant, they were tested under rather restrictive conditions and focused solely on individual key presses rather than more complex scenarios involving word or sentence typing and prediction.

Similar to Asonov and Agarwal [20], Zhuang et al. used a Hidden Markov Model (HMM) to infer keystrokes from audio signals recorded by a PC microphone, and then employed a language model to facilitate word prediction. They were able to achieve a prediction accuracy of 88% and 96% for word and keys, respectively. Narain et al. [131] proposed a framework to infer keystrokes on a touchscreen QWERTY keyboard of a mobile device using acoustic signals. They employed a Decision Tree based learning algorithm, achieving a fairly high accuracy of close to 95% in a single attempt. The most notable aspect of this study was its language-agnostic approach, suggesting that regardless of the language or the content being typed, the methodology can accurately infer the keystrokes.

Similar to Narain et al.'s approach [131], Lu et al.'s KeyListener [112] attempts to infer keystrokes on a smartphone touch screen QWERTY keyboard, but with a nearby adversarial smartphone's microphone. They used a TDoA based approach, achieving a top-1 word accuracy of around 50% and a top-10 accuracy of around 90%. Zhu et al. [210] also proposed keystroke inference framework based on TDoA and achieved a key prediction accuracy of 72%. However, their framework requires at least two malicious mobile phones to be in physical proximity of the victim keyboard, which makes executing the attack practically challenging. Lastly, Bai et al. [22], drawing parallels with Lie et al. [103] and Zhu et al. [210], employed stereo audio recordings from a mobile phone situated near a keyboard to deduce keystrokes. Their methodology integrated TDoA and Power Spectral Density (PSD) features within a support vector machine (SVM) model and achieved a top-5 accuracy of 92%. Cecconello et al. [38] proposed an attack that uses audio data collected

during VoIP application call to infer keystrokes, which is perhaps the closest work to ours in terms the attack surface/hardware setup. They use microphone data from a laptop, a smartphone and a traditional headset (with a boom microphone; a single microphone protruding near towards the mouth) to evaluate their attack. They highlight an attack scenario where the attacker obtains the victim's labeled training data through social engineering to achieve a top-5 accuracy of 80%. However, this accuracy drops below 55% when no training data from the victim is available indicating potential limitation of their framework due to the reliance of victim's labeled training data.

Several past studies have also explored the use of motion sensors for keystroke inference [36, 105, 116, 192]. These investigations typically harness motion sensors in mobile phones to deduce in-device keystrokes or employ smart wearables, such as smartwatches, to infer keystrokes on physical keyboards. Specifically, Cai and Chen in 2012 highlighted the feasibility of motion-based keystroke inference attacks, demonstrating that motion data could be exploited to deduce user keystrokes on touchscreen keyboards of mobile phones [36]. Liu et al. and Maiti et al. proposed approaches for keystroke inference attacks targeting external QWERTY keyboards using motion data from collected from victim's smartwatch achieving word level accuracies over 60% [105,116]. Marquardt et al. [123] introduced (sp)iPhone, where accelerometer data from a smartphone placed near a physical keyboard could be used to infer keystrokes. For their attack, they capitalized on the vibrations generated by the keystrokes on the keyboard and transmitted through the table to the smartphone's accelerometer.

Our extensive literature review has shown key research gaps, which motivates us to explore the feasibility of a new attack surface for carrying out keystroke inference attacks (on external keyboards). First, previous research which employed acoustic and/or motion/vibration signals for such attacks primarily relied on stationary recording devices, such as attacker-controlled mobile phones or dedicated PC microphones, to capture the sound and/or motion/vibration signals produced by the keystrokes or they relied on social engineering to obtain labeled victim training data [38]. In real-world scenarios, it's unlikely that a victim wouldn't notice unfamiliar (recording) devices nearby which makes actually carrying out such attacks challenging. Additionally, relying

on VoIP calls to obtain labeled data [38], presents practical difficulties. The attacker must depend on the victim unwittingly providing text samples during the VoIP call, which may not always occur naturally and can introduce limitations in obtaining comprehensive labeled data, especially for all the keys/characters. While an attacker might attempt to use the victim's own mobile phone (by installing a Trojan app) for such purposes, the unpredictable positioning of the phone near a keyboard makes this approach less practical. Further, headphones, especially the wireless type, are ubiquitous and often worn continuously (due to their extra portability) by users in the same position on the head, even when not in active use (e.g., for noise cancelling purposes). This constant presence and predictable position over the head provides a practically feasible eavesdropping opportunity, which has not been comprehensively evaluated before. Despite the promise of this approach, there are unique challenges to overcome. The variability introduced by human behavior, such as head movements, and individual anatomical differences can affect the consistency of the acoustic data captured by the headphones. Furthermore, the weaker signal of vibrations reaching the motion sensors in headphones, as opposed to those in closer contact with the typing surface, requires a more refined approach. To address these issues, we hypothesize that a combination of acoustic and motion sensor data, both integral to modern headphones, can significantly improve the inference accuracy. We seek to validate this hypothesis by developing a multi-sensor keystroke inference framework.

5.3 Background and Preliminaries

We now provide a brief overview of technical concepts that are critical in the design of our proposed keystroke inference framework.

5.3.1 Microphones in Headphones

Microphones capture sound waves (or acoustic energy) and convert them into electrical signals through the vibrations of a diaphragm which can then be recorded and utilized in various audio applications. There are two main types of microphones based on their conversion mechanisms:

dynamic (more robust) and condenser (high sensitivity). The two most common polar patterns (i.e., the directions from which a microphone can pick up sound) used in microphones are omnidirectional and cardioid. In an omnidirectional microphone, sound is picked up equally from all directions. Conversely, a cardioid microphone is unidirectional, which means it picks up sound primarily from one direction (usually the front) and rejects it from other directions. In modern headsets and earbuds where the microphones are built into the earpiece itself (compared to the ones where a microphone arm protrudes from the earpiece to near the mouth), omnidirectional microphones are typically used. However, one drawback of using omnidirectional microphones is the potential for more background noise. To mitigate this, many modern high-end headsets use noise-cancellation techniques. These often involve multiple microphones within the headphones—one set for user input and another dedicated to noise cancellation—effectively isolating the user's voice and reducing ambient noise [102, 162].

5.3.2 Motion Sensors in Headphones

Headphones are increasingly becoming sophisticated interactive devices with integrated motion sensors that enhance user-experience. These motion sensors, typically accelerometers and gyroscopes, facilitate a range of intelligent features by accurately capturing the device's orientation and movement in a three-dimensional space. Recent headphone models, such as the Apple AirPods and Google Pixel Buds [13, 70] have taken this integration to new heights. They employ accelerometers to enable functionalities such as automatic engagement of microphones for phone calls and voice assistant access, and they can detect when the headphones are in the user's ears, playing sound immediately upon insertion. Users can also interact with their device through gesture controls, such as double-tapping to play or skip tracks, which the headphones recognize through these sensors. The on-device motion sensor suite typically comprises of both accelerometers and gyroscopes, and together these sensors can capture holistic movement of the device in a three-dimensional space along three principal axes (i.e., x, y, and z axes). While accelerometers capture linear acceleration, encompassing actions such as tilting or straightforward motion, gyro-

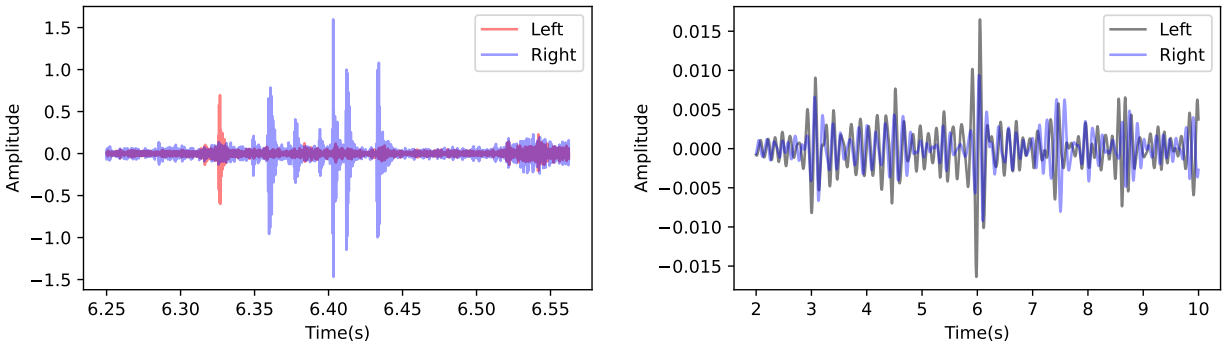
scopes are designed to measure angular velocity capturing rotational movements around the three principal axes [200].

5.3.3 Multi-Sensor Keystroke Inference

Our particular interest is in the potential application of the above sensors beyond their intended use, that is specifically for the keystroke inference task. Keyboards possess distinct mechanical characteristics, resulting in the emission of unique acoustic signatures when keys are pressed and released. As also observed in earlier research efforts [20,22], these audible vibrations are detectable by microphones integrated in nearby devices such as headphones. Figure 5.1a shows an example of a typing related audible vibrations captured by a headphone microphone. Furthermore, as a user engages with a keyboard, the act of typing creates additional non-acoustic vibrations. These vibrations, originating from keystrokes, could travel through the fingers, palms, and further along the arms of a user to their head/headphones. Figure 5.1b shows the accelerometer feedback recorded from a pair of prototype headphones (fig. 5.6a) equipped with accelerometers on each ear piece during a typing task. The noticeable peaks correspond to the key presses, indicating the potential feasibility of employing accelerometer data from headphones for keystroke inference. Although the strength of this signal diminishes with distance, the sensitivity of motion sensors in headphones might be adept enough to pick up certain subtle motions which could facilitate keystroke inference when combined with audio data. As a result of the integration of these multiple sensor data streams from the headphones, our intuition suggests that any potential keystroke detection and inference mechanisms which incorporate these multiple data sources may be able to overcome the limitations and biases inherent in individual sensors, thereby enhancing the identification/prediction of keystrokes.

5.4 Overview of our Approach

In this section, we describe the assumed adversary model followed by an overview of our keystroke inference approach.



(a) (b)

Figure 5.1: Typing captured by a pair of headphones' sensors, (a) audio, (b) accelerometer.

5.4.1 Adversary Model

The primary objective of an adversary in our attack is to infer sensitive information typed by the target user on a physical keyboard, such as passwords, credit card numbers, and personal identification details. We assume that the adversary has the ability to access and acquire both microphone (audio) and accelerometer (motion) data from the target user's headphones. This can be done via either a Trojan application installed on the paired device such as a desktop computer/laptop or a mobile phone which will have API access to the headphones. The availability of APIs such as Apple's `CMHeadphoneMotionManager` and `AVAudioRecorder` greatly simplifies the development of such applications [16, 17]. Such an application typically has benign (some useful) functionality so as not to arouse the suspicion of the target user, but surreptitiously samples the sensor (motion + audio) data and covertly sends it back to the attacker/adversary. While it is true that on a mobile device, an app that requests permission to access the microphone would automatically raise suspicion, especially since leading mobile operating systems consider this as a dangerous permission [71], the application can mask its true intent by providing a useful functionality that require the headphones/microphones without which the application may not work. Examples of such apps include voice recording apps, headphone equalizer apps, and virtual assistant apps, which naturally require microphone access and can justify the permission request which are provided by the headphone manufacturers themselves. Additionally, there may be communication back channels available to the manufacturers in which these apps might not require explicit permissions but can

still communicate with the headphones via Bluetooth. This communication can potentially encode and transmit audio and motion data, effectively bypassing OS-level permission restrictions. An attack setup such as this, which includes a covert sensor data collection app, is feasible in practice and can be realized, for instance, by the headphone device manufacturer themselves (acting as the adversary). There are numerous real world instances [2,27,149,182,184] where such an adversarial scenario has unfolded in real life. In one such instance, a popular headphone manufacturer was a part of a lawsuit which alleged them of using their headphones and the companion app to collect user information without the users' permission [2]. Furthermore, the adversary model can extend to emerging and cutting-edge technology such as standalone headphones (e.g., STREAMZ Voice Controlled Headphones [177]). These are self-contained devices that do not rely on a paired device for functionality and can run applications and collect sensor data independently. An adversary can exploit these by following a similar scenario as above by using an app which masks its true intentions, to covertly capture and transmit audio and motion data. This approach is particularly stealthy as it does not depend on compromising an external device.

We assume that the surreptitiously sampled sensor data is then (covertly) transferred to an adversary-controlled remote server, where the victim/target data is further processed by pre-trained inference models to predict the target users' keystrokes. We also assume that the adversary is able to employ an advanced keyboard identification technique, such as the one designed by us in section 5.5.5, to gain knowledge of the type of keyboard (i.e., a mechanical keyboard, membrane keyboard (external) or a laptop membrane keyboard) the target user is using. The adversary does not have any other medium of inferring the private text typed by the target user, and must rely entirely on the microphone and accelerometer data streams originating from the target victim's headphones.

5.4.2 Inference Framework

Figure 5.2 illustrates the architecture of our proposed *OverHear* inference framework . As discussed in the earlier section, a custom app (pretending to be the malicious companion app associ-

ated with the target's headphone) is first used to (covertly) capture data from the target headphone's built-in accelerometers and stereo microphones during (the target's) typing activities. The raw accelerometer and audio data are then transmitted to a remote server, which houses the remaining components of our inference framework. This server processes the data, constructs trained inference models using labeled data, and validates these trained models with the validation portion of the dataset. Subsequently, the framework is applied to analyze the victim's (unlabeled) data, enabling us to evaluate the model's effectiveness in a real-world scenario.

Recording Victim Data. Our attack framework starts by first covertly recording the victim's sensor data (audio + motion) containing typing activity through their headphones. This data, once captured, is transmitted to a remote server. Next, the type of the keyboard used by the target user is identified by employing a custom-designed keyboard identification technique (as outlined in section 5.5.5) using the raw audio and accelerometer data.

Training Dataset Assembly. Upon identifying the keyboard type from the victim's data, we then simulate a similar typing activity input environment by using the same type of keyboard as the victim. A comprehensive training dataset is curated using acoustic and motion sensor data captured using headphones. This involves data collection sessions with a select group of participants, ensuring a diverse and representative sample that captures various typing patterns and styles.

Feature Extraction and Model Training. With the training data in place, the system extracts a set of features for each keystroke which encapsulates the unique characteristics of keystrokes.

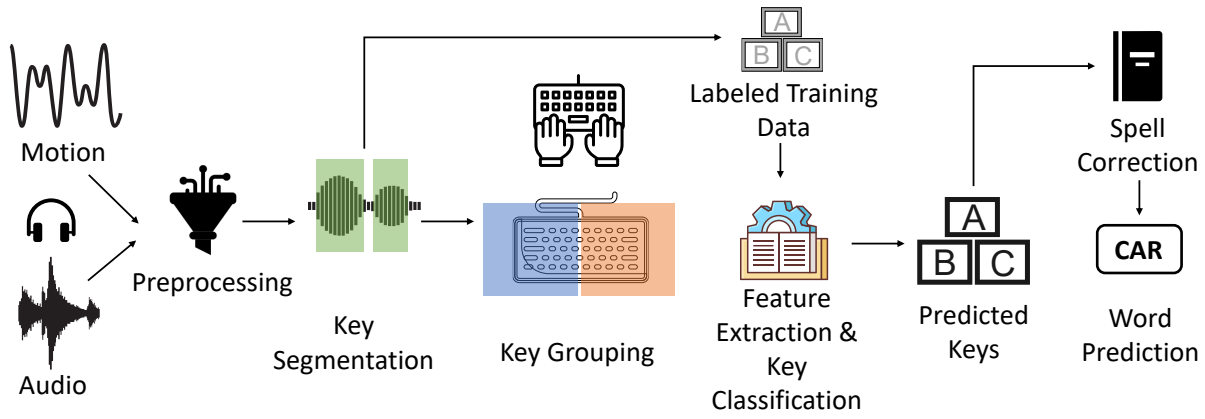


Figure 5.2: *OverHear* inference framework overview.

The keys are then clustered into three groups based on their potential typing hand; left, right and ambiguous (see section 5.5.3 for more details) and then used to train a machine learning model, optimizing it for accuracy and generalization across different users/victims.

Prediction on Victim Data. Using the trained inference model from the previous step, the framework now processes unlabeled data from the victim to infer their keystrokes. The audio and motion sensor data streams captured through victim's headphones go through pre-processing and segmentation to identify keystrokes (as detailed in section 5.5). The unlabeled keystrokes are then tested via the trained machine learning models to predict keystrokes. The sequences of keystrokes are then further processed in a word prediction module with the aid of a spell correcting algorithm to predict the closest matching words.

5.5 Design and Implementation

In this section, we outline the design of our *OverHear* framework and discuss the specifics of its implementation.

5.5.1 Data Pre-processing

Audio Noise Filtering. To ensure the clarity and relevance of our audio data in the context of keystrokes, we employed filtering to mitigate background noise, which typically occurs at higher frequencies distinct from those of keystrokes. Through analysis of the audio captured via our headphone setup, we discerned that keystrokes predominantly occur within the frequency range of 1200 - 3800 *Hz*. Consequently, a bandpass filter was tailored to retain information within this specific range while filtering out extraneous frequencies.

Accelerometer Noise Filtering. The raw accelerometer data very often contains noise, primarily stemming from involuntary body movements, especially in body-worn devices. Specifically, in our headphone based attack scenario, this can be attributed to minor head movements as empirically observed during our controlled experiments. To address this noise, we employed a low-pass filter designed to eliminate high-frequency noise while preserving the lower-frequency vibrations

induced by key presses.

5.5.2 Keystroke Segmentation

In this subsection, we detail the process by which the audio signal captured by the headphones' stereo microphones is segmented into individual keystrokes. This segmentation is critical for distinguishing the discrete acoustic events that correspond to the key hit and key release actions during typing. A keystroke consists of two main events: a key hit event and a key release event. An acoustic signal resulting from a keystroke is able to capture both these events by producing two distinct peaks in the signal, each corresponding to the key hit and key release event. fig. 5.3 illustrates the typical acoustic feedback captured from the stereo microphones on a pair of headphones from a keyboard key press event. The initial more pronounced peak represents the key hit event, while the subsequent, lower peak indicates the key release event.

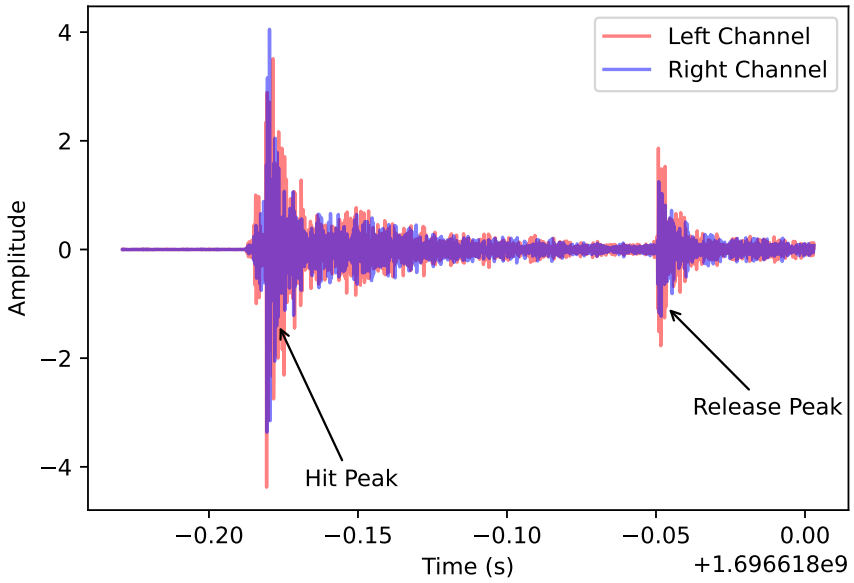


Figure 5.3: Acoustic waveform of a keystroke.

During our experiments, we observed that the average duration of a keystroke is about 80*ms*. We use a sliding window with a size of 1*ms* (empirically determined) and calculate the energy (eq. (5.1)), which is the sum of the squares of the audio amplitude values for each window and is

given by the following formula:

$$E_x = \sum_n |x(n)|^2 \tag{5.1}$$

where E is the energy of the signal, n is the index of the audio sample and $x(n)^2$ is the square of the amplitude of the signal at the n^{th} sample. Energy is a measure of the signal's strength and is particularly useful in identifying the stronger fluctuations caused by the keystroke events. We then pass each window through an adaptive thresholding peak picking algorithm called *Music Structure Analysis Framework (MSAF)* [137] which considers a local average to pick the prominent keystroke related peaks and to discard insignificant noise peaks. The windows with the detected peaks are then considered to be potential keystroke start points, p_s. The consecutive start points less than 80*ms* are discarded (as they may belong to the same keystroke). For each start point, we extract the keystroke as follows:

$$ks_i = (p_{si} - 5\text{ms}, p_{si} + 80\text{ms}) \tag{5.2}$$

where, ks_i is the i^{th} keystroke in the continuous signal, p_{si} is the start point of the keystroke, and $p_{si} + 80ms$ is the end point.

5.5.3 Key Group Clustering

One of the main challenges towards executing a successful keystroke inference attack is the number of potential keys and their arrangement on a keyboard, which can make the prediction task complex. This is because training a single model to distinguish between each key individually may require a vast amount of data for each key to achieve reasonable accuracy. By grouping keys and training for a group rather than individual keys, the dimensionality of the problem can be reduced, thus making the training and inference processes more efficient. Grouping keys can improve the robustness of the predictive model by reducing the potential for confusion between similar sounding keys/keystrokes. In all cases, keys that are physically close to each other on the keyboard produce acoustically similar signals due to their proximity and the mechanics of the

keyboard. By clustering keys into groups, we enable our models to become more sensitive to subtle differences within these clusters. This approach simplifies the overall task by reducing the number of direct comparisons a model must make, enhancing its ability to accurately discern keystrokes. We first look into methods used for key grouping in previous acoustic based keystroke inference works [103, 131] for their applicability in our attack scenario, identify challenges posed by them, followed by proposing techniques that suits our headphone based inference setting.

Traditional TDoA Based Clustering

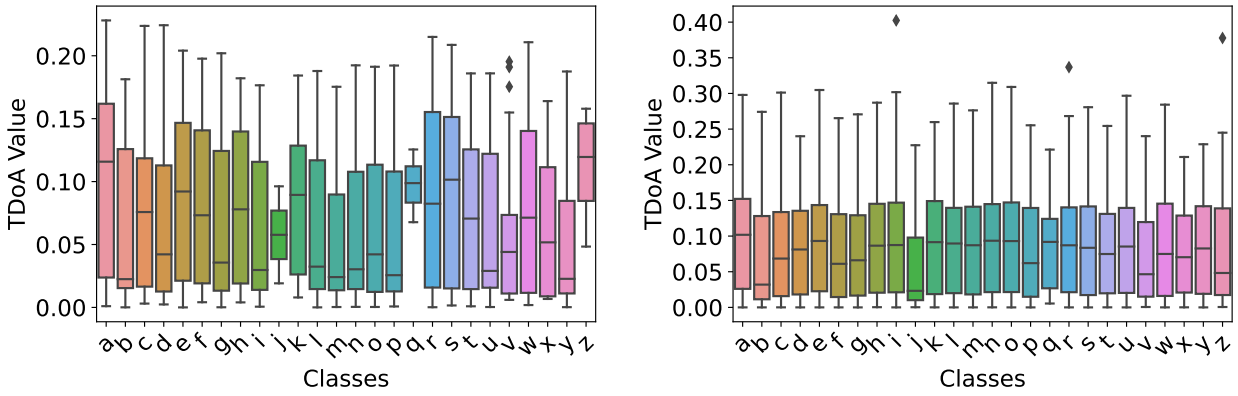
We first look into the possibility of using traditional Time Difference of Arrival (TDoA) based key identification/key group clustering methods to identify similar keystrokes with similar sound profiles [40]. Time Difference of Arrival (TDoA) is a popular technique often used in acoustic or radio signals to estimate the location of an object by using two or more microphones. The overarching principle behind TDoA is the sound produced by a source travels through a medium (often air) and reaches one microphone before the other, if the sound source isn't equidistant to the two microphones. This difference in time that the sound signal takes to reach each microphone is appositely called the Time Difference of Arrival. We compute TDoA via the cross-correlation method for the 2-channel audio signal. TDoA estimation using cross-correlation involves finding the lag (or shift) at which two signals are most similar [22]. The cross-correlation TDoA formula is as follows:

$$\text{TDoA} = \arg\max_{k} \left(\sum_{n} S_1[n] \cdot S_2[n+k] \right) \tag{5.3}$$

where $S_1[n]$ is the signal at the first microphone at discrete time n, $S_2[n+k]$ is the signal at the second microphone shifted by k samples and $\arg\max_k$ indicates the shift k at which the cross-correlation is maximized (the shift where the two signals look the most similar).

However, the dynamic nature of head movements during typing tasks presented a significant challenge in maintaining consistent/reliable TDoA calculations for individual keys. Typists frequently shift their gaze, alternating between various sections of the screen and sometimes the keyboard. This constant change in head orientation rendered TDoA an unreliable metric for

uniquely determining key positions. This is evident not only for individual users/typists, as shown in fig. 5.4a, but also when considering multiple users, as depicted in fig. 5.4b. The extensive spread of data points across all keys or classes further corroborates this observation. Moreover, our attempts to identify distinct key groups/clusters based on similar TDoA values, as was done in some previous works [40, 103], proved to be unsuccessful. The high variability and inconsistency in TDoA values, exacerbated by the previously mentioned anatomical differences and head movements of the users, rendered the task of grouping keys with similar acoustic characteristics nearly impossible. This further highlights the unique challenges posed by our headphone-based setup compared to previous keystroke inference methodologies [22, 103, 131].



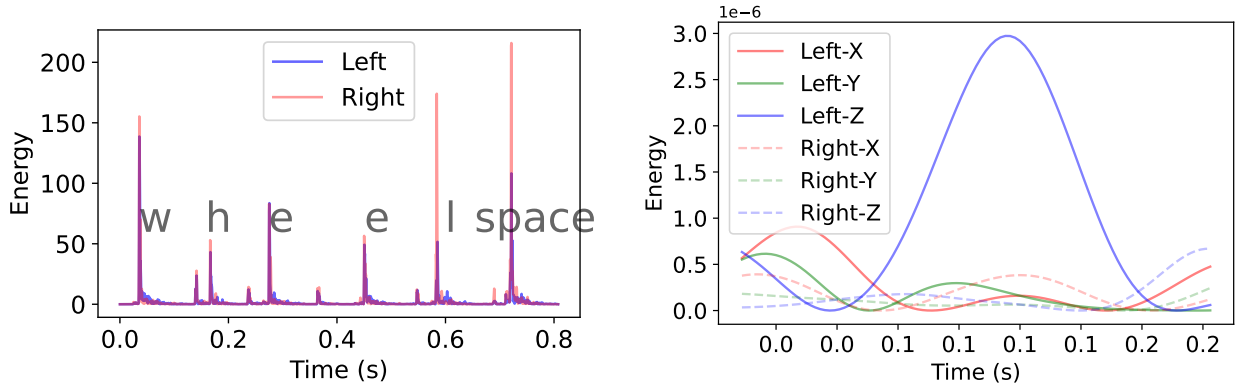
(a) (b)

Figure 5.4: Variability of TDoA values (a) for a single participant, (b) across all participants.

Energy Based Clustering

We next explored other techniques that could aid our framework in identifying key groups. Specifically, we explored the energy levels of keystrokes in the acoustic signal. As we can observe in fig. 5.5a, the energy for the keystrokes towards the right of the keyboard is higher on the right audio channel compared to left channel, and vice versa. Under a setting similar to previous works [22, 103], where the audio recording happens from a fixed position such as a phone kept nearby the keyboard, this type of energy based clustering can easily be used to cluster the keys into left and right groups. However, due to the constantly varying head direction changes that happen

during typing tasks, which may include either looking at different parts of the screen or looking at the keyboard and then looking back at the screen, the energy differences for left and right audio channels also turned out to be not reliable and consistent.



(a) (b)

Figure 5.5: Difference between energy levels of left and right (a) audio channels for keystrokes when typing the word "wheel", (b) accelerometer channels when pressing key 'a'.

Although, the audio channels are affected by the head movements and deemed unreliable for the use of key clustering, we discovered an alternative in the form of accelerometers embedded on both sides of the headset. These accelerometers showed potential in estimating whether a keystroke was made by the right or left hand, based on the motion feedback they recorded. Specifically, the upward and downward head movements, which often occur during typing (for instance, when participants glance at the keyboard and then refocus on the screen), is predominantly observed on the x-axis of the headphone accelerometers. The side-way head movements that may occur during typing are noticeable on the y-axis. The z-axis was observed to be the most stable axis to capture the key press vibrations belonging to left or right hand. fig. 5.5b illustrates the accelerometer feedback captured from our prototype headphone (described in section 5.5.7) when pressing the key 'a' with the left hand. The z-axis energy profile of the left channel is clearly distinct from all other axes.

With the aforementioned observation on the significance of energy readings from the z-axis of the accelerometer, we proceed to quantify the distribution of energy between the left and right accelerometer channels. We achieve this by introducing an energy ratio metric. For each channel,

the energy E is computed as the sum of the squares of its samples (see eq. (5.1)). The energy ratio E_R is then defined as the proportion of the left channel's energy to the total energy of both channels, formulated as:

$$E_R = \frac{E_{\text{left}}}{E_{\text{left}} + E_{\text{right}} + \epsilon} \tag{5.4}$$

where ϵ is a minuscule constant introduced to prevent division by zero. This metric provides a relative energy measurement between the two channels. If a key is pressed from the left-hand, the energy registered on the left accelerometer channel (of the headphones) will be higher than the right accelerometer, making the E_R closer to 1 and if it's a right-hand pressed key, E_R will be closer to 0. The energy ratio inherently normalizes the values between 0 and 1, making it easier to adapt the metric across different users. In contrast, the absolute energy values may vary based on factors such as the distance from the source of vibration/key press (due to anatomical differences), or even the user's typing intensity.

Given the energy ratio calculated for each key press, we establish three key groups based on their spatial positioning on the keyboard and the hand predominantly used to press them.

- $\boldsymbol{G_1}$: Keys predominantly pressed by the left hand, namely {a, s, d, z, x, q, w}.
- $\boldsymbol{G_2}$: Keys predominantly pressed by the right hand, namely {o, p, k, l, n, m, i, j}.
- $\boldsymbol{G_3}$: Ambiguous keys that could be pressed by either hand, namely {r, t, y, u, f, g, h, v, b, c, e}.

We observed that keys within groups G_1 and G_2 are predominantly pressed by the left and right hands, respectively, across different users, especially due to their spatial positioning on the keyboard (extreme left and extreme right). However, the keys in group G_3 presented ambiguity, with the choice of hand varying from one user to another. Such variations could arise from individual typing habits or a user's inclination to favor their dominant hand. During the testing phase on unseen data, the median energy ratio, E_R^{med}, is computed for all samples for a given test user/victim. The rationale behind computing the median energy ratio is to account for the vari-

ability in key pressing intensities among different participants. Different participants may exert different pressures when pressing keys, leading to variations in the vibrational feedback recorded by the accelerometers. By using the median, we aim to normalize this variability and achieve an adaptive clustering mechanism. Keys with an energy ratio greater than E_R^{med} are classified into G_1, while those with an energy ratio less than E_R^{med} are classified into G_2. Keys with an energy ratio close to E_R^{med}, within a threshold γ, are classified as G_3.

Later in section 5.5.4 we train three different classification models, one for each group, to predict exact keys within each group. During this prediction, if a test keystroke is initially classified into G_1 but the prediction probability is below a certain threshold λ, the test keystroke is then also evaluated by the models for G_2 and G_3. The final prediction is chosen from the model that yields the highest prediction probability (see algorithm 5.1).

Algorithm 5.1 Energy Ratio-Based Key Group Classification.

Require: E_R, E_R^{med}, λ
if $E_R > E_R^{med}$ **then**
  if Prediction Probability of $G_1 < \lambda$ **then**
    Evaluate using G_2, G_3
    MaxProbability(G_1,G_2,G_3)
  else
    classify using G_1
else if $E_R < E_R^{med}$ **then**
  if Prediction Probability of $G_2 < \lambda$ **then**
    Evaluate using G_1, G_3
    MaxProbability(G_1,G_2,G_3)
  else
    classify using G_2
else
  if Prediction Probability of $G_3 < \lambda$ **then**
    Evaluate using G_1, G_2
    MaxProbability(G_1,G_2,G_3)
  else
    classify using G_3

5.5.4 Feature Extraction and Model Training

After the key clustering step, we then investigate acoustic based features which could be used to identify individual keys. One such set of features include the *Mel Frequency Cepstral Coeffi-*

cients (MFCC), which have been widely used in the field of speech and audio signal processing, particularly for applications such as speech recognition and speaker identification [6]. However, more recently MFCC based features have been used in other acoustic related applications such as keystroke recognition and acoustic activity recognition [103, 154]. The process broadly involves the following steps: (i) First the acoustic signal is divided into fixed sized frames, and for each frame *Fast Fourier Transform (FFT)* is applied to to the acoustic signal to calculate the power spectrum. (ii) Then, *Mel Filter Bank* is applied on the power spectrum computed for each frame. The Mel Filter bank is a set of 20-40 (usually) triangular filters that are spaced according to the Mel scale, which approximates the human ear's response more closely than the linearly-spaced frequency bands. This process converts the frequency power spectrum into Mel spectrum [103]. For our inference framework, we empirically determined the optimal number of Mel Frequency Cepstral Coefficients (MFCC), extracting 14 coefficients for each audio channel (left and right). To capture the variability and characteristics of these coefficients, we computed several statistical measures: mean, standard deviation, skewness, maximum value, median, and minimum value. This resulted in $14 \times 6 = 84$ features for each channel. Thus, by combining both channels, we derived a total of 168 features. In addition to the MFCC features, we also included the *Root Mean Square Energy (RMSE)* [194] of each keystroke per channel, bringing the total feature count to 170. Building on this, we tested several models including *Random Forest classifier*, *Decision Tree Classifier* and a *Deep Neural Network* for our analysis. To optimize its performance, we utilized a Grid Search Cross-Validation approach for hyperparameter tuning using a hold-out dataset. The *Random Forest Classifier* was configured with balanced class weights to counteract the uneven distribution of class frequencies, enhancing fairness and accuracy in the model's predictions. It was set with a maximum depth of 20, and used 'sqrt' for the number of features considered when looking for the best split. The *Decision Tree Classifier* employed a gini impurity criterion with balanced class weights for splits, and a moderate tree depth of 10 to prevent overfitting, alongside tuned parameters for minimum samples per leaf and split to ensure robust nodes. For the *Deep Neural Network*, a sequential architecture was implemented beginning with an input layer of 256

neurons, followed by a 50% dropout rate to mitigate overfitting. The output layer uses a softmax activation function to provide a probability distribution across the classes, also addressing class imbalance by weighting the loss function accordingly. This model was compiled with an Adam optimizer and categorical cross-entropy loss function, which is particularly suited for multi-class key classification where class imbalance is a concern. As detailed in section 5.5.3, we utilize three distinct multi-class classification models corresponding to the key groups G_1, G_2, and G_3. Each classifier is specifically trained to identify keystrokes from within its designated group. The training and testing were executed in a Leave-One-Out Cross-Validation (LOOCV) manner, ensuring that a test/victim participant was excluded in each iteration.

5.5.5 Keyboard Type Inference

A preliminary step for an attacker aiming to execute a keystroke inference attack is to infer the type of keyboard the victim employs. Given the distinct acoustic signatures produced by different keyboard types, such as mechanical or membrane, understanding the keyboard type can pave the way for a more targeted and effective attack. In our study, we gathered data from two distinct brands for each of the keyboard categories: **K1**: mechanical, **K2**: membrane, and **K3**: laptop-based membrane keyboards. The rationale behind using two models for each category was to introduce a level of complexity to the inference task. If the model were trained solely on data from a single brand for each category, it would trivially classify that brand during testing. Our objective with keyboard type inference is to generalize across brands and variations within each category, ensuring the model can identify the overarching category to which they belong. For the purpose of type inference, we segment the keyboard input audio data into 30-second windows, extracting 6 MFCC features and Root-Mean-Square-Energy or RMSE (eq. (5.1)) for each segment. We observed during our experiments that, while keystroke inference demands a much more detailed feature set, keyboard type inference can be effectively achieved with just these 6 MFCC features. Subsequently, we employ a *multi-class logistic regression* model trained on this data to predict the keyboard type. The keyboards tested in our keyboard type inference experiment are as follows:

- **K1**: Monoprice MP810 (with red switches) [129] and Aukey KMG12 (with blue switches) [21].
- **K2**: Logitech K120 [108] and Dynex DX-PNC2019 [55].
- **K3**: Tecknet Ultra Slim Compact [181] and the keyboard of the HP Envy x360 15" laptop [80].

Algorithm 5.2 Word Prediction with SymSpell.

Require: $predictions$: top-k letter predictions
Require: $topK$: number of words to return
function SpellCorrection($predictions$, $topK$)
Initialize sym_spell and load dictionary
$possible\_combinations \leftarrow$ GenerateCombinations($predictions$)
Initialize $predicted\_words\_with\_counts$ as empty list
for each $input\_term$ in $possible\_combinations$ **do**
    Get $suggestions$ for $input\_term$ from sym_spell
    Append unique $suggestions$ to $predicted\_words\_with\_counts$
Sort $predicted\_words\_with\_counts$ by word frequency
return First $topK$ words from $predicted\_words\_with\_counts$
end function

5.5.6 Word Prediction

After predicting individual keystrokes, we further look into the possibility of increasing the effectiveness of our attack in a context aware manner by predicting the possible word (comprising of the inferred keystrokes). To this end, we mainly explored two methods to tackle our word prediction task. The first approach is a naive dictionary-based method, where each sequence derived from the top-k predicted keys (i.e., the correct key is among the top k predicted keys from the aforementioned key prediction models) is cross-referenced with a predefined dictionary. If a sequence matches an entry within the dictionary, it is deemed a valid word. However, this method has its limitations, especially when the key predictions contain inaccuracies such incorrect key predictions, missing certain keystrokes or having extra keystrokes. Our second method leverages the *SymSpell* algorithm [66]. SymSpell is a spelling correction algorithm that works by pre-computing possible spelling errors for every word in its dictionary, up to a given edit distance. Instead of searching for possible corrections during the lookup, it directly utilizes this pre-computed data to

identify close matches. This design allows for rapid and memory-efficient word predictions and corrections, making it particularly suitable for our scenario due to the possibility of presence of missing, incorrect or extra keystrokes. The procedure (see algorithm 5.2) starts by initializing the SymSpell library and loading a comprehensive frequency dictionary. Next, given a set of top-k letter predictions, we generate all possible word combinations. For each of these generated terms, we consult SymSpell to gather the closest matching words in the dictionary. This results in a collection of suggested words, each associated with its frequency of usage. Finally, to provide the most probable corrections, we sort the accumulated suggestions based on their word frequencies and return the top-w predictions as the output. The rationale being that words that appear more frequently in the language (or specific corpus) are more likely to be the intended word when a spelling error is made. In essence, the frequency of usage helps in prioritizing common words over less common ones when suggesting corrections.

5.5.7 Experimental Setup

Due to the absence of published publicly-available APIs for accelerometer data extraction in the current generation of commercial headphones, and the limitations associated with the few available APIs such as those from Apple, we devised a custom setup to comprehensively evaluate all the parameters and settings of our framework. Nonetheless, to test the efficacy of our framework on commercially available hardware, we also conducted some limited experiments using Apple AirPods.

Our primary experimental setup comprises of a Raspberry Pi, and a 3D-printed over-the-ear headphone prototype (see fig. 5.6a). To capture audio, we equipped each earpiece with an Adafruit I2S MEMS Microphone [3] and a MPU-6500 accelerometer [179] to record the motion data. The audio was sampled at a maximum of 96kHz while accelerometer was sampled at a maximum of 500Hz. The microphones and the accelerometers were connected to the Raspberry Pi via the GPIO interface and a Python script was used to record the data from each microphone/sensor. For our tests involving commercially-available hardware, we employed a pair of Apple AirPods Pro (2nd

Generation) [15] connected to an iPhone SE [18]. We developed an iOS application using Swift, which utilized the `CMHeadphoneMotionManager` API [17] to collect motion data (note: this API does not allow specifying a sampling rate) and captured audio data at a sampling rate of $44kHz$, the maximum permitted. The data processing and inference framework evaluation was done on a Ubuntu 22.04 virtual machine with 32GB memory and 32 cores using Python 3.10.

(a) (b)

Figure 5.6: A user wearing (a) our custom-built headphone setup, (b) AirPods.

Our experiments included a diverse range of keyboards to ensure a comprehensive evaluation. We used a AUKEY KMG12 [21], a full-sized mechanical keyboard (104 keys) to represent **K1** category. For the **K2** category, we utilized a Logitech K120 [108], another full-sized model. To closely replicate a keyboard of modern laptops a Tecknet Ultra Slim Compact keyboard (68 keys) [181], was used (representing the **K3** category). This diversity in keyboard types allowed us to assess the robustness and adaptability of our framework across different tactile feedback mechanisms and form factors.

5.5.8 Data Collection

We recruited 17 participants in the age range of 18-38, to collect typing data using our custom prototype pair of headphones. The participants conducted two experiments across three types of keyboards while wearing our custom headset (as described in section 5.5.7). The experiments

were conducted inside a closed office space. For the first experiment, they typed individual English alphabets displayed on the screen one at a time, where each alphabet is repeated for five times at random. The second experiment involved them typing 300 English words from a list of 5000 most frequent words, with number of letters (in each word) ranging from three to seven [197]. Participants engaged in the typing tasks using a 24-inch computer monitor. To ensure comfort and a natural typing posture, they were provided with a height adjustable chair, allowing them to choose a seating level they found most comfortable. We did not impose any specific typing techniques on the participants; instead, they were encouraged to type in a manner consistent with their daily habits. For the additional commercial headphone setup using AirPods, we recruited 5 participants who performed the same aforementioned experiments on keyboard **K1**. The participants also filled out a short survey (see section 5.6.11) at the end of the experiments which included questions relating to their typing behaviors and headphone usage. Some useful/interesting insights obtained via the survey are given in section 5.6.12. Our data collection procedures were approved by our institution's Institutional Review Board (IRB).

5.6 Evaluation

Next, we present the evaluation results for *OverHear* under a wide variety of different experimental settings and conditions.

5.6.1 Metrics

We use the following metrics for quantifying the performance of *OverHear*.

Precision and Recall. Precision measures the number of correctly predicted keystroke segments out of the total predicted keystroke segments, while recall (or sensitivity) calculates the number of correctly predicted keystroke segments out of the actual keystroke segments. We also use precision and recall to measure the prediction performance of our keyboard type inference module.

Top-k_{key} Accuracy. This metric evaluates the accuracy of the top-k key predictions. Specifically, if the true label is within the top-k predicted labels, then the prediction is considered correct.

We utilized top-k_{key} accuracy for assessing the performance of our *OverHear* framework at an individual key prediction level.

Top-k_{word} Accuracy. Evaluates the accuracy of the top-k word predictions. If the true word is within the top-k predicted words, the prediction is deemed accurate.

5.6.2 Keyboard Type Inference

Across all keyboard type categories, our keyboard type identification model demonstrated robust performance, consistently achieving an accuracy exceeding 0.95 when the training data includes data from the same brand of keyboard. However, when the type inference model is trained using one keyboard brand for each category, the performance slightly degrades. Nevertheless, except for **K2** category of keyboards, both the other categories (see table 5.1) demonstrated a recall over 0.95.

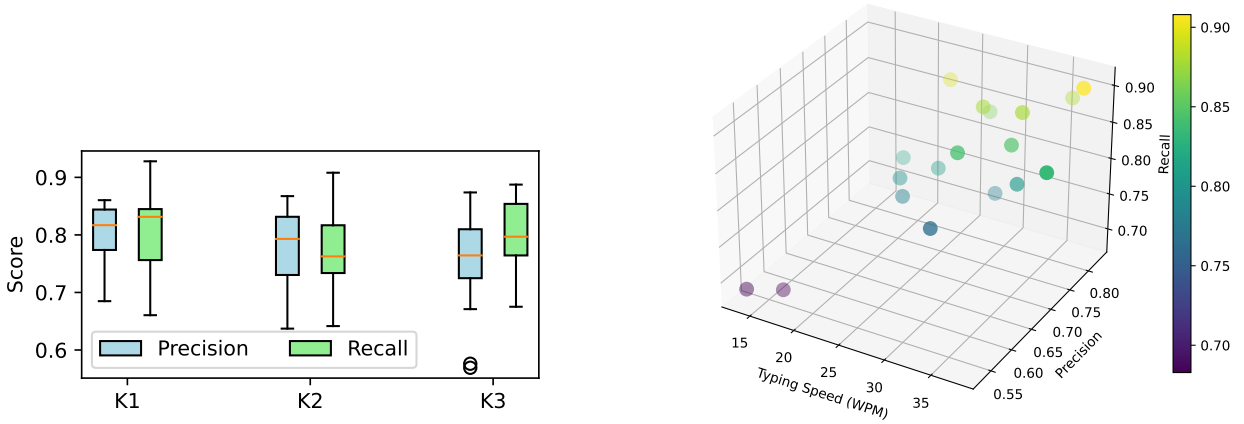
Table 5.1: Keyboard Type Inference Performance.

Keyboard Type	Precision	Recall
Mechanical (K1)	0.86	0.96
Membrane Type 1 (K2)	0.98	0.76
Membrane Type 2 (K3)	0.88	0.99

5.6.3 Keystroke Detection

Our keystroke segmentation algorithm exhibited consistent performance across various keyboard types (see fig. 5.7a). For keyboard type **K1**, the precision and recall were both measured at 0.80 (σ=0.05 and σ=0.08, respectively). For **K2**, the precision was 0.78 (σ=0.07) and the recall was 0.77 (σ=0.07). For **K3**, we observed a precision of 0.75 (σ=0.08) and a recall of 0.80 (σ=0.06). While these results indicate stability in performance, there are inherent challenges that contribute to the slightly lower accuracy. One primary challenge arises when keys are pressed quickly in succession. Particularly with adept typists, the acoustic energy from one key can overlap with the subsequent key, complicating the distinction between individual keystrokes. Further, the unique typing dynamics of each individual introduce variability. Some users exert varied force

on keys, while others occasionally press two keys nearly concurrently. These issues add layers of complexity to the detection process.



(a) (b)

Figure 5.7: (a) Keystroke detection performance for different keyboards types. (b) Precision and recall for keystroke detection vs. user typing speed (WPM) for **K1**.

5.6.4 Overall Performance

As mentioned earlier, we evaluated *OverHear*'s performance across three keyboard type categories, **K1**, **K2**, **K3**. The results revealed that, **K1**, achieved the highest top-5_{key} accuracy of 0.77 and a top-10_{key} accuracy of 0.88. In comparison, **K2**, recorded a top-5_{key} accuracy of 0.58, and **K3**, had a top-5_{key} accuracy of 0.53. Mechanical keyboards (category **K1**) tend to produce near distinct tactile feedback and sound profiles for each key press. This unique acoustic signature for each key can make it easier for the system to differentiate between keystrokes, leading to higher accuracy as observed in our results. While larger external membrane keyboards (category **K2**) too produce some amount of tactile feedback, the sound profiles might not be as distinct as those of mechanical keyboards. The smaller membrane keyboards (type **K3**), which closely resembles laptop keyboards, typically have keys closer together leading to overlapping or less distinct sound profiles, especially when keys are pressed in rapid succession. Additionally, the build and material of such keyboards might dampen the sound further, making it harder to infer keystrokes accurately.

For evaluating the efficacy of our clustering algorithm, we compared the accuracy of our model

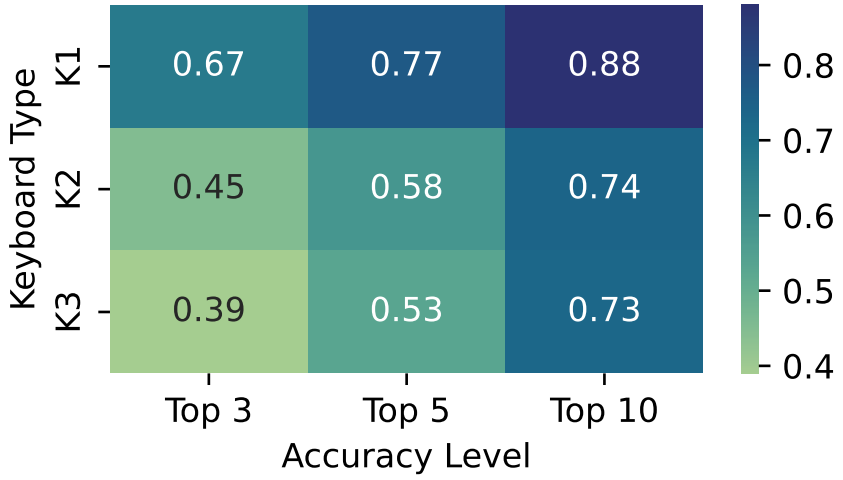


Figure 5.8: Top-k_{key} accuracy across keyboard types.

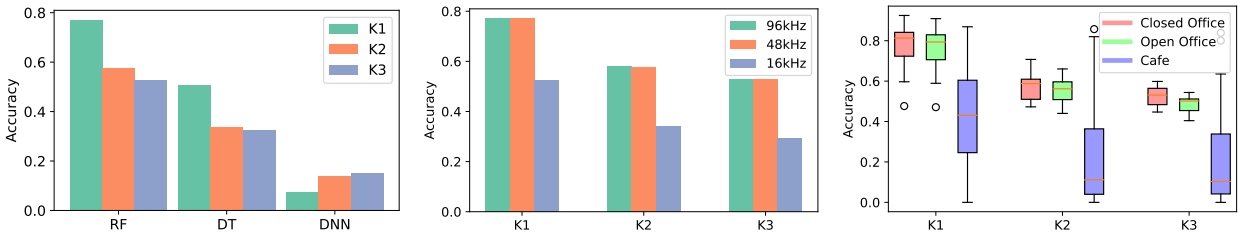
with and without the clustering approach. As presented in table 5.2, the clustering algorithm considerably enhanced the accuracy across all keyboard types. For the mechanical keyboard type, **K1**, the top-5$_{key}$ accuracy improved from 0.59 to 0.77. Similarly, for the membrane type, **K2**, there was a noticeable increase from 0.41 to 0.58. The membrane type, **K3**, also saw an enhancement in accuracy, with top-5$_{key}$ accuracy rising from 0.37 to 0.52. These results shows the effectiveness of the accelerometer based clustering algorithm in refining the keystroke inference, making it an important component of *OverHear*.

Table 5.2: Top-5$_{key}$ accuracy with and without clustering.

Keyboard Type	w/o Clustering	with Clustering
K1	0.59	0.77
K2	0.41	0.58
K3	0.37	0.53

In our evaluations, the Random Forest classifier consistently outperformed other models across all keyboard types (see fig. 5.9a). Specifically, for **K1**, it achieved a top-5$_{key}$ accuracy of approximately 0.77, while for **K2** and **K3**, the accuracies were 0.57 and 0.53, respectively. In contrast, the Decision Tree classifier managed a top-5$_{key}$ accuracy of 0.57 for **K1** and around 0.33 for both **K2** and **K3**. The Deep Neural Network (DNN) model was the least effective, with accuracies falling below 0.15 for all keyboard categories. The Random Forest classifier outperformed other models

likely due to its ensemble nature, effectively capturing complex patterns without overfitting. In contrast, the Deep Neural Network (DNN) model struggled, likely because DNNs require substantial amount of data for effective training, beyond the dataset that we collected. Moving forward, exploring alternative models such as single- and few-shot learning techniques could potentially offer more robust solutions, especially when training data is limited [195].



(a) (b) (c)

Figure 5.9: Top-5_{key} accuracy for different (a) models, (b) sampling rates, (c) types of ambient noise.

5.6.5 Sampling Rates

In our experiments, we explored the impact of different sampling rates on the performance of *OverHear* (see fig. 5.9b), starting from our default rate of 96kHz for the audio signal. Our findings indicate that the performance at 48kHz is nearly on par with that at 96kHz. However, when the sampling rate is reduced to 16kHz, we observed a considerable degradation in performance. This indicates that our framework can work fairly well even at lower sampling rates.

5.6.6 Ambient Noise

We evaluated the robustness of *OverHear* under various ambient noise conditions. These noise profiles were separately collected in three different environments and later combined with the keystroke data from all participants. In a university cafeteria setting, the environment was busy with people eating, working on their laptops, and background music playing. The open office space had 2-3 individuals working nearby on computers, accompanied by the typical sounds of typing, mouse clicks, and the occasional mobile phone ringing. In contrast, the closed office space

provided a quiet environment with minimal background noise. From the results (see fig. 5.9c), it's evident that accuracy is highest in quieter environments, such as a closed office, and decreases with increasing ambient noise, with the cafeteria setting being the most challenging with accuracies dropping below 0.45. This trend is consistent across all three keyboard types. This shows that our framework works well in quieter (ideal) environments, but the accuracy is reasonable even in noisier (less ideal) environments.

5.6.7 Word Prediction

Our word prediction technique, leveraging the *SymSpell* library, as seen in fig. 5.10, demonstrated considerable efficacy, achieving top-50_{word} accuracies nearing 0.50 across all keyboard categories. Further, **K1** reached a 0.76 accuracy for top-100_{word} predictions, while **K2** and **K3** closely followed with 0.71 and 0.70, respectively. For **K1**, five out of six, participants achieved an accuracy exceeding 0.60 for top-50_{word} predictions, with only one participant falling below the 0.40 mark. In the case of **K2** and **K3**, barring one outlier in each category, all participants consistently achieved around the 0.50 accuracy level for top-50_{word} predictions. In contrast, the naive dictionary-based approach, which relies solely on exact matches, lagged behind. Its limited adaptability to variations in keystroke data meant it consistently registered accuracies below 0.4 for all keyboard types. These results can be attributed to SymSpell's ability to efficiently handle typographical errors, which aids in more accurate word suggestions, especially in the presence of potentially inaccurate key predictions along with extra or missing keystrokes.

5.6.8 Effect of Typing Speeds

Encompassing keystroke segmentation and subsequent processing, our framework demonstrates consistent key prediction performance across varying typing speeds (see fig. 5.11). While the majority of participants yielded consistent accuracy, an exception was observed in one participant with a typing speed of approximately 35 WPM, who registered an accuracy below 0.5. Despite this outlier, the overall robustness of the framework across varied typing speeds is evident,

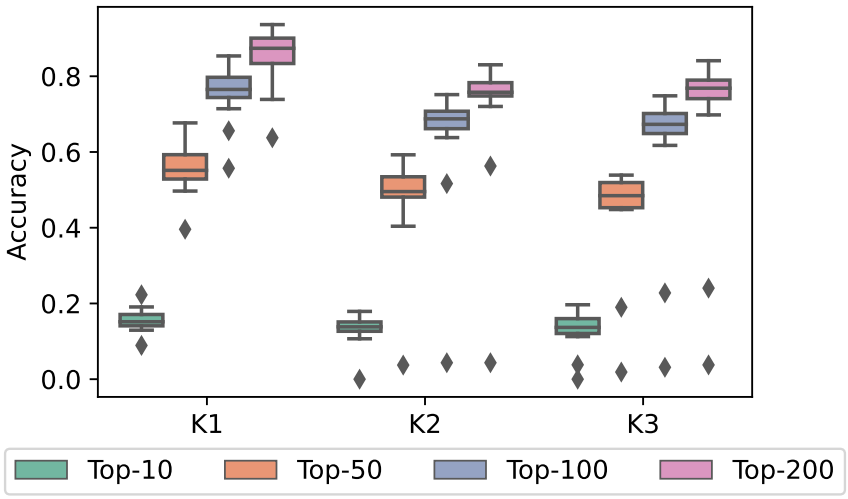


Figure 5.10: Top-k_{word} accuracies.

highlighting its adaptability to diverse user behaviors.

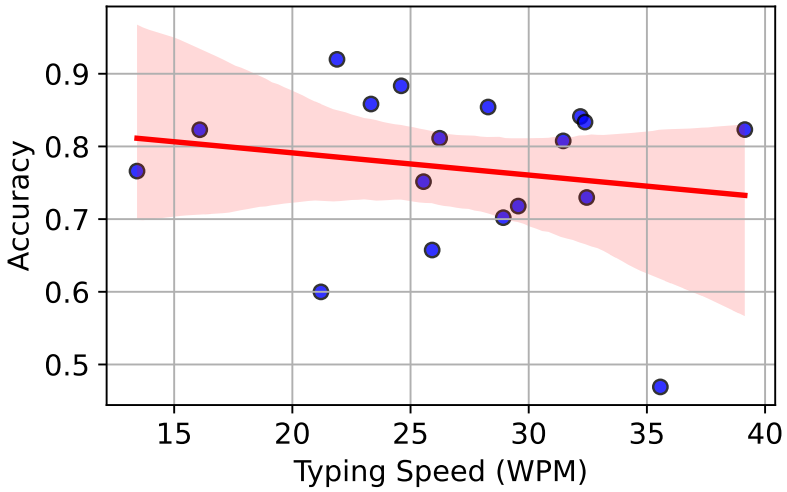
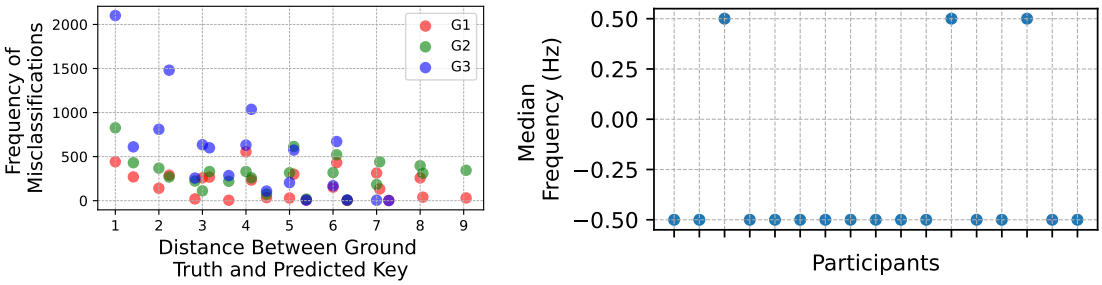


Figure 5.11: Typing speeds vs. key-level accuracy.

5.6.9 Factors Affecting Accuracy

In our analysis of the factors that may be affecting accuracy of *OverHear*, several patterns emerged. For the key group G_2, no major misclassification patterns were observed. However, there were instances where the key 'k' is misinterpreted as 'i', and 'm' is confused with 'n'. In the G_3 group, the keys 'j' and 'h' as well as 'y' and 'r' are often interchanged. Furthermore, the

keys ‘q’ and ‘w’ consistently exhibit lower accuracy rates. In G_1 we observed that the key ‘z’ is frequently misclassified as ‘x’, and ‘x’ is often mistaken for ‘s’. Figures 5.13 to 5.15 present confusion matrices for key predictions across groups G_1, G_2, and G_3. Notably, while G_2 and G_3 exhibit minimal misclassifications, in G_1, most keys surrounding ‘a’ are frequently mistaken for ‘a’. These confusion patterns can be mostly accounted to the spatial closeness of these keys on the keyboard leading to similar acoustic profiles that can be challenging to distinguish. Figure 5.12a visualizes this relationship between the frequency of misclassifications and the Euclidean distance between ground truth and predicted keys on a QWERTY keyboard, categorized into our three key groups: G_1, G_2, and G_3. A prominent observation from the plot is that misclassifications are more frequent for keys that are closer in distance, particularly for the G_3 group. This suggests that keys in the G_3 group are often confused with their immediate neighbors on the keyboard. The plot shows the inherent challenge in distinguishing between keystrokes of adjacent keys, emphasizing the spatial aspect of the misclassification problem on a physical keyboard layout.



(a) (b)

Figure 5.12: (a) Frequency of misclassifications vs. key distance. (b) Participants vs. their median gyroscope frequency.

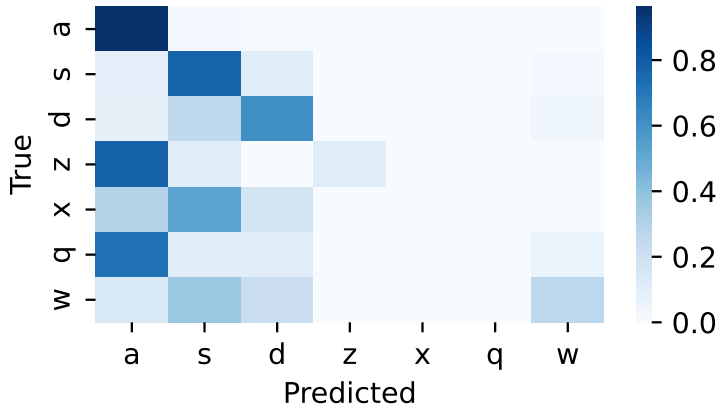


Figure 5.13: Confusion matrix for group G_1 keys.

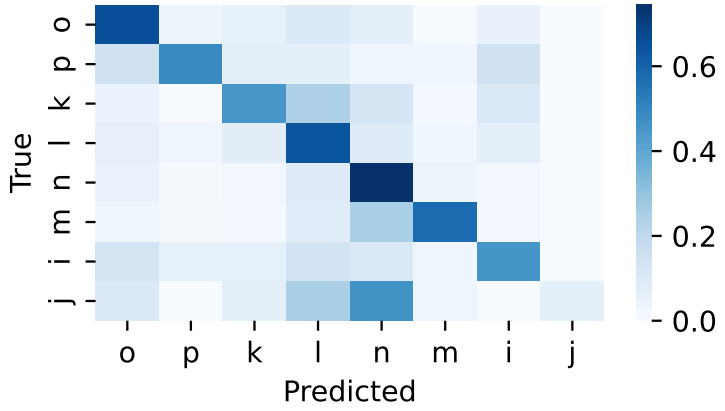


Figure 5.14: Confusion matrix for group G_2 keys.

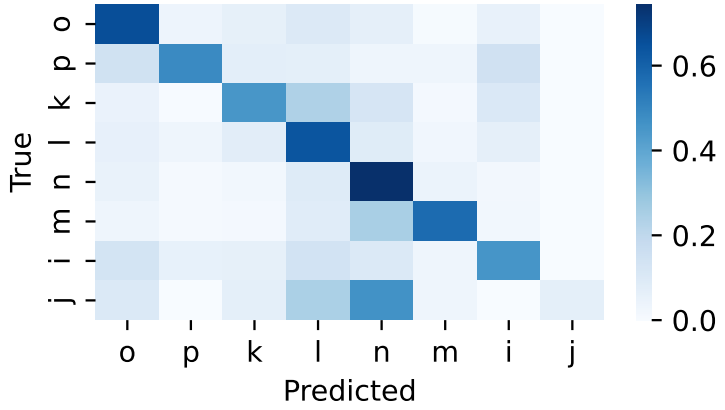


Figure 5.15: Confusion matrix for group G_3 keys.

Furthermore, certain users possess the ability to type without glancing at the keyboard, maintaining a steady gaze on the screen. This consistent posture ensures minimal head movement, leading to relatively stable acoustic profiles. Conversely, users who frequently look down at the keyboard introduce regular vertical head movements. These continuous and pronounced shifts can also influence the acoustic signatures, potentially challenging the models employed by our framework, to accurately distinguish and classify keystrokes. To get a better understanding of such head movements and their potential correlation to the accuracy of *OverHear*, we analyzed the frequency spectrum of the gyroscope data (collected alongside accelerometer data in our custom setup). Particularly, we looked at the median frequency of each participant, which can be considered as the frequency below which 50% of the power of the signal lies. A higher median frequency in gyroscope data typically indicates more rapid changes in the signal, which can be interpreted as more intense or faster head movements. As it can be seen in fig. 5.12b, the three participants with the higher median gyroscope frequency showed the lowest top-5_{key} accuracies with values below 0.70. This could be intuitively attributed to the fact that more intense head movements might introduce more noise or variability in the audio/accelerometer data, making keystroke inference more challenging.

5.6.10 Performance on Commercial Devices

We tested our inference framework using a pair of Apple AirPods and focused on keyboard type **K1**. However, challenges arose due to the accelerometer API provided by iOS, which did not deliver a consistent data stream. Significant gaps in the accelerometer data collection affected our ability to utilize this data effectively. Consequently, we were compelled to rely solely on audio data for our analysis. Despite these limitations, our framework, when tested only with audio data, achieved a top-5_{key} prediction accuracy of 56% and a top-10_{key} prediction accuracy of 80%. These results are comparable (despite being slightly lower) to the performance of our framework when tested without the clustering step using our custom hardware setup (refer to table 5.2). These results underscore the potential of audio-based keystroke inference using smart headphones, even

under limiting conditions and despite the noise-cancelling (in the microphones) features inherent in these commercial headphones' audio recording functionality.

5.6.11 Participant Survey

Below we present the participant survey the study participants filled out during the data collection.

1. What is your age?
2. What is your dominant hand?
   Right, Left
3. Do you currently own any of the following digital devices?
   - Yes, No - Bluetooth Over-the-Ear Headsets
   - Yes, No - Bluetooth Earbuds
4. How often do you type on a computer keyboard while wearing a pair of headsets/earbuds?
   Never, Rarely, Sometimes, Often, Always
5. How many hours a day do you perform typing tasks in general?
6. How many hours a day do you perform typing tasks while wearing a headset/earbuds?
7. What type of a keyboard do you own?
   Membrane, Mechanical,Not Sure
8. Where do you typically perform your typing tasks?
   café, library, classroom, home, other (please specify)
9. Do you keep your headphones/earbuds near your computer/ keyboard while not wearing them?
   Yes, No

10. Have you installed the smartphone app that comes bundled with your headset/earbuds?
    Yes, No

11. Are you aware that modern headphones and earbuds have embedded motion sensors in them?
    Yes, No

5.6.12 Insights from Participant Survey Responses

Through the responses from our participant survey, we were able to gain the below insights into the individual typing behaviors and the prevalent patterns of headphone usage among participants.

- A majority of participants (52.94%) often type on a computer keyboard while wearing headphones or earbuds, with 17.65% doing so always. On average, 11.76% type for 2.5 hours per day with these devices on (see fig. 5.16b).
- Regarding awareness of modern headphone technology, 52.94% of participants know that headphones and earbuds often have embedded motion sensors..
- In terms of keyboard ownership (see fig. 5.16c):
  - 47.06% own a membrane keyboard.
  - 11.76% own a mechanical keyboard.
  - 41.18% are uncertain about their keyboard type.
- A significant 88.24% of participants keep their headphones or earbuds near their computer or keyboard when not in use, while only 11.76% store them away (see fig. 5.16d).
- As for typing locations, 33.33% of participants typically type at their home or apartment, and 24.24% prefer the library (see fig. 5.16a).

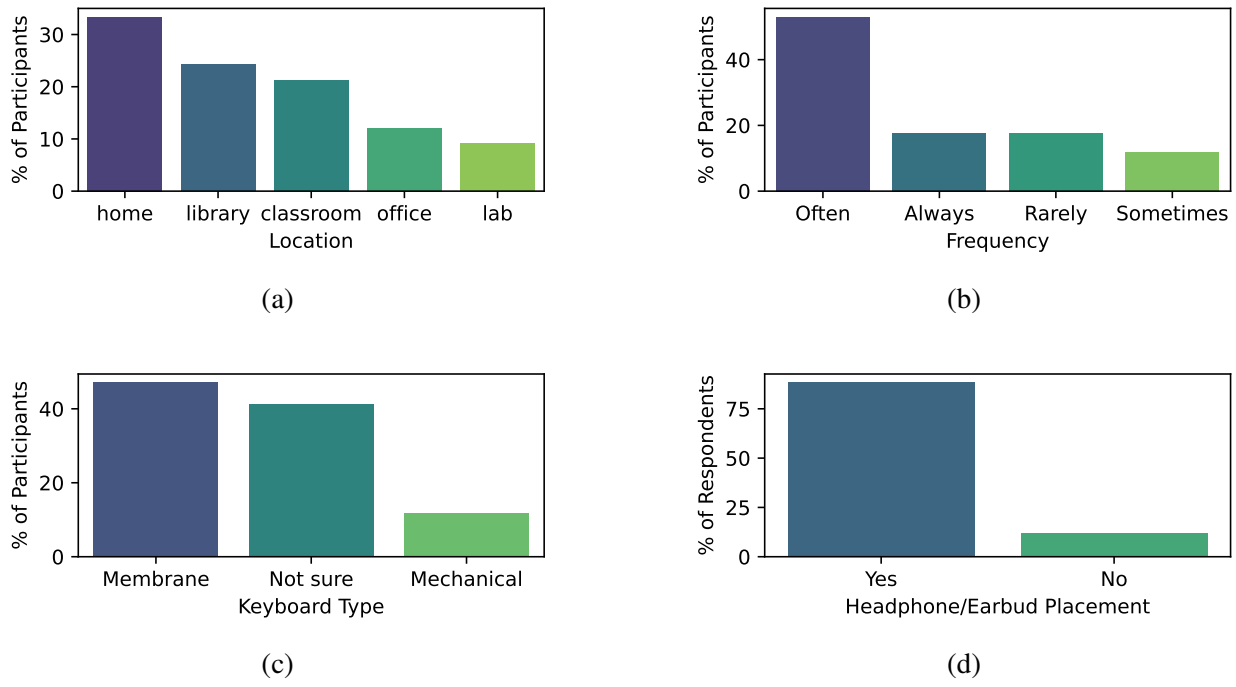


Figure 5.16: (a) Distribution of locations for typing tasks. (b) Frequency of typing on a computer keyboard while wearing headsets/earbuds. (c) Distribution of keyboard types owned. (d) Distribution participants who do/do not keep their headphones/earbuds near the computer.

5.7 Discussion and Conclusion

Attack Impact. The widespread use of headphones, such as Pixel Buds [70] and Apple AirPods [15], equipped with advanced microphones and motion sensors, underscores the potential reach of our keystroke inference attack. These features enhance the user experience but also make the devices susceptible to *OverHear*. Our results demonstrate high accuracy on both mechanical and membrane keyboards, confirming the attack's feasibility in real-world scenarios. Question 7 of the participant survey (section 5.6.11) further corroborates the threat landscape. With 12% of respondents using mechanical keyboards and 47% on membrane keyboards it's evident that a large segment of users could be vulnerable. Specifically, mechanical keyboards have seen a resurgence in popularity over the past decade, especially among certain demographics, such as, the gaming community, technology enthusiasts and professional typists [180]. Given the projected Compound Annual Growth Rate (CAGR) of 6.79% from 2022-2027, it's evident that this user base is not only substantial but also expanding [180]. Our inference framework, which demonstrates heightened

efficacy for mechanical keyboards, poses a notable threat to this growing user base.

Mitigations. Noise-canceling features included in most modern headphones offers a promising avenue to counteract the threat of acoustic keystroke inference. Originally designed to minimize ambient sounds, this can be further optimized to specifically target and suppress the key press sounds. While it's challenging to completely mute the sound of keystrokes, integrating such targeted noise-canceling features to the headphones can significantly degrade the quality of captured keystroke sounds, thereby reducing the effectiveness of inference attacks. Using quieter keyboards such as Scissor keyboards [92], commonly found in laptops, can also mitigate such attacks due to their minimal key-press noise from their scissor-like hinge structure. Additionally, implementing system-level monitoring services on paired devices, such as smartphones or computers, can help. These services would track the volume and frequency of data transmission by apps; any unusual activity, such as excessive data offloading by a headphone companion app, could then be flagged for review.

Limitations. Despite promising results, our keystroke inference framework using headphones has limitations. Accurately predicting complex passwords (mixing alphanumeric characters, symbols, and cases) remains challenging, especially for unconventional passwords. Additionally, performance degrades considerably in very noisy environments (as shown in section 5.6.6).

Conclusion. In this study, we presented *OverHear*, a framework that adeptly harnesses both acoustic and accelerometer data from smart headphones to infer keystrokes, achieving a top-5 key accuracy nearing 80% for mechanical keyboards and 60% for membrane keyboards. Further, we were able to achieve top-100 word accuracy of over 70% for all categories of keyboards. While our results highlight the vulnerabilities introduced by modern headphones in real-world scenarios, they also emphasize the importance of understanding and addressing these emerging security challenges.

Acknowledgment

Research reported in this publication was supported by the Division of Computer and Network Systems (CNS) of the National Science Foundation (NSF) under award numbers 1828071 and 1943351.

CHAPTER 6: CONCLUSION

The rapid advancement and widespread adoption of smart wearables have revolutionized the digital landscape, offering users capabilities that extend beyond traditional smartphone features. These wearable devices, equipped with a diverse array of sensors, provide extensive functionality but also introduce significant privacy risks. This dissertation explores practical applications using such sensors and addresses privacy concerns (arising from such sensors) by evaluating the potential for privacy breaches via sensory side-channels, with a special focus on practical usage scenarios.

This dissertation begins by exploring potential practical applications and communication protocols for wearable devices using motion sensors. To this end, it first investigates the feasibility of wrist motion-based user authentication from handwriting. The investigation highlights promising results from some of the tested schemes; however, practical challenges related to user-dependent behavior and technique-specific limitations hinder their widespread adoption in mainstream applications. Then, as another practical application, this dissertation introduces a vibration-based communication protocol that leverages the human body as a communication medium. This protocol demonstrates the potential for creating a low-bandwidth and covert communication channel between mobile and wearable devices, achieving stable bandwidth with low bit error rates.

Next, the dissertation explores sensory side-channel attacks on wearable devices with a focus on motion sensors. To this end, it first examines the practicality of handwriting inference using wrist-wearable motion sensors. The findings reveal that accurate and practical handwriting inference using consumer-grade wrist wearables is challenging due to unique and inconsistent handwriting behaviors observed in natural writing settings. This dissertation then explores the feasibility of inferring user keystrokes on external keyboards through sensory side-channels in modern headphones. The proposed framework combines acoustic and motion data to infer keystrokes with notable accuracy, highlighting the potential privacy risks associated with sensor-equipped modern headphone devices.

In summary, this dissertation provides comprehensive insights into both the feasibility of prac-

tical applications and the privacy risks posed by sensory side-channels in smart wearables. The findings identify threats to user privacy and provide defenses and mitigations, underscoring the need for continued research and development of secure applications and communication protocols for wearable devices to safeguard user privacy while leveraging their full potential.

BIBLIOGRAPHY

[1] Martín Abadi, Ashish Agarwal, Paul Barham, and et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] ABC. Lawsuit alleges Bose collecting user data without permission. `https://abc7chicago.com/bose-consumer-concerns-spying-data-mining/1897655/`, 2017. [Online; accessed 15-Jun-2024].

[3] Adafruit. Adafruit I2S MEMS Microphone. `https://www.adafruit.com/product/3421`, 2023. [Online; accessed 15-Oct-2023].

[4] Ariel Adams. Poll: What is Your Hand-Orientation & What Wrist Do You Wear Your Watch On? `http://www.ablogtowatch.com/poll-your-hand-orientation-what-wrist-wear-your-watch/`, 2019. Online; accessed 2019-01-25.

[5] Kamran Ali and Alex X Liu. Fine-grained vibration based sensing using a smartphone. *IEEE Transactions on Mobile Computing*, 2021.

[6] Shalbbya Ali, Safdar Tanweer, Syed Sibtain Khalid, and Naseem Rao. Mel frequency cepstral coefficient: a review. *Proceedings of the 2nd International Conference on ICT for Digital, Smart, and Sustainable Development (ICIDSSD)*, 2020.

[7] Christoph Amma, Dirk Gehrig, and Tanja Schultz. Airwriting recognition using wearable motion sensors. In *Proceedings of the 1st Augmented Human International Conference*, pages 1–8, 2010.

[8] Christoph Amma, Marcus Georgi, and Tanja Schultz. Airwriting: Hands-free mobile text input by spotting and continuous recognition of 3d-space handwriting with inertial sensors. In *ACM International Symposium on Wearable Computers (ISWC)*, 2012.

[9] Christoph Amma, Marcus Georgi, and Tanja Schultz. Airwriting: A wearable handwriting recognition system. *Springer Personal and Ubiquitous Computing*, 18(1), 2014.

[10] S Abhishek Anand and Nitesh Saxena. Keyboard emanations in remote voice calls: Password leakage and noise (less) masking defenses. In *Proceedings of the 8th ACM Conference on Data and Application Security and Privacy (CODASPY)*, 2018.

[11] S Abhishek Anand and Nitesh Saxena. Speechless: Analyzing the threat to speech privacy from smartphone motion sensors. In *IEEE Symposium on Security and Privacy (S&P)*, 2018.

[12] Anker. soundcore. `https://apps.apple.com/us/app/soundcore/id1331876603`, 2024. [Online; accessed 15-Feb-2024].

[13] Apple. AirPods. `https://www.apple.com/airpods-2nd-generation/`, 2022. [Online; accessed 15-Oct-2023].

[14] Apple. AirPods Max. `https://www.apple.com/airpods-max/`, 2022. [Online; accessed 15-Oct-2023].

[15] Apple. AirPods Pro. `https://www.apple.com/airpods-pro/specs/`, 2022. [Online; accessed 15-Oct-2023].

[16] Apple. AVAudioRecorder. `https://developer.apple.com/documentation/avfaudio/avaudiorecorder`, 2024. [Online; accessed 15-Feb-2024].

[17] Apple. CMHeadphoneMotionManager. `https://developer.apple.com/documentation/coremotion/cmheadphonemotionmanager`, 2024. [Online; accessed 15-Feb-2024].

[18] Apple. iPhone SE. `https://www.apple.com/iphone-se/`, 2024. [Online; accessed 15-Feb-2024].

[19] Luca Ardüser, Pascal Bissig, Philipp Brandes, and Roger Wattenhofer. Recognizing text using motion data from a smartwatch. In *IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pages 1–6, 2016.

[20] Dmitri Asonov and Rakesh Agrawal. Keyboard acoustic emanations. In *IEEE Symposium on Security and Privacy (SP)*, 2004.

[21] AUKEY. AUKEY KMG12 Mechanical Keyboard 104key with Gaming Software. `https://www.aukey.com/products/km-g12-mechanical-keyboard-blue-switches`, 2023. [Online; accessed 15-Oct-2023].

[22] Jia-Xuan Bai, Bin Liu, and Luchuan Song. I know your keyboard input: a robust keystroke eavesdropper based-on acoustic signals. In *Proceedings of the 29th ACM International Conference on Multimedia*, 2021.

[23] Yang Bai, Jian Liu, Li Lu, Yilin Yang, Yingying Chen, and Jiadi Yu. Batcomm: enabling inaudible acoustic communication with high-throughput for mobile devices. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, pages 205–217, 2020.

[24] Adeola Bannis, Hae Young Noh, and Pei Zhang. Bleep: motor-enabled audio side-channel for constrained uavs. In *Proceedings of ACM MobiCom'20*, 2020.

[25] Michael Barr. Pulse width modulation. *Embedded Systems Programming*, 14(10):103–104, 2001.

[26] M. Bashir, G. Scharfenberg, and J. Kempf. Person authentication by handwriting in air using a biometric smart pen device. *Proceedings of the Biometrics Special Interest Group (BIOSIG)*, 2011.

[27] BBC. Ring doorbell 'gives Facebook and Google user data. `https://www.bbc.com/news/technology-51281476`, 2020. [Online; accessed 15-Oct-2023].

[28] Vincent Becker, Linus Fessler, and Gábor Sörös. Gestear: combining audio and motion sensing for gesture recognition on smartwatches. In *Proceedings of the 2019 ACM International Symposium on Wearable Computers (ISWC)*, 2019.

[29] John Bellardo and Stefan Savage. 802.11 denial-of-service attacks: Real vulnerabilities and practical solutions. In *12th USENIX Security Symposium (USENIX Security 03)*, volume 12, pages 2–2. Washington DC, 2003.

[30] Yoshua Bengio, Yann LeCun, Craig Nohl, and Chris Burges. Lerec: A nn/hmm hybrid for on-line handwriting recognition. *Neural Computation*, 7(6), 1995.

[31] Ralph E Blake. Basic vibration theory. *Shock and vibration handbook*, 1:2–8, 1961.

[32] Gerrit Bloothooft, Eldrid Bringmann, Marieke Van Cappellen, Jolanda B Van Luipen, and Koen P Thomassen. Acoustics and perception of overtone singing. *The Journal of the Acoustical Society of America*, 92(4):1827–1836, 1992.

[33] A. Buriro, B. Crispo, F. Del Frari, and K. Wrona. Touchstroke: Smartphone user authentication based on touch-typing biometrics. In *International Conference on Image Analysis and Processing (ICIAP)*, pages 27–34. Springer, 2015.

[34] A. Buriro, R. Van Acker, B. Crispo, and A. Mahboob. Airsign: a gesture-based smartwatch user authentication. In *International Carnahan Conference on Security Technology (ICCST)*, pages 1–5, 2018.

[35] Liang Cai and Hao Chen. Touchlogger: Inferring keystrokes on touch screen from smartphone motion. In *HotSec*, 2011.

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

[37] Liang Cai, Sridhar Machiraju, and Hao Chen. Defending against sensor-sniffing attacks on mobile phones. In *ACM MobiHeld*, pages 31–36, 2009.

[38] Stefano Cecconello, Alberto Compagno, Mauro Conti, Daniele Lain, and Gene Tsudik. Skype & type: Keyboard eavesdropping in voice-over-ip. *ACM Transactions on Privacy and Security (TOPS)*, 2019.

[39] Christina Summer Chen. Earpieces with gesture control, 2015. US Patent App. 13/959,109.

[40] Kefei Cheng, Wenqi Li, Liang Zhang, Xiangjun Ma, and Jinghao Chen. Dictionary attacks based on tdoa using a smartphone. In *IEEE 6th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, 2022.

[41] L. Chevalier, S. Sahuguède, and A. Julien-Vergonjanne. Wireless optical technology based body area network for health monitoring application. In *2015 IEEE International Conference on Communications (ICC)*, pages 2863–2868, 2015.

[42] Ludovic Chevalier, Stephanie Sahuguede, and Anne Julien-Vergonjanne. Optical wireless links as an alternative to radio-frequency for medical body area networks. *IEEE Journal on Selected Areas in Communications*, 2015.

[43] Ludovic Chevalier, Stéphanie Sahuguede, and Anne Julien-Vergonjanne. Wireless optical technology based body area network for health monitoring application. In *Proceedings of IEEE ICC'15*, 2015.

[44] In-Koo Cho and David M Kreps. Signaling games and stable equilibria. *The Quarterly Journal of Economics*, 102(2):179–221, 1987.

[45] F. Ciuffo and G. M. Weiss. Smartwatch-based transcription biometrics. In *IEEE Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*, pages 145–149, 2017.

[46] Guglielmo Cola, Marco Avvenuti, Alessio Vecchio, Guang-Zhong Yang, and Benny Lo. An unsupervised approach for gait-based authentication. In *2015 IEEE 12th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*, pages 1–6. IEEE, 2015.

[47] Christian Collberg and Todd A Proebsting. Repeatability in computer systems research. *Communications of the ACM*, 59(3), 2016.

[48] Components101. L298N Motor Driver Module. https://components101.com/modules/l293n-motor-driver-module, 2020. [Online; accessed 15-Nov-2021].

[49] K. R. Corpus, R. J. DL Gonzales, A. Scott Morada, and L. A. Vea. Mobile user identification through authentication using keystroke dynamics and accelerometer biometrics. In *Proceedings of the International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, pages 11–12, 2016.

[50] M. Cote, F. Jean, A. B. Albu, and D. Capson. Video summarization for remote invigilation of online exams. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–9, 2016.

[51] A. De Luca, A. Hang, F. Brudy, C. Lindner, and H. Hussmann. Touch me once and i know it's you! implicit authentication based on touch screen patterns. In *ACM Conference on Human Factors in Computing Systems (CHI)*, pages 987–996, 2012.

[52] M. O Derawi, P. Bours, and K. Holien. Improved cycle detection for accelerometer based gait authentication. In *IEEE International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP)*, pages 312–317, 2010.

[53] Durai Rajan Dhatchayeny, Willy Anugrah Cahyadi, Shivani Rajendra Teli, and Yeon-Ho Chung. A novel optical body area network for transmission of multiple patient vital signs. In *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 542–544, 2017.

[54] B Dorizzi, R Cappelli, M Ferrara, D Maio, D Maltoni, N Houmani, S Garcia-Salicetti, and A Mayoue. Fingerprint and on-line signature verification competitions at icb 2009. In *International Conference on Biometrics (ICB)*, 2009.

[55] Dynex. Dynex - DX-PNC2019 Wireless Ergonomic Wireless Keyboard. https://www.dynexproducts.com/pdp/DX-PNC2019/6350929, 2023. [Online; accessed 15-Oct-2023].

[56] Antonino D'Alessandro, Salvatore Scudero, and Giovanni Vitale. A review of the capacitive mems for seismology. *Sensors*, 19(14):3093, 2019.

[57] Xiaoran Fan, Longfei Shangguan, Siddharth Rupavatharam, Yanyong Zhang, Jie Xiong, Yunfei Ma, and Richard Howard. Headfi: bringing intelligence to all headphones. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2021.

[58] A. L. Fantana, S. Ramachandran, C. H. Schunck, and M. Talamo. Movement based biometric authentication with smartphones. In *IEEE International Carnahan Conference on Security Technology (ICCST)*, pages 235–239, 2015.

[59] Adrienne Porter Felt, Erika Chin, Steve Hanna, Dawn Song, and David Wagner. Android permissions demystified. In *ACM Conference on Computer and Communications Security (CCS)*, 2011.

[60] Adrienne Porter Felt, Matthew Finifter, Erika Chin, Steve Hanna, and David Wagner. A survey of mobile malware in the wild. In *ACM SPSM*, 2011.

[61] L. J. Fennelly. *Effective Physical Security*. Elsevier Science, 2003.

[62] Laura Galluccio, Tommaso Melodia, Sergio Palazzo, and Giuseppe Enrico Santagati. Challenges and implications of using ultrasonic communications in intra-body area networks. In *Proceedings of IEEE WONS'12*, 2012.

[63] Robert Gao and Li Zhang. Micromachined microsensors for manufacturing. *IEEE Instrumentation & Measurement Magazine*, 7(2):20–26, 2004.

[64] Shuo Gao, Shuo Yan, Hang Zhao, and Arokia Nathan. *Touch-Based Human-Machine Interaction: Principles and Applications*, pages 91–108. Springer, 2021.

[65] Xianyi Gao, Bernhard Firner, Shridatt Sugrim, Victor Kaiser-Pendergrast, Yulong Yang, and Janne Lindqvist. Elastic pathing: You speed is enough to track you. In *ACM UbiComp*, 2014.

[66] Wolf Garbe. SymSpell. `https://github.com/wolfgarbe/SymSpell`, 2012. [Online; accessed 15-Oct-2023].

[67] Enrique Garcia-Ceja, Carlos E Galván-Tejada, and Ramon Brena. Multi-view stacking for activity recognition with sound and accelerometer data. *Elsevier Information Fusion*, 40, 2018.

[68] Dirk Goldhahn, Thomas Eckart, and Uwe Quasthoff. Building large monolingual dictionaries at the leipzig corpora collection: From 100 to 200 languages. In *Language Resources and Evaluation Conference*, volume 29, 2012.

[69] Google. Step detector sensor. `https://developer.android.com/reference/android/hardware/Sensor`, 2020. [Online; accessed 15-Nov-2021].

[70] Google. Google Pixel Buds requirements and specifications. `https://support.google.com/googlepixelbuds/answer/7544332`, 2022. [Online; accessed 15-Oct-2023].

[71] Google. Android Developers - Permissions. `https://developer.android.com/guide/topics/manifest/permission-element`, 2024. [Online; accessed 15-Jun-2024].

[72] Google. Google Pixel Buds. `https://play.google.com/store/apps/details?id=com.google.android.apps.wearables.maestro.companion`, 2024. [Online; accessed 15-Feb-2024].

[73] I. Griswold-Steiner, R. Matovu, and A. Serwadda. Handwriting watcher: A mechanism for smartwatch-driven handwriting authentication. In *IEEE International Joint Conference on Biometrics (IJCB)*, pages 216–224, 2017.

[74] I. Griswold-Steiner, R. Matovu, and A. Serwadda. Wearables-driven freeform handwriting authentication. *IEEE Transactions on Biometrics, Behavior, and Identity Science (T-BIOM)*, 1(3):152–164, 2019.

[75] Yu Guan and Thomas Plötz. Ensembles of deep lstm learners for activity recognition using wearables. *Proceedings of ACM IMWUT*, 1(2):1–28, 2017.

[76] K. Guk, G. Han, J. Lim, K. Jeong, T. Kang, E. Lim, and J. Jung. Evolution of wearable devices with real-time disease monitoring for personalized healthcare. *Nanomaterials*, 9(6):813, 2019.

[77] Mahmoud Hammad, Hamid Bagheri, and Sam Malek. Determination and enforcement of least-privilege architecture in android. In *IEEE ICSA*, 2017.

[78] J. Han, E. Owusu, L. Nguyen, A. Perrig, and J. Zhang. Accomplice: Location inference using accelerometers on smartphones. In *ACM COMSNETS*, 2012.

[79] M. Harbach, A. De Luca, and S. Egelman. The anatomy of smartphone unlocking: A field study of android lock screens. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 4806–4817, 2016.

[80] HP. HP ENVY x360. `https://support.hp.com/us-en/document/c06038609`, 2023. [Online; accessed 15-Oct-2023].

[81] Pengfei Hu, Hui Zhuang, Panneer Selvam Santhalingam, Riccardo Spolaor, Parth Pathak, Guoming Zhang, and Xiuzhen Cheng. Accear: Accelerometer acoustic eavesdropping with unconstrained vocabulary. In *IEEE Symposium on Security and Privacy (SP)*, 2022.

[82] C. Huang, Z. Yang, H. Chen, and Q. Zhang. Signing in the air w/o constraints: Robust gesture-based authentication for wrist wearables. In *IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2017.

[83] Huaiqi Huang, Tao Li, Christian Antfolk, Christian Enz, Jörn Justiz, and Volker M Koch. Experiment and investigation of two types of vibrotactile devices. In *Proceedings of IEEE BioRob'16*, 2016.

[84] Inhwan Hwang, Jungchan Cho, and Songhwai Oh. Privacy-aware communication for smartphones using vibration. In *Proceedings of IEEE RTCSA'12*, 2012.

[85] Inhwan Hwang, Jungchan Cho, and Songhwai Oh. Vibecomm: Radio-free wireless communication for smart devices using vibration. *Sensors*, 14(11):21151–21173, 2014.

[86] D. Impedovo and G. Pirlo. Automatic signature verification in the mobile cloud scenario: survey and way ahead. *IEEE Transactions on Emerging Topics in Computing*, 2018.

[87] InvenSense. MPU-6050 Six-Axis (Gyro + Accelerometer) MEMS MotionTracking™ Devices. `https://invensense.tdk.com/products/motion-tracking/6-axis/mpu-6050`, 2020. [Online; accessed 15-Nov-2021].

[88] Peter Ivie and Douglas Thain. Reproducibility in scientific computing. *ACM Computing Surveys (CSUR)*, 51(3), July 2018.

[89] Anne Julien-Vergonjanne, Stéphanie Sahuguède, and Ludovic Chevalier. *Optical Wireless Body Area Networks for Healthcare Applications*, pages 569–587. Springer International Publishing, Cham, 2016.

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

[91] N. A. Karim and Z. Shukur. Review of user authentication methods in online examination. *Asian Journal of Information Technology*, 14(5):166–175, 2015.

[92] KBE team. What Are Scissor Switch Keyboards. https://keyboardsexpert.com/what-are-scissor-switch-keyboards/, 2023. [Online; accessed 15-Oct-2023].

[93] M. Kheirkhahan, S. Nair, A. Davoudi, P. Rashidi, A. A. Wanigatunga, D. B. Corbett, T. Mendoza, T. M. Manini, and S. Ranka. A smartwatch-based framework for real-time and online assessment and mobility monitoring. *Elsevier Journal of Biomedical Informatics*, 89:29–40, 2019.

[94] Younghyun Kim, Woo Suk Lee, Vijay Raghunathan, Niraj K Jha, and Anand Raghunathan. Vibration-based secure side channel for medical devices. In *Proceedings of ACM/EDAC/IEEE DAC'15*, 2015.

[95] Diederik P. Kingma and Jimmy B. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[96] I. Kononenko. Estimating attributes: analysis and extensions of relief. In *European Conference on Machine Learning*, pages 171–182. Springer, 1994.

[97] Gierad Laput, Robert Xiao, and Chris Harrison. Viband: High-fidelity bio-acoustic sensing using commodity smartwatch accelerometers. In *ACM Annual Symposium on User Interface Software and Technology*, 2016.

[98] Kyuin Lee, Vijay Raghunathan, Anand Raghunathan, and Younghyun Kim. Syncvibe: Fast and secure device pairing through physical vibration on commodity smartphones. In *Proceedings of IEEE ICCD'18*, 2018.

[99] A. Levy, B. Nassi, Y. Elovici, and E. Shmueli. Handwritten signature verification using wrist-worn devices. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(3):1–26, 2018.

[100] S. Li, A. Ashok, Y. Zhang, C. Xu, J. Lindqvist, and M. Gruteser. Whose move is it anyway? authenticating smart wearable devices using unique head movement patterns. In *IEEE*

International Conference on Pervasive Computing and Communications (PerCom), pages 1–9, 2016.

[101] G. Liang, X. Xu, and J. Yu. User-authentication on wearable devices based on punch gesture biometrics. In *ITM Web Conf.*, volume 11, page 01003. EDP Sciences, 2017.

[102] Stefan Liebich, Johannes Fabry, Peter Jax, and Peter Vary. Signal processing challenges for active noise cancellation headphones. In *ITG-Symposium: Speech Communication*, 2018.

[103] Jian Liu, Yan Wang, Gorkem Kar, Yingying Chen, Jie Yang, and Marco Gruteser. Snooping keystrokes with mm-level audio ranging on a single phone. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2015.

[104] Xiangyu Liu, Zhe Zhou, Wenrui Diao, Zhou Li, and Kehuan Zhang. When good becomes evil: Keystroke inference with smartwatch. In *ACM Conference on Computer and Communications Security (CCS)*, 2015.

[105] Xiangyu Liu, Zhe Zhou, Wenrui Diao, Zhou Li, and Kehuan Zhang. When good becomes evil: Keystroke inference with smartwatch. In *Proceedings of the 2015 ACM SIGSAC conference on computer and communications security (CCS)*, 2015.

[106] Yu Liu, Cristina Comaniciu, and Hong Man. A bayesian game approach for intrusion detection in wireless ad hoc networks. In *ACM Workshop on Game theory for Communications and Networks*, page 4. ACM, 2006.

[107] James Llinas and David L Hall. An introduction to multi-sensor data fusion. In *Proceedings of the 1998 IEEE International Symposium on Circuits and Systems (ISCAS)*, 1998.

[108] Logitech. Keyboard K120. `https://www.logitech.com/en-us/products/keyboards/k120-usb-standard-computer.920-002478.html`, 2023. [Online; accessed 15-Oct-2023].

[109] C. X. Lu, B. Du, H. Wen, S. Wang, A. Markham, I. Martinovic, Y. Shen, and N. Trigoni. Snoopy: Sniffing your smartwatch passwords via deep sequence learning. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(4):1–29, 2018.

[110] D. Lu, D. Huang, Y. Deng, and A. Alshamrani. Multifactor user authentication with in-air-handwriting and hand geometry. In *IEEE International Conference on Biometrics (ICB)*, pages 255–262, 2018.

[111] D. Lu, K. Xu, and D. Huang. A data driven in-air-handwriting biometric authentication system. In *IEEE International Joint Conference on Biometrics (IJCB)*, 2017.

[112] Li Lu, Jiadi Yu, Yingying Chen, Yanmin Zhu, Xiangyu Xu, Guangtao Xue, and Minglu Li. Keylistener: Inferring keystrokes on qwerty keyboard of touch screen through acoustic signals. In *IEEE Conference on Computer Communications (INFOCOM)*, 2019.

[113] Pengfei Luo, Tong Jiang, Paul Anthony Haigh, Zabih Ghassemlooy, and Stanislav Zvanovec. Undersampled pulse width modulation for optical camera communications. In *Proceedings of IEEE ICC Workshops*, 2018.

[114] Dong Ma, Yuezhong Wu, Ming Ding, Mahbub Hassan, and Wen Hu. Skin-mimo: Vibration-based mimo communication over human skin. In *Proceedings of IEEE INFOCOM'20*, 2020.

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

[116] Anindya Maiti, Oscar Armbruster, Murtuza Jadliwala, and Jibo He. Smartwatch-based keystroke inference attacks and context-aware protection mechanisms. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security (AsiaCCS)*, 2016.

[117] Anindya Maiti, Ryan Heard, Mohd Sabra, and Murtuza Jadliwala. Towards inferring mechanical lock combinations using wrist-wearables as a side-channel. In *ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, 2018.

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

[119] Anindya Maiti, Murtuza Jadliwala, Jibo He, and Igor Bilogrevic. Side-channel inference attacks on mobile keypads using smartwatches. *IEEE Transactions on Mobile Computing*, 17(9), 2018.

[120] Louise R Manfredi, Andrew T Baker, Damian O Elias, John F Dammann III, Mark C Zielinski, Vicky S Polashock, and Sliman J Bensmaia. The effect of surface wave propagation on neural responses to vibration in primate glabrous skin. *PLOS ONE*, 7(2), 2012.

[121] J. Mantyjarvi, M. Lindholm, E. Vildjiounaite, S-M Makela, and HA Ailisto. Identifying users of portable devices from gait pattern with accelerometers. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, 2005.

[122] Xiaodong Mao and Noam Rimon. Method and apparatus for enhancing the generation of three-dimensional sound in headphone devices, 2012. US Patent 8,160,265.

[123] Philip Marquardt, Arunabh Verma, Henry Carter, and Patrick Traynor. (sp) iphone: Decoding vibrations from nearby keyboards using mobile phone accelerometers. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 551–562, 2011.

[124] Philip Marquardt, Arunabh Verma, Henry Carter, and Patrick Traynor. (sp)iphone: Decoding vibrations from nearby keyboards using mobile phone accelerometers. In *ACM Conference on Computer and Communications Security (CCS)*, 2011.

[125] Y. Michalevsky, D. Boneh, and G. Nakibly. Gyrophone: Recognizing speech from gyroscope signals. In *USENIX Security*, 2014.

[126] Yan Michalevsky, Dan Boneh, and Gabi Nakibly. Gyrophone: Recognizing speech from gyroscope signals. In *Proceedings of USENIX SECURITY'14*, 2014.

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

[128] Emiliano Miluzzo, Alexander Varshavsky, Suhrid Balakrishnan, and Romit Roy Choudhury. Tapprints: Your finger taps have fingerprints. In *ACM MobiSys*, 2012.

[129] Monoprice. Monoprice MP810 Optical Mechanical Gaming Keyboard. `www.monoprice.com/product?p_id=34563`, 2023. [Online; accessed 15-Oct-2023].

[130] Emmanuel Munguia Tapia. *Using machine learning for real-time activity recognition and estimation of energy expenditure*. PhD thesis, Massachusetts Institute of Technology, 2008.

[131] Sashank Narain, Amirali Sanatinia, and Guevara Noubir. Single-stroke language-agnostic keylogging using stereo-microphones and domain specific machine learning. In *Proceedings of the 2014 ACM conference on Security and privacy in wireless & mobile networks (WiSec)*, 2014.

[132] Sashank Narain, Triet D Vo-Huu, Kenneth Block, and Guevara Noubir. Inferring user routes and locations using zero-permission mobile sensors. In *IEEE S&P*, 2016.

[133] Krishna S Nathan, Homayoon SM Beigi, Jayashree Subrahmonia, Gregory J Clary, and Hiroshi Maruyama. Real-time on-line unconstrained handwriting recognition using statistical methods. *International Conference on Acoustics, Speech, and Signal Processing*, 4, 1995.

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

[135] T. Nguyen and N. D. Memon. Smartwatches locking methods: A comparative study. In *USENIX Symposium on Usable Privacy and Security (SOUPS)*, 2017.

[136] C. Nickel, T. Wirtl, and C. Busch. Authentication of smartphone users based on the way they walk using k-nn algorithm. In *IEEE International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pages 16–20, 2012.

[137] Oriol Nieto and Juan Pablo Bello. Systematic Exploration of Computational Music Structure Research. In *International Society for Music Information Retrieval Conference*, 2016.

[138] Jordan Novet. Why some techies are obsessed with mechanical computer keyboards — and how I learned to build my own. `https://www.cnbc.com/2020/01/12/why-mechanical-keyboards-are-becoming-more-popular-among-techies.html`, 2020. [Online; accessed 15-Oct-2023].

[139] U.S. Department of Health & Human Services. Coronavirus disease 2019 (covid-19) - cases, data, and surveillance, 2020. [Online; accessed 18-June-2020].

[140] Emmanuel Owusu, Jun Han, Sauvik Das, Adrian Perrig, and Joy Zhang. Accessory: Password inference using accelerometers on smartphones. In *ACM HotMobile*, 2012.

[141] Animesh Patcha and J-M Park. A game theoretic approach to modeling intrusion detection in mobile ad hoc networks. In *IEEE SMC Information Assurance Workshop*, pages 280–284, 2004.

[142] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[143] G. Peng, G.and Zhou, D. T. Nguyen, X. Qi, Q. Yang, and S. Wang. Continuous authentication with touch behavioral biometrics and voice on wearable glasses. *IEEE transactions on human-machine systems*, 47(3):404–416, 2016.

[144] Réjean Plamondon and Sargur N Srihari. Online and off-line handwriting recognition: a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1), 2000.

[145] Christopher Prest and Quin C Hoellwarth. Sports monitoring system for headphones, earbuds and/or headsets, 2014. US Patent 8,655,004.

[146] A. Primo, V. Phoha, R. Kumar, and A. Serwadda. Context-aware active authentication using smartphone accelerometer measurements. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 98–105, 2014.

[147] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2), 1989.

[148] Soundarya Ramesh, Thomas Pathier, and Jun Han. Sounduav: Towards delivery drone authentication via acoustic noise fingerprinting. In *Proceedings of the 5th Workshop on Micro Aerial Vehicle Networks, Systems, and Applications*, 2019.

[149] Trusted Reviews. Is your smart TV spying on you? All you need to know about smart TVs and your privacy. `https://www.trustedreviews.com/news/smart-tv-privacy-problems-vizio-samsung-lg-sony-panasonic-2952175`, 2020. [Online; accessed 15-Oct-2023].

[150] Jörn Rittweger. *Manual of Vibration Exercise and Vibration Therapy*. Springer, 2020.

[151] Tobias Röddiger, Christopher Clarke, Paula Breitling, Tim Schneegans, Haibin Zhao, Hans Gellersen, and Michael Beigl. Sensing with earables: A systematic literature review and taxonomy of phenomena. *Proceedings of the ACM on interactive, mobile, wearable and ubiquitous technologies (IMWUT)*, 6(3), 2022.

[152] Michèle Rombaut and Yue Min Zhu. Study of dempster–shafer theory for image segmentation applications. *Elsevier Image and Vision Computing*, 20(1), 2002.

[153] M. Roshandel, A. Munjal, P. Moghadam, S. Tajik, and H. Ketabdar. Multi-sensor finger ring for authentication based on 3d signatures. In *International Conference on Human-Computer Interaction*, pages 131–138. Springer, 2014.

[154] Joseph Roth, Xiaoming Liu, Arun Ross, and Dimitris Metaxas. Biometric authentication via keystroke sound. In *International Conference on Biometrics (ICB)*, 2013.

[155] EH Rothauser. Ieee recommended practice for speech quality measurements. *IEEE Trans. on Audio and Electroacoustics*, 17:225–246, 1969.

[156] Nirupam Roy and Romit Roy Choudhury. Ripple II: Faster communication through physical vibration. In *Proceedings of USENIX NSDI'16*, 2016.

[157] Nirupam Roy, Mahanth Gowda, and Romit Roy Choudhury. Ripple: Communicating through physical vibration. In *Proceedings of USENIX NSDI'15*, 2015.

[158] Michael Rushanan, Aviel D Rubin, Denis Foo Kune, and Colleen M Swanson. Sok: Security and privacy in implantable medical devices and body area networks. In *Proceedings of IEEE S&P'14*, 2014.

[159] Mohd Sabra, Anindya Maiti, and Murtuza Jadliwala. Keystroke inference using ambient light sensor on wrist-wearables: A feasibility study. In *ACM Workshop on Wearable Systems and Applications (WearSys)*, 2018.

[160] Mohd Sabra, Anindya Maiti, and Murtuza Jadliwala. Zoom on the keystrokes: Exploiting video calls for keystroke inference attacks. In *Network and Distributed Systems Security (NDSS) Symposium (NDSS)*, 2021.

[161] Samsung. Galaxy Buds Manager. `https://play.google.com/store/apps/details?id=com.samsung.accessory.fridaymgr`, 2024. [Online; accessed 15-Feb-2024].

[162] Jon R Sank. Microphones. In *Audio Engineering Society Conference: The Art and Technology of Recording*, 1984.

[163] R. Schlegel, K. Zhang, X. Zhou, M. Intwala, A. Kapadia, and X. Wang. Soundcomber: A stealthy and context-aware sound trojan for smartphones. In *NDSS*, 2011.

[164] Matthias Schulz, Adrian Loch, and Matthias Hollick. Practical known-plaintext attacks against physical layer security in wireless mimo systems. In *NDSS*, 2014.

[165] D. Schurmann, A. Brusch, S. Sigg, and L. Wolf. Bandana body area network device-to-device authentication using natural gait. In *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 190–196, 2017.

[166] Caitlyn Seim, James Hallam, Shashank Raghu, Tri-An Le, Greg Bishop, and Thad Starner. Perception in hand-worn haptics: Placement, simultaneous stimuli, and vibration motor comparisons. Technical report, Georgia Institute of Technology, 2015.

[167] Sougata Sen and David Kotz. Vibering: Using vibrations from a smart ring as an out-of-band channel for sharing secret keys. In *Proceedings of IEEE IoT'20*, 2020.

[168] Skyler Seto, Wenyu Zhang, and Yichen Zhou. Multivariate time series classification using dynamic time warping template selection for human activity recognition. *arXiv:1512.06747*, 2015.

[169] Valay A Shah, Maura Casadio, Robert A Scheidt, and Leigh A Mrotek. Vibration propagation on the skin of the arm. *Applied Sciences*, 9(20):4329, 2019.

[170] Haiyong Shi, Zhenjie Hou, Jiuzhen Liang, En Lin, and Zhuokun Zhong. Dsfnet: A distributed sensors fusion network for action recognition. *IEEE Sensors*, 23(1), 2022.

[171] R. Shilkrot, J. Huber, J. Steimle, S. Nanayakkara, and P. Maes. Digital digits: A comprehensive survey of finger augmentation devices. *ACM Computing Surveys (CSUR)*, 48(2):1–29, 2015.

[172] Shimmer. Shimmer Sensing. http://www.shimmersensing.com, 2019. Online; accessed 2019-01-25.

[173] P. Shinde, S. Shetty, and M. Mehra. Survey of keystroke dynamics as a biometric for static authentication. *International Journal of Computer Science and Information Security*, 14(4):203, 2016.

[174] Muhammad Shoaib, Stephan Bosch, Ozlem Durmaz Incel, Hans Scholten, and Paul JM Havinga. Complex human activity recognition using smartphone and wrist-worn motion sensors. *Sensors*, 16(4):426, 2016.

[175] Simone Soderi. Acoustic-based security: A key enabling technology for wireless sensor networks. *International Journal of Wireless Information Networks*, 27(1):45–59, 2020.

[176] Y. Song, Z. Cai, and Z. Zhang. Multi-touch authentication using hand geometry and behavioral information. In *IEEE Symposium on Security and Privacy (S&P)*, pages 357–372, 2017.

[177] STREAMS. Streamz Voice Controlled Headphones. https://www.streamzmedia.com, 2024. [Online; accessed 15-Jun-2024].

[178] Ahren Studer, Timothy Passaro, and Lujo Bauer. Don't bump, shake on it: The exploitation of a popular accelerometer-based smart phone exchange and its secure replacement. In *Proceedings of the 27th Annual Computer Security Applications Conference*, pages 333–342, 2011.

[179] TDK. MPU-6500 Six-Axis (Gyro + Accelerometer) MEMS. https://invensense.tdk.com/products/motion-tracking/6-axis/mpu-6500/, 2023. [Online; accessed 15-Oct-2023].

[180] technavio. Mechanical Keyboard Market by Distribution Channel, Type, and and Geography - Forecast and Analysis 2023-2027. https://www.technavio.com/report/mec

hanical-keyboard-market-industry-analysis, 2023. [Online; accessed 15-Oct-2023].

[181] Tecknet. TECKNET 2.4G Wireless Keyboard, Ultra Slim Compact Computer Keyboard. https://tecknet.com/products/tecknet-2-4g-wireless-quiet-keyboard, 2023. [Online; accessed 15-Oct-2023].

[182] Maria Temming. Smartphones Put Your Privacy At Risk. https://www.commonlit.org/en/texts/smartphones-put-your-privacy-at-risk, 2018. [Online; accessed 13-Apr-2024].

[183] Jeffrey J Terlizzi. Systems and methods for noise cancellation and power management in a wireless headset, 2012. US Patent 8,285,208.

[184] TheVerge. Anker finally comes clean about its Eufy security cameras. https://www.theverge.com/23573362/anker-eufy-security-camera-answers-encryption, 2023. [Online; accessed 15-Oct-2023].

[185] Panagiotis Tzirakis, Stefanos Zafeiriou, and Björn Schuller. Real-world automatic continuous affect recognition from audiovisual signals. In *Multimodal Behavior Analysis in the Wild*. Elsevier, 2019.

[186] T. T. Um, F. MJ. Pfister, D. Pichler, S. Endo, M. Lang, S. Hirche, U. Fietzek, and D. Kulić. Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks. In *ACM International Conference on Multimodal Interaction*, pages 216–220, 2017.

[187] Marcus Varanis, Anderson Silva, Arthur Mereles, and Robson Pederiva. Mems accelerometers for mechanical vibrations analysis: A comprehensive review with applications. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 40(11):1–18, 2018.

[188] Anita Venkatanarayanan and Elaine Spain. 13.03 - review of recent developments in sensing materials. In *Comprehensive Materials Processing*, pages 47–101. Elsevier, 2014.

[189] Chen Wang, Xiaonan Guo, Yan Wang, Yingying Chen, and Bo Liu. Friend or Foe?: Your Wearable Devices Reveal Your Personal Pin. In *ACM Asia Conference on Computer and Communications Security (AsiaCCS)*, pages 189–200, 2016.

[190] Chen Wang, Xiaonan Guo, Yan Wang, Yingying Chen, and Bo Liu. Friend or foe?: Your wearable devices reveal your personal pin. In *ACM Asia Conference on Computer and Communications Security (AsiaCCS)*, 2016.

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

[192] He Wang, Ted Tsung-Te Lai, and Romit Roy Choudhury. Mole: Motion leaks through smartwatch sensors. In *Proceedings of the 21st annual international conference on mobile computing and networking (MobiCom)*, 2015.

[193] J. Wang, Y. Hsu, and J. Liu. An inertial-measurement-unit-based pen with a trajectory reconstruction algorithm and its applications. *IEEE Transactions on Industrial Electronics*, 57(10):3508–3521, 2009.

[194] Jian Wang, Rukhsana Ruby, Lu Wang, and Kaishun Wu. Accurate combined keystrokes detection using acoustic signals. In *12th IEEE International Conference on Mobile Ad-Hoc and Sensor Networks MSN*, 2016.

[195] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys*, 53(3), 2020.

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

[197] WordFrequency. Introduction to the top 60,000 words in English. `https://www.wordfrequency.info/intro.asp`, 2023. [Online; accessed 15-Oct-2023].

[198] Qingxin Xia, Feng Hong, Yuan Feng, and Zhongwen Guo. Motionhacker: Motion sensor based eavesdropping on handwriting via smartwatch. In *IEEE Conference on Computer Communications Workshops (INFOCOM Workshops)*, pages 468–473, 2018.

[199] Chao Xu, Parth H Pathak, and Prasant Mohapatra. Finger-writing with smartwatch: A case for finger and hand gesture recognition using smartwatch. In *ACM Workshop on Mobile Computing Systems and Applications (HotMobile)*, pages 9–14, 2015.

[200] Zhi Xu, Kun Bai, and Sencun Zhu. Taplogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors. In *Proceedings of the 5th ACM conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, 2012.

[201] J. Yang, Y. Li, and M. Xie. Motionauth: Motion-based authentication for wrist worn smart devices. In *IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pages 550–555, 2015.

[202] W. Yang, S. Wang, J. Hu, G. Zheng, and C. Valli. Security and accuracy of fingerprint-based biometrics: A review. *Symmetry*, 11(2):141, 2019.

[203] Takuro Yonezawa, Tomotaka Ito, and Hideyuki Tokuda. Transferring information from mobile devices to personal computers by using vibration and accelerometer. In *Proceedings of ACM UbiComp'11*, 2011.

[204] Takuro Yonezawa, Hiroshi Nakahara, and Hideyuki Tokuda. Vib-connect: A device collaboration interface using vibration. In *Proceedings of IEEE RTCSA'11*, 2011.

[205] Zhiyuan Yu, Zack Kaplan, Qiben Yan, and Ning Zhang. Security and privacy in the emerging cyber-physical world: A survey. *IEEE Communications Surveys & Tutorials*, 23(3), 2021.

[206] Cheng Zhang, Sinan Hersek, Yiming Pu, Danrui Sun, Qiuyue Xue, Thad E Starner, Gregory D Abowd, and Omer T Inan. Bioacoustics-based human-body-mediated communication. *Computer*, 50(2):36–46, 2017.

[207] Liang Zhang, Yongyuan Qin, and Jinliang Zhang. Study of polynomial curve fitting algorithm for outlier elimination. In *IEEE International Conference on Computer Science and Service System (CSSS)*, 2011.

[208] Q. Zhang, H. Li, Z. Sun, and T. Tan. Deep feature fusion for iris and periocular biometrics on mobile devices. *IEEE Transactions on Information Forensics and Security*, 13(11):2897–2912, 2018.

[209] Yue Zhang, Jian Weng, Rajib Dey, Yier Jin, Zhiqiang Lin, and Xinwen Fu. Breaking secure pairing of bluetooth low energy using downgrade attacks. In *29th USENIX Security Symposium (USENIX Security 14)*, pages 37–54, 2020.

[210] Tong Zhu, Qiang Ma, Shanfeng Zhang, and Yunhao Liu. Context-free attacks using keyboard acoustic emanations. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security (CCS)*, 2014.

VITA

Raveen Wijewickrama originates from Sri Lanka. He earned his Bachelor of Science in Computer Science from the Asia Pacific Institute of Information Technology (APIIT), Sri Lanka, in 2015, followed by a Master of Science in Computer Science from Wichita State University, Kansas, USA, in 2017. His research interests lie in the areas of privacy and security, mobile sensing, and micromobility. In his free time and often when he is stressed, Raveen enjoys baking, an activity he refers to as "stress baking". Looking ahead, he plans to move to a colder climate, hoping that the grass will indeed be greener on the other side.