UNDERSTANDING AND MITIGATING THE PRIVACY RISKS OF SMARTPHONE SENSOR FINGERPRINTING

BY

ANUPAM DAS

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2016

Urbana, Illinois

Doctoral Committee:

Associate Professor Nikita Borisov, Chair
Associate Professor Matthew Caesar
Associate Professor Romit Roy Choudhury
Assistant Professor Paris Smaragdis
Professor Dan Boneh, Stanford University

ABSTRACT

The widespread use of smartphones in our everyday life gives rise to privacy concerns. Fingerprinting smartphones can jeopardize user privacy by enabling remote identification of users without users' awareness. In this dissertation we study the feasibility of using onboard sensors such as microphones, accelerometers and gyroscopes to fingerprint smartphones. During fabrication, subtle imperfections arise in device sensors which induce distinctive anomalies in the generated signal. Using machine learning techniques we can distinguish smartphones generating such distinctive anomalies.

We first look at fingerprinting smartphones through onboard microphones and speakers. We explore different acoustic features and analyze their ability to successfully fingerprint smartphones. Our study identifies the prominent acoustic features capable of fingerprinting smartphones with a high success rate, and also examines the impact of background noise and other variables on fingerprinting accuracy. Next, we surreptitiously fingerprint smartphones using the imperfections of motion sensors (i.e., accelerometers and gyroscopes) embedded in modern smartphones, through a web page. We analyze how well motion sensor fingerprinting works under real-world constraints by collecting data from a large number of smartphones under both lab and public environments. Our study demonstrates that motion sensor fingerprinting is effective even with 500 users. We also develop a model to estimate prediction accuracy for larger user populations; our model provides a conservative estimate of at least 10% classification accuracy with 100 000 users, which suggests that motion sensor fingerprinting can be effective when combined with even a weak browser fingerprint. We then investigate the use of motion sensors on the web and find, distressingly, that many sites send motion sensor data to servers for storage and analysis, paving the way for potential fingerprinting. Finally, we consider the problem of developing countermeasures for motion sensor fingerprinting; we propose several practical countermeasures

and evaluate their usability through a large-scale user study. We find that countermeasures such as data obfuscation and sensor quantization are really promising in the sense that they not only drastically reduce fingerprinting accuracy but also remain benign to applications using motion sensors.

To my family, for their love and support.

ACKNOWLEDGMENTS

First and foremost I thank my advisor, Nikita Borisov, whose support and guidance made this dissertation work possible. It has been an honor to have spent the last six years under his tutelage. I appreciate the many hours of discussion and brainstorming we had all of which has made my Ph.D. experience productive and stimulating. I specially thank him for providing me with the freedom to grow as a researcher. His advice on both research as well as on my career has been invaluable to me.

I would also like to thank Matthew Caesar who has been like a second advisor to me. He has been really supportive and has always placed a high level of trust on my judgment to pursue various projects. He has also provided insightful directions regarding my research and helped me acquire funds to carry out my research projects. I also have to thank my other committee members, Romit Roy Choudhury, Paris Smaragdis and Dan Boneh for their constructive advice and suggestion; with special thanks to Romit for providing me with many of the smartphones used in my experiments. The work that follows would not be possible without their valuable feedback.

Several people have helped and shared their knowledge with me at the University of Illinois. Firstly, I want to thank the present and past members of the Hatswitch research group: Amir Houmansadr, Sonia Jahid, Qiyan Wang, Joshua Juen, Giang Nyugen and Xun Gong for providing a stimulating yet enjoyable research environment. I would like to specially thanks Prateek Mittal, a former member of Hatswitch research group, for his valuable advice and guidance regarding both research and career path choice. Also, I would like to thank Edward Chou and Muhammad Haris Mughees for their help with data collection. Secondly, I would like to thank Carol Wisniewski for handling and managing all my travel and reimbursement related paperworks to make my life easier. Her help meant I could seamlessly concentrate on my research. Lastly, I would like to extend my appreciation to all the members of

Bangladesh Student Association (BSA) group at the University of Illinois. I was lucky to have so many fellow countrymen near me throughout my Ph.D. life. They not only acted as a source of stress relief for me but also many of them actively participated in my user studies. I am grateful to them for their participation.

I would like to take this opportunity to thanks all my teachers and advisers at Bangladesh University of Engineering and Technology (BUET) for their support and effort to prepare students like myself for higher studies. The strong undergraduate curriculum at BUET had prepared me for the hard road ahead. I am ever indebted to BUET for providing me with the opportunity to not only learn from the best minds of the country, but also mentor the best students of our country.

Last but not least, everything I have done is only possible due to the love and support from my family. I would like to thank my father and mother for their unconditional support and belief as I pursued my education. My father who is a professor himself has been my role model from childhood. A special thanks goes to my mother who has sacrificed her career for me and my sister. Without her care and attention to detail we would not have made it this far. My sister has been my good friend all my life and I thank her for all her mental support and advice when times were tough. Also, I would like to thank my newborn niece who cheers up my day with her smiling face. Finally, special thanks to the newest member of my family, Toma, my wife as well as her wonderful family who all have been supportive and caring. The best outcome from the past one year of my life is finding Toma, my best friend and soul-mate. Toma has been a true supporter and has unconditionally loved me during my good and bad times. She has been non-judgmental of me and instrumental in instilling confidence. She has faith in me and my intellect even when I doubt myself. The last few months have been really crucial for me as I have been deciding my future career path. I thank Toma for nudging me in the right direction.

Thanks to all of my close family and friends who have supported and encouraged me throughout the years. I dedicate this dissertation to those who believed in me and to God above who has provided me with this great opportunity.

TABLE OF CONTENTS

LIST OF TABLES xi

LIST OF FIGURES xiii

LIST OF ALGORITHMS xvii

CHAPTER 1 INTRODUCTION 1
1.1 Motivation 1
1.2 Contributions 3
1.3 Dissertation Outline 5
1.4 Collaborators and Published Works 7

CHAPTER 2 BACKGROUND AND RELATED WORK 9
2.1 Acoustic Hardware 9
2.1.1 Microphone 9
2.1.2 Microspeaker 11
2.2 Motion Sensors 12
2.2.1 Accelerometer 13
2.2.2 Gyroscope 13
2.3 Related Work 15
2.3.1 Browser Fingerprinting 15
2.3.2 Microphone and Speaker Fingerprinting 17
2.3.3 Motion Sensor Fingerprinting 18

CHAPTER 3 FINGERPRINTING SMARTPHONES VIA MICROPHONES AND SPEAKERS 20
3.1 Overview 20
3.2 Methodology 21
3.2.1 Procedure for Fingerprinting Speakers 22
3.2.2 Procedure for Fingerprinting Microphones 22
3.2.3 Procedure for Fingerprinting both Speakers and Microphones 23
3.3 Experimental Setup 24
3.3.1 Device Types 24
3.3.2 Audio Genre Types 24

3.3.3 Analytic Tools . . . 25
3.4 Acoustic Features . . . 25
3.5 Classification Algorithms and Evaluation Metrics . . . 31
3.5.1 Classification Algorithms . . . 31
3.5.2 Evaluation Metrics: . . . 32
3.6 Feature Exploration . . . 33
3.6.1 Feature Exploration for Different Make and Model . . . 35
3.6.2 Feature Exploration for Same Make and Model . . . 36
3.6.3 Feature Exploration for Large Pool of Devices . . . 38
3.7 Experimental Evaluations . . . 39
3.7.1 Fingerprinting Different Make and Model Devices . . . 40
3.7.2 Fingerprinting Same Make and Model Devices . . . 41
3.7.3 Fingerprinting All Make and Model Devices . . . 42
3.8 Sensitivity Analysis . . . 44
3.8.1 Impact of Sampling Rate . . . 44
3.8.2 Impact Training Set Size . . . 45
3.8.3 Impact of Distance between Speaker and Recorder . . . 45
3.8.4 Impact of Ambient Background Noise . . . 46
3.9 Limitations . . . 48
3.10 Summary . . . 48

CHAPTER 4 FINGERPRINTING SMARTPHONES VIA MOTION SENSORS . . . 50
4.1 Overview . . . 50
4.2 Data Collection Setup and Data Processing . . . 51
4.2.1 Data Collection Setup . . . 51
4.2.2 Data Processing . . . 54
4.3 Temporal and Spectral Features . . . 55
4.4 Classification Algorithms and Evaluation Metrics . . . 57
4.4.1 Classification Algorithms . . . 58
4.4.2 Evaluation Metrics . . . 58
4.5 Experimental Setup . . . 59
4.5.1 Lab Setting . . . 59
4.5.2 Public Setting . . . 59
4.5.3 Analytic Tools . . . 60
4.6 Feature Exploration . . . 60
4.7 Experimental Evaluations . . . 61
4.7.1 Results from Lab Setting . . . 62
4.7.2 Results from Public Setting . . . 63
4.7.3 Results from Combined Setting . . . 63
4.8 Sensitivity Analysis . . . 64
4.8.1 Impact of Device Number . . . 64
4.8.2 Impact of Training Set Size . . . 64
4.8.3 Impact of Temperature . . . 65

4.8.4 Temporal Stability 67
4.9 Summary 68

CHAPTER 5 LARGE-SCALE MOTION SENSOR FINGERPRINTING 69
5.1 Overview 69
5.2 Data Collection 69
5.3 Classification Algorithms and Metrics 71
5.4 Fingerprinting Large Number of Smartphones 71
5.5 Large-Scale Simulation 72
5.5.1 Distance Metric Learning 72
5.5.2 Intra- and Inter-Device Distance Modeling 74
5.5.3 Simulating k-NN for Large Number of Devices 77
5.6 Summary 80

CHAPTER 6 USAGE PATTERNS OF MOTION SENSORS IN THE WILD 82
6.1 Data Collection Framework 82
6.2 Feature Extraction 82
6.3 Measurement Study Results 83
6.3.1 Websites Accessing Motion Sensors 84
6.3.2 Types of Usage for Motion Sensors 84
6.4 Summary 87

CHAPTER 7 COUNTERMEASURES FOR MOTION SENSOR FINGERPRINTING 88
7.1 Sensor Calibration 88
7.1.1 Calibrating Accelerometers 89
7.1.2 Calibrating Gyroscopes 91
7.1.3 Fingerprinting Calibrated Data 93
7.2 Data Obfuscation 94
7.2.1 Uniform Noise 95
7.2.2 Laplace Noise 99
7.2.3 White Noise 101
7.3 Sensor Quantization 102
7.3.1 Fingerprinting Quantized Data 104
7.4 Determining Feasible Countermeasures 105
7.4.1 Utility under Calibration 105
7.4.2 Utility under Obfuscation 106
7.4.3 Utility under Quantization 108
7.4.4 Deployment Considerations 109
7.5 Effectiveness of Countermeasures at Large-Scale 109
7.6 Large-Scale User Study of Privacy vs. Utility 111
7.6.1 Study Design 111
7.6.2 Study Results 113

7.6.3 Limitations 114
7.7 Summary 116

CHAPTER 8 CONCLUSION 118

REFERENCES 121

APPENDIX A FEATURE DISTRIBUTIONS FOR MOTION SENSOR DATA 136

APPENDIX B DEVICE MODELS IN OUR DATASET 138

APPENDIX C DEVICE MODELS IN OUR USER STUDY 139

LIST OF TABLES

2.1 Comparison with other motion sensor fingerprinting studies. . 19

3.1 Types of phones used for fingerprinting acoustic components. . 24
3.2 Types of audio excerpts used in our experiments. 25
3.3 Explored acoustic features. 26
3.4 Feature exploration for different make and model smartphones using only speaker. 36
3.5 Feature exploration for different make and model smartphones using only microphone. 37
3.6 Feature exploration for different make and model smartphones using both speaker and microphone. 38
3.7 Feature exploration for same make and model smartphones using only speaker. 39
3.8 Feature exploration for same make and model smartphones using only microphone. 40
3.9 Feature exploration for same make and model smartphones using both speaker and microphone. 41
3.10 Feature exploration for 50 android smartphones. 43
3.11 Fingerprinting different make and model devices. 43
3.12 Fingerprinting same make and model devices. 43
3.13 Fingerprinting heterogeneous devices. 44
3.14 Impact of ambient background noise. 48

4.1 Sampling frequency from different browsers. 53
4.2 Types of background audio stimulation. 54
4.3 Explored temporal and spectral features. 56
4.4 Types of lab phones used. 59
4.5 Average F-score under lab setting. 63
4.6 Average F-score under public setting where smartphones are kept *on top of a desk*. 63
4.7 Average F-score under both lab and public setting where smartphones are kept *on top of a desk*. 64
4.8 Types of phones used for analyzing temperature effect. 66
4.9 Impact of temperature on motion sensor fingerprinting. 67
4.10 Fingerprinting sensors at different dates. 67

5.1 Average F-score for different size of training set. 72
5.2 Performance of different metric learning algorithms. 74
5.3 Average F-score of k-NN after LDML. 75

6.1 Top websites accessing motion sensors. 84
6.2 Silhouette coefficient for different number of clusters. 85
6.3 Generic use cases for accessing motion sensor data. 87
6.4 Description of use cases for accessing motion sensor data. . . . 87

7.1 Average F-score for calibrated data under lab setting. 95
7.2 Average F-score for obfuscated data under lab setting. 96
7.3 Average F-score for obfuscated data under public setting (63 phones) where smartphones were kept *on top of a desk*. . . 97
7.4 Average F-score for obfuscated data under both lab and public setting (93 phones) where smartphones were kept *on top of a desk*. 97
7.5 Privacy vs. Utility tradeoff for different countermeasures. . . . 107
7.6 Comparing obfuscation and quantization with baseline for 545 devices. 110
7.7 Number of users that completed the first n levels recruited through Amazon's Mechanical Turk and other means. 113

LIST OF FIGURES

2.1 The internal architecture of a MEMS microphone chip used in modern smartphones. 10
2.2 (a) The basic components of a speaker, (b) A typical MEMS microspeaker, (c) The internal architecture of a microspeaker chip. 11
2.3 Internal architecture of a MEMS accelerometer. Differential capacitance is proportional to the applied acceleration. . . 14
2.4 MEMS-based gyros use *Coriolis force* to compute angular velocity. The Coriolis force induces change in capacitance which is proportional to the angular velocity. 14

3.1 Fingerprinting smartphone speakers in public location. 21
3.2 Fingerprinting smartphone microphones in public location. . . 21
3.3 Fingerprinting both smartphone speakers and microphones. . . 21
3.4 Steps for fingerprinting speakers. 22
3.5 Steps for fingerprinting microphones. 23
3.6 Steps for fingerprinting both microphone and speaker. 23
3.7 Procedure for extracting MFCCs from audio signals. 30
3.8 MFCCs for a given audio sample taken from three different handsets manufactured by the same vendor. We can see that some of the coefficients vary significantly, thus enabling us to exploit this feature to fingerprint smartphones. . . 42
3.9 Impact of sampling frequency on precision/recall. 45
3.10 Impact of varying training set size on accuracy. 46
3.11 Impact of varying the distance between smartphone and microphone. 47
3.12 Experimental setup for determining the impact of ambient background noise. 47

4.1 Fingerprinting motion sensors through HTML5. 51
4.2 Screenshots of our data collection website. Users are first asked to place the device on a flat surface before selecting a specific background audio-stimulation. 54
4.3 Distribution of participant device models. 60

4.4 Exploring the number optimal features for different sensors. a) For accelerometer using more than the top 10 features leads to diminished returns, b) For gyroscope all 75 features contribute to obtaining improved accuracy, c) For the combined sensor stream using more than 70 features leads to diminished returns. 62
4.5 Average F-score for different numbers of smartphones. F-score generally tends to decrease slightly as more devices are considered. 65
4.6 Average F-score for different ratio of training and testing data. With only two training data we achieved an F-score of 98%. 66

5.1 Distribution of the number of data samples per smartphone. . 70
5.2 Comparing mutual information for different metric learning algorithms. Mutual information per feature for (a) untransformed data (b) LMNN transformation (c) ITML transformation, and (d) LDML transformation. 74
5.3 Estimated inter-device distance distributions for 4 subsets of devices where each subset contains 141 devices. 76
5.4 Estimated intra-device distance distributions for 4 subsets of devices where each subset contains 141 devices. 77
5.5 Estimated distributions for (a) inter-device distance (C_{inter}) (b) intra-device distance (C_{intra}). (c) Difference between intra- and inter-device distance distribution. 78
5.6 Comparing real world results with simulation results. Simulation results closely match real world results for $k = 1$. . . . 81

6.1 Overview of our JavaScript analysis setup. 83
6.2 Scatter plot for JavaScript snippets accessing motion sensors along reduced dimensions. 85
6.3 Silhouette plot for the estimated 8 clusters. 86

7.1 Calibrating accelerometer along three axes. We collect measurements along all 6 directions ($\pm x, \pm y, \pm z$). 90
7.2 Accelerometer offset and gain error from 30 smartphones. . . . 91
7.3 a) Offset and gain error in gyroscope impact systems that use them for angular-displacement measurements, b) Calibrating the gyroscope by rotating the device 180° in the positive x-axis direction. 92
7.4 Gyroscope offset and gain error from 30 smartphones. 94
7.5 Impact of obfuscation range as the range is linearly scaled up from 1x to 10x of the base range. 98
7.6 Impact of randomly inserting new data points. 99

7.7 Approximation of the probability of correct classification under differential privacy approach where noise is modeled through only offset and gain errors along all three axes for both accelerometer and gyroscope. 100
7.8 Impact of randomly selecting offset and gain error from a Laplace distribution inspired by differential privacy. 101
7.9 Impact of Gaussian white noise on F-score. 102
7.10 Convertion from Cartesian coordinate system (x, y, z) to Polar coordinate system (r, θ, ϕ). 103
7.11 Impact of sensor quantization on F-score. 104
7.12 Accelerometer magnitude for different mitigation schemes. . . 106
7.13 Accelerometer magnitude after removing disruptive countermeasures. 106
7.14 Impact of Laplace noise on utility. 108
7.15 Impact of Gaussian white noise on sensor utility. 108
7.16 Comparing large-scale classification accuracy for obfuscation and quantization. 110
7.17 Game interface. The object is to roll the ball to the flag while avoiding traps by tilting the smartphone. The user is then asked for feedback about the relative difficulty of each level using different privacy settings. 112
7.18 Subjective and objective difficulty metrics increase across game levels. Box plots show the median (horizontal line) and the interquartile range (box), while whiskers show the range between the 5th and 95th percentile, with the outliers being individually represented. The notch in the box denotes the 95% confidence interval around the median. 114
7.19 Game durations and number of restarts, as each level is played three times. A large training effect is observed between the first and second attempt, with a smaller effect between the second and third. 115
7.20 Impact of privacy method on subjective and objective ratings, when considering second and third attempts only. Shown are the histogram of subjective ratings and CDFs of game durations and number of restarts for all 5 levels. No significant difference found in any of the metrics. 116

A.1 Distributions for the top 12 original features. 136
A.2 Distributions for the top 12 features selected based on *JMI* criterion after LDML transformation is performed. 137

B.1 Distribution of the different make and model of smartphones that provided sensor data for our study. 138

C.1 Distribution of the different make and model of smartphones that participated in our user study. 139

LIST OF ALGORITHMS

1 Sequential Feature Selection (SFS). 35
2 Simulating k-NN classifier. 79
3 Obfuscated Data Injection. 98

CHAPTER 1

INTRODUCTION

The world is more connected than ever before and smartphones are making it easy for users to stay connected to the world around them. According to new figures from eMarketer the number of smartphone users worldwide will surpass 2 billion in 2016 [1]. The rapid uptake of intelligent smartphones is not surprising, due to the numerous advantages they provide to consumers, from entertainment and social applications to business and advanced computing capabilities. As a result smartphones have unprecedented access to sensitive personal information, and impose threatening concerns for user privacy [2, 3, 4, 5, 6, 7].

1.1 Motivation

Smartphones have made it easy for users to access online services and information from anywhere at anytime. However, such seamless connectivity with the web has made it lucrative for advertisers to track users' activities at different websites. Traditionally, advertisers implant *cookies* to build consumer profiles on users by tracking a user's surfing history without informing the user. Such privacy-invasive tracking raised concerns in the way cookies were being used and as a result legislation was passed to make changes to the rules on using cookies [8]. This prompted browser developers to provide ways to clear cookies, and also provide options to browse in private modes which do not store long-term cookies. Moreover, around 2010 the "Do Not Track" policy was proposed, which enabled users to opt out of tracking by websites they do not visit, including analytics services, advertising networks, and social platforms [9]. However, even though major online advertising trade groups initially pledged to support "Do Not Track" that promise still remains unfulfilled. And even with "Do Not Track" enabled most websites

in the wild were found to not honor the user's preference [10]. So after the failure of the "Do Not Track" proposal, users have increasingly started using tools such as ad- and tracker-blocking extensions, as well as private browsing modes, to protect their privacy.

In turn, advertisers have started fingerprinting user devices through the *browser* [11] to track users across the web without the use of cookies. A device fingerprint is a set of system attributes that, with high likelihood, uniquely characterizes a device. These attributes generally include, for example, the device's screen size, the versions of installed softwares, and the list of installed fonts. Attributes that are more diverse and stable (e.g., the list of fonts) facilitate better identification compared to those that are more common or unpredictable. Such stateless user tracking allows advertising companies to overcome the restrictions imposed by regulation on cookies. Thus, browser-based fingerprinting raises serious privacy concerns for everyday users because its stateless nature makes it hard to detect and even harder to opt-out. Moreover, this fingerprinting technique works just as well in the private browsing mode. To make things worse, in recent years researchers have come up with a more advanced technique that uses HTML5 *canvas elements* to fingerprint the fonts and rendering engines used by the browser [12]. Many studies have shown that all of these techniques are actually used in the wild [13, 10, 14].

With the advent of smartphones the battle for user privacy is shifting towards mobile platforms, which are quickly becoming the dominant mode for web browsing [15, 16, 17, 18]. Although existing fingerprinting techniques become less effective on mobile platforms due to constrained hardware and software environment [19, 20]; modern smartphones open door to new threats as they are equipped with a wide variety of sensors such as microphones, accelerometers and gyroscopes all of which are accessible to applications and websites for a variety of novel uses. These sensors can be exploited to threaten user privacy by enabling *sensor fingerprinting*. During manufacturing, imperfections are introduced in the analog circuitry of these sensors, and as such, two sensors never produce the same signal. This dissertation conveys the following statement–

"*It is feasible to fingerprint smartphones by exploiting the manufacturing imperfections of onboard sensors. There are also ways to mitigate some of these sensor fingerprinting techniques.*"

1.2 Contributions

In this dissertation, we study how acoustic hardware such as microphones and speakers, and motion sensors such as accelerometers and gyroscopes can to utilized to fingerprint smartphones under realistic settings. First, we look at how these sensors can be exploited as *side-channels* to enable an advertiser to fingerprint smartphones. Second, we focus on deriving countermeasures for motion sensor fingerprinting as accessing these sensors require no explicit user permission, whereas accessing microphone requires explicit user permission. Next, we perform real world web scanning to see how websites are accessing motion sensors. Lastly, through user study we look at how our proposed countermeasures impact the utility of the motion sensors.

This dissertation makes the following major contributions:

- **Fingerprinting Onboard Sensors in Smartphones:** We explore ways to fingerprint smartphones through embedded sensors. We look at acoustic sensors like microphones (and speakers), and also investigate motion sensors like accelerometers and gyroscopes.

  - **Fingerprinting Acoustic Hardware**: We exploit hardware-level imperfections in speakers and microphones to uniquely distinguish smartphones. Manufacturing process introduces subtle imperfections into the analog circuitry of these components, and as such the audio streams produced by two speakers or received by two microphones are never alike. Through an observational study, we find that these imperfections are substantial and prevalent enough that we can reliably fingerprint devices by conducting spectral analysis on recorded audio streams. We also identify the most dominant acoustic features capable of distinguishing devices with high accuracy.
  - **Fingerprinting Motion Sensors**: We investigate the feasibility of fingerprinting motion sensors such as accelerometers and gyroscopes in smartphones. These motion sensors are used by many applications such as health monitoring and interactive gaming. However, by measuring the anomalies in the signal generated by these motion sensors it is possible to uniquely track smartphones. Distressingly, such measurements can be conducted *surreptitiously*

in the browser as they do not require explicit user permission and can thus be used to track users across applications and websites. We find that simultaneous use of both accelerometer and gyroscope produces a more accurate fingerprint for the device than using only the accelerometer or gyroscope. We also show that the use of inaudible sound, played through the smartphone speaker, stimulates the onboard gyroscope uniquely and thus improves fingerprinting accuracy. To verify whether our fingerprinting technique holds for a large number of devices, we perform a large-scale user study to demonstrate that motion sensor fingerprinting is effective even with 500 users. We also develop a model to estimate prediction accuracy for larger user populations; our model provides a conservative estimate of at least 10% classification accuracy with 100 000 devices.

- **Mitigating Sensor Fingerprinting:** We next explore ways to counteract sensor fingerprinting and study their implication on the utility of the sensors.
  - **Possible Mitigation Techniques**: We look at how to mitigate fingerprinting of sensors like accelerometers and gyroscopes that are deemed non-sensitive (i.e., accessing them does not require explicit user permission). We investigate three different countermeasure techniques. First, we consider the use of *calibration* to eliminate some of the error that results from manufacturing imperfections. Promisingly, we find that calibrating the accelerometer is easy and has a significant impact on classification accuracy. Gyroscope calibration, however, is more challenging without specialized equipment, and attempts to calibrate the gyroscope by hand do not result in an effective countermeasure. Second, we introduce an alternative countermeasure called *obfuscation*, which introduces additional noise to the sensor readings in the hopes of hiding the natural errors. Obfuscation has the advantage of not requiring a calibration step; we find that by adding noise that is similar in magnitude to the natural errors that result from manufacturing imperfections, we can reduce the accuracy of fingerprinting more effectively than by calibration. We also investigate

a few variations of obfuscation where we explore adding noise in a differential privacy preserving manner (inspired by Differential Privacy), as well as adding white Gaussian noise to obfuscate frequency domain features. Lastly, we look at *quantization* where we lower the resolution of the sensor by mapping their values to fixed bins. The basic idea behind quantization is that human brain cannot discriminate minute changes in angle and/or magnitude, and as a result if the raw values from a sensor are altered slightly, this should not adversely impact the functionality of the sensor. We find that quantization also drastically reduces fingerprinting accuracy.

– **Analyzing Utility of Sensors**: Finally, we look at how our countermeasure techniques impact the utility of the underlying sensors. To evaluate this we first identify how motion sensors are being used in the wild. With this in mind we analyze the static and dynamic JavaScripts used by the top 100 000 Alexa websites [21]. We discover several common applications of motion sensors such as orientation and gesture detection. But distressingly, we find that a large fraction of scripts send motion data back to a server. Thus, although we have not been able to identify cases of motion sensor fingerprinting in the wild, the infrastructure for collecting and analyzing this data is already present in some cases. After identifying the most common applications, we evaluate the impact of our countermeasures for a step-counter and a tilt-based video game. These two applications are user-centric and thus provide us with both subjective and objective measures. We find that data obfuscation and sensor quantization do not adversely affect the utility of the motion sensors.

1.3 Dissertation Outline

The rest of the thesis is organized as follows. Chapter 2 provides background on the internal architecture of these hardware and highlights the most likely source of idiosyncrasies for these hardware. We also provide an elaborate description of various related works on fingerprinting devices.

Next, we describe how we used audio streams generated and/or received by smartphones to uniquely distinguish smartphones in Chapter 3. We start off by describing the experimental framework for capturing the unique characteristics of the onboard microphone and speaker. We then describe our acoustic feature extraction process and look at how different features contribute to generating a unique device fingerprint. Lastly, we look at how well we can fingerprint smartphones through microphones and speakers, both separately and collectively. We also investigate how different environmental factors impact the stability of our fingerprints.

Chapter 4 describes how onboard motion sensors can be exploited to track smartphones through the web browser. We first describe our data collection setup where we develop our own web page to collect motion sensor data. We then discuss the features that we used to generate the device fingerprint, along with the machine learning tools that we use for matching the fingerprints. Next, we evaluate how well we can fingerprint smartphones in both lab setting and public setting. Finally, we perform sensitivity analysis to determine how stable our fingerprints are under different environmental conditions.

Having shown that it is feasible to fingerprint motion sensors embedded in smartphones through a web page, we extend our experiment for a large-scale analysis in Chapter 5. Through Amazon's Mechanical Turk we were able to collect data from more than 500 devices and we study how our fingerprinting approach scales for this range of devices. We also generate parametric intra- and inter-cluster distance distributions from this large dataset. We utilize state-of-the-art distance metric learning algorithms to optimize our intra- and inter-cluster distances for a k-NN classifier. Such distance distributions enable us to emulate distance-based classifiers like k-NN to estimate how well our fingerprinting approach performs with 100 000 devices.

In Chapter 6 we perform a real world measurement study to determine how many of the top websites access motion sensors from smartphones. Our measurement study reveals that about 1% of the top 100 000 websites access motion sensors. We then broadly determine the use cases for accessing motion sensors. We find that there are broadly 8 different use cases for accessing motion sensor data, but disturbingly, majority of the websites send motion sensor data to third party sites for storage or analysis.

We discuss and explore the effectiveness of several countermeasures against

motion sensor fingerprinting in Chapter 7. All of our proposed countermeasures significantly thwart device fingerprinting accuracy. However, some of our countermeasures also adversely affect the utility of motion sensors. We, therefore, study and determine the countermeasures that have minimal impact on the utility of the underlying motion sensors. We perform a large-scale user study to show that our proposed countermeasures can be readily adopted by web browsers to better protect user privacy.

Lastly, we summarize all our findings and provide our final statement in Chapter 8.

1.4 Collaborators and Published Works

Most of our work has been peer-reviewed and published in top tier security conferences. In this context we would like to acknowledge our collaborators – Edward Chou for his help in designing user-studies and Muhammad Haris Mughees for his help in collecting JavaScripts accessing motion sensors in the wild. Following are the conference publications and technical reports related to the different chapters in this dissertation.

- **Fingerprinting smartphones using acoustic components**:
  - A. Das, N. Borisov, and M. Caesar, "Fingerprinting Smart Devices Through Embedded Acoustic Components," *CoRR*, vol. abs/1403.3366, 2014. [Online]. Available: http://arxiv.org/abs/1403.3366
  - A. Das, N. Borisov, and M. Caesar, "Do You Hear What I Hear?: Fingerprinting Smart Devices Through Embedded Acoustic Components," in *Proceedings of the 21st ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2014, pp. 441–452
- **Fingerprinting smartphones using motion sensors**:
  - A. Das, N. Borisov, and M. Caesar, "Exploring Ways To Mitigate Sensor-Based Smartphone Fingerprinting," *CoRR*, vol. abs/1503.01874, 2015. [Online]. Available: http://arxiv.org/abs/1503.01874

  - A. Das, N. Borisov, and M. Caesar, "Tracking Mobile Web Users Through Motion Sensors: Attacks and Defenses," in *Proceedings of the 23rd Annual Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2016

- **Fingerprinting smartphones at large-scale and analyzing motion sensor access patterns in the web**:

  - A. Das, N. Borisov, E. Chou, and M. H. Mughees, "Smartphone Fingerprinting Via Motion Sensors: Analyzing Feasibility at Large-Scale and Studying Real Usage Patterns," *CoRR*, vol. abs/1605.08763, 2016. [Online]. Available: http://arxiv.org/abs/1605.08763

CHAPTER 2

BACKGROUND AND RELATED WORK

In this chapter we take a closer look at some of the onboard hardware such as microphone, speaker, accelerometer and gyroscope. This will provide us with an understanding of how these hardware components can be used to uniquely fingerprint smartphones. We also discuss some of the most recent and well-known studies regarding device fingerprinting.

2.1 Acoustic Hardware

Microphone and speaker are the most common acoustic hardware available on any type of smartphone. They provide the fundamental functionality of a phone, which is placing and receiving phone calls. So we first take a closer look at their internal architecture to identify the potential source of imperfections that make them distinguishable.

2.1.1 Microphone

Microphones in modern smartphones are based on Micro-Electro-Mechanical Systems (MEMS) [27, 28, 29]. To enhance active noise and echo canceling capabilities, most smartphones today have more than one MEMS microphone. For example, the iPhone 5 has a total of three embedded MEMS microphones [28]. According to the IHS-iSuppli report, Apple and Samsung were the top consumers of MEMS microphones in 2012, accounting for a combined 54% of all shipped MEMS microphones [27].

A MEMS microphone, sometimes called a microphone chip or silicon microphone, consists of a coil-less pressure-sensitive diaphragm directly etched into a silicon chip. It comprises of a MEMS die and a complementary metal-oxide-semiconductor (CM-OS) die combined in an acoustic housing [30, 31]. The

CMOS often includes both a preamplifier and an analog-to-digital (AD) converter. Modern fabrication techniques enable highly compact deigns, making them well suited for integration in digital mobile devices. The internal architecture of a MEMS microphone is shown on Figure 2.1. From the figure we can see that the MEMS microphone's physical design is based on a *variable capacitor* consisting of a highly flexible diaphragm in close proximity to a perforated, rigid back-plate. The perforations permit the air between the diaphragm and back-plate to escape. When an acoustic signal reaches the diaphragm through the acoustic holes, the diaphragm is set in motion. This mechanical deformation causes capacitive change which in turn causes voltage change. In this way sound pressure is converted into an electrical signal for further processing. The back-chamber acts as an acoustic resonator and the ventilation hole allows the air compressed inside the back chamber to flow out, allowing the diaphragm to move back into its original place.

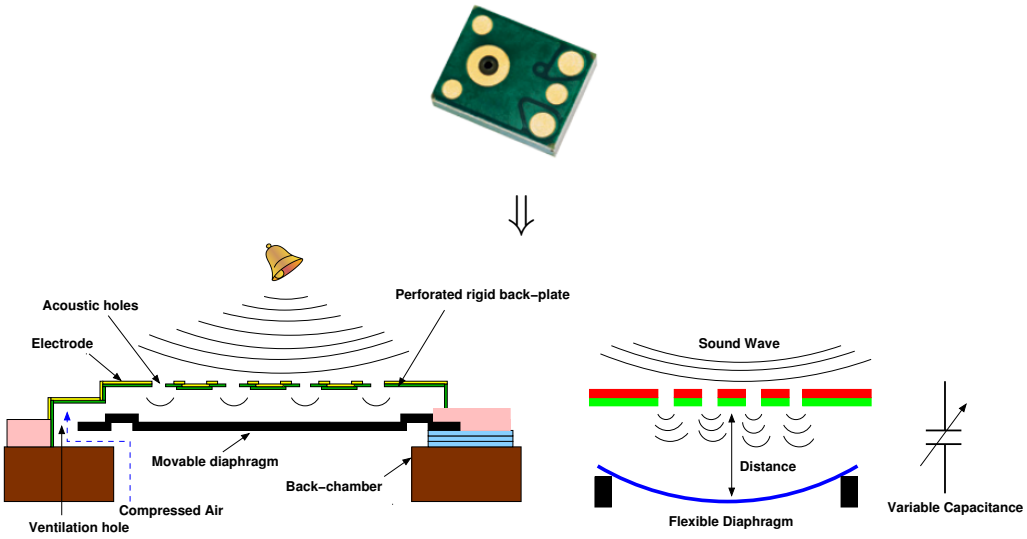


Figure 2.1: The internal architecture of a MEMS microphone chip used in modern smartphones.

The sensitivity of the microphone depends on how well the diaphragm deflects to acoustic pressure; it also depends on the gap between the static back-plate and the flexible diaphragm. Unfortunately, even though the manufacturing process for these microphones has been streamlined, no two chips roll off the assembly line functioning in exactly the same way. Imperfections can arise for the following reasons: slight variations in the chemical

composition of components from one batch to the next, wear in the manufacturing machines or changes in temperature and humidity. While subtle imperfections in the microphone chips may go unnoticed by human ears, computationally such discrepancies may be sufficient to discriminate them, as we later show.

2.1.2 Microspeaker

A microspeaker is a scaled down version of a basic acoustic speaker. So let us first look at how speakers work before we discuss how microspeakers can be used to generate unique fingerprints. Figure 2.2(a) shows the basic components of a speaker. The diaphragm is usually made of paper, plastic or metal and its edges are connected to the suspension which is a rim of flexible material that allows the diaphragm to move. The narrow end of the diaphragm's cone is connected to the voice coil. Voice coil is attached to the basket by a spider (damper), which holds the coil in position, but allows it to move freely back and forth. A permanent magnet is positioned directly below the voice coil.

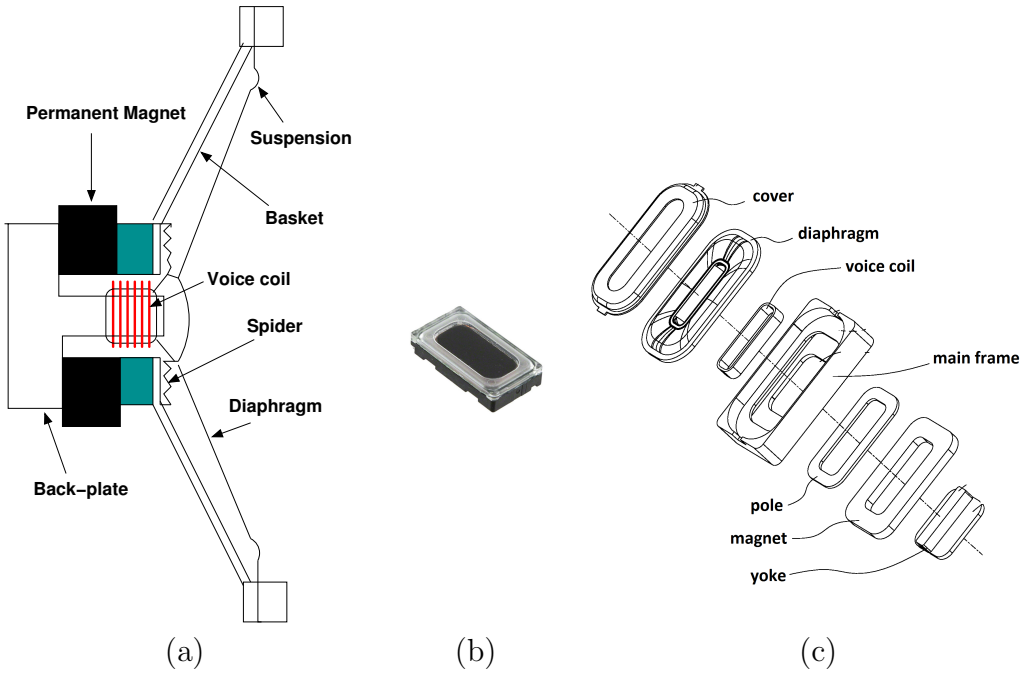


Figure 2.2: (a) The basic components of a speaker, (b) A typical MEMS microspeaker, (c) The internal architecture of a microspeaker chip.

Sound waves are produced whenever electrical current flows through the voice coil, which acts as an electromagnet. Running varying electrical current through the voice coil induces a varying magnetic field around the coil, altering the magnetization of the metal it is wrapped around. When the electromagnet's polar orientation switches, so does the direction of repulsion and attraction. In this way, the magnetic force between the voice coil and the permanent magnet causes the voice coil to vibrate, which in turn vibrates the speaker diaphragm to generate sound waves.

Figure 2.2(b) shows a typical MEMS microspeaker chip and Figure 2.2(c) shows the components inside the microspeaker [32]. The components are similar to that of a basic speaker; the only difference is the size and fabrication process [33, 34, 35]. The amplitude and frequency of the sound wave produced by the speaker's diaphragm is dictated respectively by the distance and rate at which the voice coil moves. Each speaker component can introduce variations into the generated sound. For example, variations in the electromagnetic properties of the driver can cause differences in the rate and smoothness at which the diaphragm moves. Therefore, due to the inevitable variations and imperfections of the manufacturing process, no two speakers are going to be alike, resulting in subtle differences in their produced sound. In our work, we develop techniques to computationally localize and evaluate these differences.

2.2 Motion Sensors

Motion sensing is transforming how users interact with smartphones and as a result motion sensing has become a "must have" feature for all major operating systems and hardware platform providers. Motion sensors enable a rich user-interactive experience in the form of gesture recognition, activity recognition and tracking. These capabilities enable sophisticated applications such as immersive gaming, augmented reality, fitness/health monitoring and accurate navigation, all of which are providing new ways to generate revenue for carriers and manufacturers. Accelerometer and gyroscope are the most common motion sensors available on smartphones. Accelerometer and gyroscope sensors in modern smartphones are based on Micro-Electro-Mechanical Systems (MEMS). STMicroelectronics [36] and InvenSense [37] are among

the top vendors supplying MEMS-based accelerometer and gyroscope sensor to different smartphone manufacturers [38]. Traditionally, Apple [39, 40][1] and Samsung [42, 43] favor using STMicroelectronics motion sensors, while Google [44, 45] tends to use InvenSense sensors.

2.2.1 Accelerometer

Accelerometer is a device that measures proper acceleration. Proper acceleration is different from coordinate acceleration (linear acceleration) as it measures the *g-force*. For example, an accelerometer at rest on a surface will measure an acceleration of $g = 9.81\mathtt{ms}^{-2}$ straight upwards, while for a free falling object it will measure an acceleration of zero. MEMS-based accelerometers are based on differential capacitance [46]. Figure 2.3 shows the internal architecture of a MEMS-based accelerometer. As we can see there are several pairs of fixed electrodes and a movable seismic mass. Under zero force the distances d_1 and d_2 are equal, and as a result the two generated capacitance are equal, but a change in force will cause the movable seismic mass to shift closer to one of the fixed electrodes (i.e., $d_1 \neq d_2$) causing a change in the generated capacitance. This difference in capacitance is detected and amplified to produce a voltage proportional to the acceleration. The slightest gap difference between the structural electrodes, introduced during the manufacturing process, can cause a change in the generated capacitance. Also, the flexibility of the seismic mass can be slightly different from one chip to another. These form of minute imprecisions in the electro-mechanical structure induce subtle imperfections in accelerometer chips.

2.2.2 Gyroscope

Gyroscope measures the rate of rotation (in $\mathtt{rads}^{-1}$) along the device's three axes. MEMS-based gyroscopes use the Coriolis effect to measure the angular rate. Whenever an angular velocity of $\hat{\omega}$ is exerted on a moving mass of weight m, and velocity $\hat{v}$, the object experiences a Coriolis force in a direction perpendicular to the rotation axis and to the velocity of the moving object (as shown in figure 2.4). The Coriolis force is calculated by the following

[1] iPhone 6 has been reported to use sensors made by InvenSense [41]

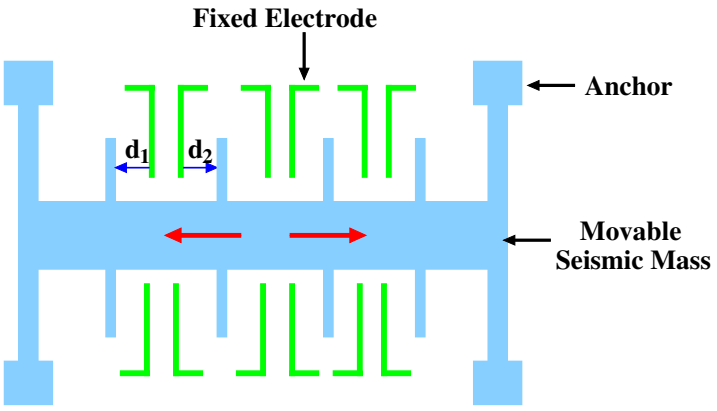


Figure 2.3: Internal architecture of a MEMS accelerometer. Differential capacitance is proportional to the applied acceleration.

equation, $\hat{F} = -2m\hat{\omega} \times \hat{v}$. Generally, the angular rate ($\hat{\omega}$) is measured by sensing the magnitude of the Coriolis force exerted on a vibrating proof-mass within the gyro [47, 48]. The Coriolis force is sensed by a capacitive sensing structure where a change in the vibration of the proof-mass causes a change in capacitance which is then converted into a voltage signal by the internal circuitry. Again the slightest imperfection in the electro-mechanical structure will introduce idiosyncrasies across chips.

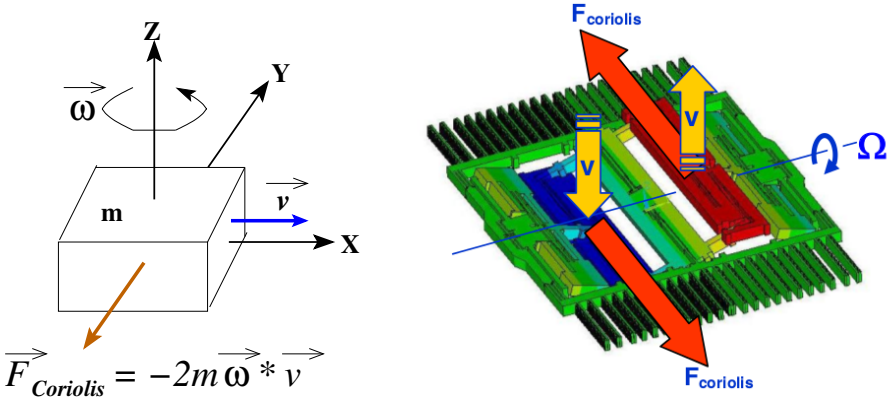


Figure 2.4: MEMS-based gyros use *Coriolis force* to compute angular velocity. The Coriolis force induces change in capacitance which is proportional to the angular velocity.

2.3 Related Work

Human fingerprints, due to their unique nature, are a very popular tool used to identify people in forensic and biometric applications [49, 50]. Researchers have long sought to find an equivalent of fingerprint in computer systems by finding characteristics that can help identify an individual device. Such fingerprints exploit variations in both the hardware and software of devices to aid in identification.

As early as 1960, the US government used unique transmission characteristics to track mobile transmitters [51]. Later, with the introduction of cellular network researchers were able to successfully distinguish transmitters by analyzing the spectral characteristics of the transmitted radio signal [52]. Researchers have suggested using radio-frequency fingerprints to enhance wireless authentication [53, 54], as well as localization [55]. Others have leveraged the minute manufacturing imperfections in network interface cards (NICs) by analyzing the radio-frequency of the emitted signals [56, 57]. Computer clocks have also been used for fingerprinting: Moon et al. showed that network devices tend to have a unique and constant clock skews [58]; Kohno et al. exploited this to distinguish network devices through TCP and ICMP timestamps [59].

Software can also serve as a distinguishing feature, as different devices have a different installed software base. Researchers have long been exploiting the difference in the protocol stack installed on IEEE 802.11 compliant devices. Desmond et al. [60] have looked at distinguishing unique devices over Wireless Local Area Networks (WLANs) simply by performing timing analysis on the 802.11 probe request packets. Others have investigated subtle differences in the firmware and device drivers running on IEEE 802.11 compliant devices [61]. 802.11 MAC headers have also been used to uniquely track devices [62]. Moreover, there are well-known open source toolkits like Nmap [63] and Xprobe [64] that can remotely fingerprint an operating system by analyzing unique response from the TCP/IP networking stack.

2.3.1 Browser Fingerprinting

A common application of fingerprinting is to track a user across multiple visits to a website, or a collection of sites. Traditionally, this was done with the aid

of cookies explicitly stored by the browser. However, privacy concerns have prompted web browsers to implement features that clear the cookie store, as well as provide private browsing modes that do not store cookies long-term. This has prompted site operators to develop other means of uniquely identifying and tracking users. Eckersley's *Panopticon* project showed that many browsers can be uniquely identified by enumerating installed fonts and other browser characteristics that are easily accessible via JavaScript [11]. A more advanced technique uses HTML5 canvas elements to fingerprint the fonts and rendering engines used by the browser [12]. Others have proposed the use of performance benchmarks for differentiating between JavaScript engines [65]. Lastly, browsing history can to used to profile and track online users [66]. Numerous studies have found evidence of these and other techniques being used in the wild [10, 14, 13]. A number of countermeasures to these techniques exist; typically they disable or restrict the ability of a website to probe the characteristics of a web browser. Nikiforakis et al. propose using random noise to make fingerprints non-deterministic which essentially breaks linkability across multiple visits [67].

With the rapid growth of smart devices, researchers are now focusing on adopting existing fingerprinting techniques in the context of smartphones. Like cookies, app developers have looked at using device IDs such as Unique Device Identifier (UDID) or International Mobile Station Equipment Identity (IMEI) to track users across multiple applications. However, Apple ceased the use of UDID since iOS 6 [68] and for Android accessing IMEI requires explicit user permission [69]. Moreover, due to constrained hardware and software environment existing methods often lack in precision for smartphones, and recent studies have shown this to be true [19, 20]. However, this year Laperdrix et al. have shown that it is in fact possible to fingerprint smartphones effectively through *user-agent* string which is becoming richer every day due to the numerous vendors with their different firmware updates [70]. Others have looked at fingerprinting smartphones by exploiting the personal configuration settings which are often accessible to third party apps [71].

2.3.2 Microphone and Speaker Fingerprinting

Our work is inspired by hardware-based fingerprinting techniques. We, firstly, focus on fingerprinting onboard speakers and microphones, and in this context the following works are closely related to ours.

Clarkson's work [72] showed that it is possible to distinguish loudspeakers by analyzing recorded audio samples emitting from them. However, his experiments used special audio clips that contained 65 different frequencies, whereas we are using common audio excerpts like ringtones. Moreover, his experiments ignored the subtlety introduced by microphones. In fact in one experiment, though statistically not meaningful as it tested only two similar microphones, they found no variation across microphones. We, on the other hand found that microphones can vary across different units. Finally, his study did not thoroughly analyze the different acoustic features that can be used to successfully carry out device fingerprinting. As a result, he was able to achieve only 81% accuracy in distinguishing heterogeneous loudspeakers.

Bojinov et al. [73] were investigating the feasibility of fingerprinting smartphones through both speakers and microphones around the same time. They were looking at the intensity ratio of the transmitted and received audio signal at 13 different frequencies. However, our experimental setup differs from them in several ways. Firstly, they experimented with only 16 devices all of the same make and model whereas we experimented with 52 devices from five different manufacturers. So, we test our approach for not only same make and model devices but also different make and model devices. Secondly, we look at fingerprinting the speaker and microphone individually as well as combining both of them. Lastly, we study the impact of ambient background noise on fingerprinting accuracy.

Contemporary to our work, Zhou et al. [74] have also looked at fingerprinting smartphones through speakers. They used high frequency inaudible sound to minimize the impact of background noise; while this is generally true, there are everyday environments like metro station that exhibit high frequency white noise. One point to note is that in the case of recording audio through the phone's built-in microphone they are actually fingerprinting both the speaker and microphone and not just the speaker. Also, their experiments were conducted using 50 OEM (Original Equipment Manufacturer) speaker chips on a single Samsung Galaxy S3. However, the internal

packaging of the different components inside the phone along with the surface on which the phone is kept impacts the produced audio signal [73]. So, compared to our work their experimental setup is less realistic because we use individual smartphones instead of individual speaker chips. Finally, their study only looks at speakers of the same make and model whereas we test our approach for both same and different make and model devices.

On a different note *audio fingerprinting* has a rich history of notable research [75]. There are studies that have looked at classifying audio excerpts based on their content [76, 77]. Others have looked at distinguishing human speakers from audio segments [78, 79]. There has also been work on exploring various acoustic features for audio classification [80]. One of the more popular applications of audio fingerprinting has been genre and artist recognition [81, 82]. Our work takes advantage of the large set of acoustic features that have been explored by existing work in audio fingerprinting. However, instead of classifying the content of audio segments, we utilize acoustics features to capture the manufacturing imperfections of microphones and speakers embedded in smartphones.

2.3.3 Motion Sensor Fingerprinting

While it is possible to fingerprint smartphones through microphones and speakers, such techniques require access to the microphone, which is typically controlled with a separate permission due to the obvious privacy concerns with the ability to capture audio. On the other hand accessing motion sensors is considered not sensitive and as such no explicit user permission is required to access them. We, therefore, also focus on fingerprinting smartphones surreptitiously through accelerometers and gyroscopes. The following studies closely resemble our work in this dissertation.

Bojinov et al. [73] consider using accelerometers to fingerprint smartphones. Their techniques, however, rely on having the user perform a calibration of the accelerometer, the parameters of which are used to distinguish phones. Dey et al. [83] apply machine learning techniques to create an accelerometer fingerprint, but they require the vibration motor to be active to stimulate the accelerometer sensor; in the absence of external stimulation, they report an average precision and recall of around 87% for 25 station-

ary phones. In contrast, our work studies phones that are in a natural web-browsing setting, either in a user's hand or resting on a flat surface. Additionally, we consider the simultaneous use of both accelerometer and gyroscope to produce a more *accurate fingerprint*. Inspired by prior work that uses the gyroscope to recover audio signals [84], we also stimulate the gyroscope with an inaudible tone. Moreover, our work provides a real world perspective on the problem. We not only show that sensor-based fingerprinting works at large-scale but also show how websites are accessing the sensor data in the wild. Finally, we propose and evaluate several *countermeasures* to reduce fingerprinting accuracy without entirely blocking access to the motion sensors. To analyze the impact of our countermeasures we perform a large-scale user study where users play an online game to show that our countermeasures do not affect the utility of the motion sensors. Table 2.1 highlights some comparisons with related works.

Table 2.1: Comparison with other motion sensor fingerprinting studies.

Work	Sensors[a]	Setting	Stimulation	Features Explored	Features Used	# of Devices	Result[b]
[83]	A	Lab	Vibration	80	36	107[c]	99% *Acc*
[83]	A	Lab	None	80	36	25	87% F_s
[73]	A	Lab	Flip phone	2	2	33	100% *Acc*
[73]	A	Public	Flip phone	2	2	3583[d]	15% *Acc*
Ours	A,G	Lab	None	100	70	30	99% F_s
Ours	A,G	Public	None	100	70	471[e]	86% F_s
Ours	A,G	Lab+Public	None	100	70	501	86% F_s
Ours	A,G	Lab	In hand	100	70	30	93% F_s
Ours	A,G	Lab	In hand+Audio	100	70	30	98% F_s

[a] here **'A'** means accelerometer and **'G'** refers to gyroscope
[b] here '*Acc*' means Accuracy and 'F_s' refers to F-score
[c] 80 external chips, 25 phones and 2 tablets
[d] considering only devices with two submissions
[e] considering only devices with at least 5 training samples

CHAPTER 3

FINGERPRINTING SMARTPHONES VIA MICROPHONES AND SPEAKERS

All smartphones have acoustic components like microphone and speaker. These are the very basic hardware available in any mobile phone. These acoustic components are used for many purposes in today's multimedia-rich ecosystem starting from simple call making to interacting with games. However, these acoustic hardware can be exploited to fingerprint smartphones. During fabrication, subtle imperfections arise in device microphone and speaker which induce anomalies in the generated and received audio signal. These anomalies can be utilized to generate unique fingerprints for smartphones.

3.1 Overview

We start with an overview of our approach and present several viable attack scenarios. The key observation behind our work is that imperfections in smartphone hardware induce unique signatures on the received and transmitted audio streams, and these unique signatures, if identified, can be used by an adversary to fingerprint the device. We consider three fingerprinting scenarios: speaker, microphone, and joint speaker-microphone fingerprinting. In the first case, an attacker in a public environment, such as a cafe or shopping mall, records audio generated by a smartphone speaker, such as a ringtone (malicious app can also transmit inaudible sounds to remain undetectable). The attacker can then use the recorded audio samples to track and identify users as shown in Figure 3.1. Alternately, the attacker may obtain audio recorded by a smartphone microphone and use that to identify the user who made the recording as shown in Figure 3.2; this can have forensic applications. A third way to track users is to convince them to install a malicious application (e.g., a free online game), which can play and record

audio clips using the device's speaker and microphone. The app can then stealthily upload the recorded audio clips to the attacker (e.g., piggybacking it on log-in information or game state), who can then use the audio samples to uniquely distinguish each user as shown in Figure 3.3. To do this, the application would require access to both the speaker and microphone, as well as network access, but such permissions are very common [85] and are unlikely to raise alarm, especially given that a significant portion of the users cannot comprehend the full consequences of smartphone permissions [86, 87, 88].

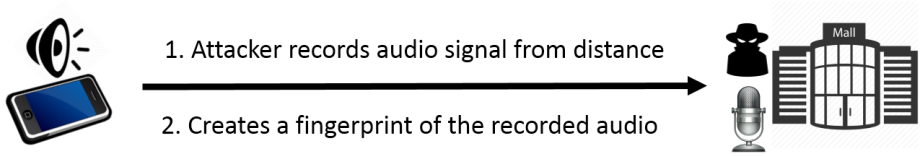


Figure 3.1: Fingerprinting smartphone speakers in public location.

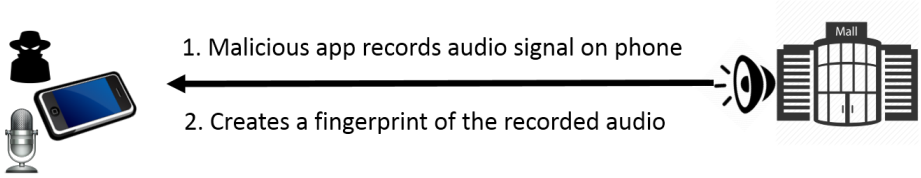


Figure 3.2: Fingerprinting smartphone microphones in public location.

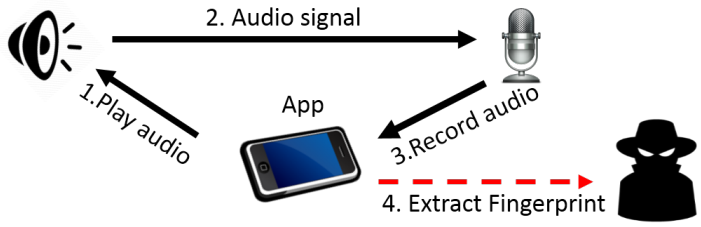


Figure 3.3: Fingerprinting both smartphone speakers and microphones.

3.2 Methodology

Our approach consists of two main tasks. The first task is acquiring a set of audio samples for analysis in the first place. To do this, we have a *listener* module, responsible for receiving and recording device audio. The listener

module could be deployed as an application on the smartphone (many mobile OSes allow direct access to microphone input), or as a stand-alone service (e.g., the adversary has a microphone in a public setting like shopping mall to pick up device audio). The next task is to effectively identify device signatures from the received audio stream. To do this, we have an *analyzer* module, which leverages signal processing techniques to localize spectral anomalies, and constructs a 'fingerprint' of the auditory characteristics of the device. We individually evaluate the feasibility of fingerprinting speakers, microphones and a combination of both.

3.2.1 Procedure for Fingerprinting Speakers

An attacker can leverage our technique to passively observe audio signals (e.g., ringtones or inaudible sound) emitted from device speakers in public environments. To investigate this, we first look at fingerprinting speakers integrated inside smartphones. For fingerprinting speakers we record audio clips played from smartphones onto a laptop and we then extract acoustic features from the recorded audio excerpts to generate fingerprints as shown in Figure 3.4. We look at devices manufactured by both same vendor and different vendors.

Figure 3.4: Steps for fingerprinting speakers.

3.2.2 Procedure for Fingerprinting Microphones

Attackers may also attempt to fingerprint devices by observing imperfections in device microphone, for example by convincing the user to install an application on their phone, which can observe inputs from the device's microphone. To investigate the feasibility of this attack, we next look at

fingerprinting microphones embedded in smartphones. To do this, we record audio clips played from a laptop onto smartphones as shown in Figure 3.5. Again we consider devices made by both same vendor and different vendors.

Figure 3.5: Steps for fingerprinting microphones.

3.2.3 Procedure for Fingerprinting both Speakers and Microphones

An attacker may attempt to fingerprint devices by observing imperfections in both device microphone and speaker, for example by convincing the user to install a game on their phone which requires access to device speaker and microphone to interact with the game (something like *My Talking Tom* [89]). The attacker could potentially play a theme song at the start of the game and at the same time make a recording of the audio clip. To investigate the feasibility of this attack, we develop our own android app that plays and records audio clips simultaneously and uploads the data to a remote server. The recorded audio clips would then enable the attacker to characterize the imperfections in microphones and speakers embedded inside smartphones. Figure 3.6 summarizes the whole process.

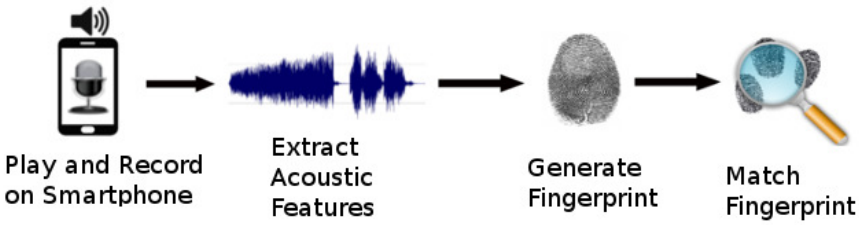


Figure 3.6: Steps for fingerprinting both microphone and speaker.

3.3 Experimental Setup

To perform our experiments, we constructed a small testbed environment with real smartphone device hardware. In particular, our default environment consisted of a 266 square foot (14'x19') office room, with nine-foot dropped ceilings with polystyrene tile, comprising a graduate student office in a University-owned building. The room was filled with desks and chairs, and opens out to a public hall with footstep traffic. The room also receives a minimal amount of ambient noise from air conditioning, desktop computers, and florescent lighting. We placed smartphones in various locations in the room. To emulate an attacker, we placed an ACER Aspire–5745 laptop in the room. To investigate performance with inexpensive hardware, we used the laptop's built-in microphone to collect audio samples. We investigate how varying this setup affects the performance of the attack in latter parts of this chapter.

3.3.1 Device Types

We test our device fingerprinting approach on devices from five different manufacturers. Table 3.1 highlights the models and quantities of the different phones used in our experiments.

Table 3.1: Types of phones used for fingerprinting acoustic components.

Maker	Model	Quantity
Apple	iPhone 5	1
HTC	Nexus One	14
Samsung	Nexus S	8
	Galaxy S3	3
	Galaxy S4	10
Motorola	Droid A855	15
Sony Ericsson	W518	1
Total		52

3.3.2 Audio Genre Types

We also investigate different genres of audio excerpts. Table 3.2 describes the different types of audio excerpts used in our experiments. Duration of the

audio clips varies from 3 to 10 seconds. The default sampling frequency for all audio excerpts is 44.1 kHz unless explicitly stated otherwise. All audio clips are stored in WAV format [90] using 16-bit pulse-code-modulation (PCM) technique.

Table 3.2: Types of audio excerpts used in our experiments.

Type	Description	Variations
Instrumental	Musical instruments playing together, e.g., ringtone	4
Human speech	Small segments of human speech	4
Song	Combination of human voice & instrumental sound	3

3.3.3 Analytic Tools

For analysis, we leverage the following audio tools and analytic modules: *MIRtollbox* [91], *Netlab* [92], *Audacity* [93] and *Hertz* [94]. Both MIRtoolbox and Netlab are MATLAB modules providing a rich set of functions for analyzing and extracting audio features. Audacity and Hertz are mainly used for recording audio clip on laptop and smartphone, respectively.

3.4 Acoustic Features

Given our knowledge that imperfections exist in device audio hardware, we now need some way to detect them. To do this, our approach identifies *acoustic features* from an audio stream, and uses the features to construct a *fingerprint* of the device. Computing acoustic features from an audio stream has been a subject of much research [80, 75, 95, 76]. To gain an understanding of how a broad range of acoustic features are affected by device imperfections we investigate 15 different acoustic features (listed in Table 3.3), all of which have been well-documented by researchers. A brief description of each acoustic feature follows.

Root-Mean-Square (RMS) Energy: This feature computes the square root of the arithmetic mean of the squares of the original audio signal strength at various frequencies. In the case of a set of N values $\{x_1, x_2, \ldots, x_N\}$, the

Table 3.3: Explored acoustic features.

#	Feature	Dimension
1	RMS	1
2	ZCR	1
3	Low-Energy-Rate	1
4	Spectral Centroid	1
5	Spectral Entropy	1
6	Spectral Irregularity	1
7	Spectral Spread	1
8	Spectral Skewness	1
9	Spectral Kurtosis	1
10	Spectral Rolloff	1
11	Spectral Brightness	1
12	Spectral Flatness	1
13	MFCCs	13
14	Chromagram	12
15	Tonal Centroids	6

RMS value is given by the following formula:

$$x_{\text{rms}} = \sqrt{\frac{1}{n}\left(x_1^2 + x_2^2 + \cdots + x_N^2\right)} \tag{3.1}$$

The RMS value provides an approximation of the average audio signal strength.

Zero Crossing Rate (ZCR): The zero-crossing rate is the rate at which the signal changes sign from positive to negative or back [96]. This feature has been used heavily in both speech recognition and music information retrieval, for example to classify percussive sounds [97]. ZCR for a signal s of length T can be defined as:

$$ZCR = \frac{1}{T}\sum_{t=1}^{T} |s(t) - s(t-1)| \tag{3.2}$$

where $s(t) = 1$ if the signal has a positive amplitude at time t and 0 otherwise. Zero-crossing rate provides a measure of the noisiness of the signal.

Low Energy Rate: The low energy rate computes the percentage of frames (typically 50 `ms` chunks) with RMS power less than the average RMS power for the whole audio signal. For instance, a musical excerpt with some very loud frames and a lot of silent frames would have a high low-energy rate.

Spectral Centroid: Spectral centroid represents the "center of mass" of a spectral power distribution. It is calculated as the weighted mean of the frequencies present in the signal, determined using a Fourier transform, with their magnitudes as the weights:

$$Centroid, \mu = \frac{\sum_{i=1}^{N} f_i \cdot m_i}{\sum_{i=1}^{N} m_i} \tag{3.3}$$

where m_i represents the magnitude of bin number i, and f_i represents the center frequency of that bin.

Spectral Entropy: Spectral entropy captures the spikiness of a spectral distribution. As a result spectral entropy can be used to capture the formants or peaks in the sound envelope [98]. To compute spectral entropy, a Digital Fourier Transform (DFT) of the signal is first carried out. Next, the frequency spectrum is converted into a probability mass function (PMF) by normalizing the spectrum using the following equation:

$$w_i = \frac{m_i}{\sum_{i=1}^{N} m_i} \tag{3.4}$$

where m_i represents the energy/magnitude of the i-th frequency component of the spectrum, $w = (w_1, w_2, \ldots, w_N)$ is the PMF of the spectrum and N is the number of points in the spectrum. This PMF can then be used to compute the spectral entropy using the following equation:

$$H = \sum_{i=1}^{N} w_i \cdot log_2 w_i \tag{3.5}$$

The central idea of using entropy as a feature is to capture the peaks of the spectrum and their location.

Spectral Irregularity: Spectral irregularity measures the degree of variation of the successive peaks of a spectrum. This feature provides the ability to capture the jitter or noise in a spectrum. Spectral irregularity is computed as the sum of the square of the difference in amplitude between adjoining spectral peaks [99] using the following equation:

$$Irregularity = \frac{\sum_{i=1}^{N} (a_i - a_{i+1})^2}{\sum_{i=1}^{N} a_i^2} \tag{3.6}$$

where the $(N+1)$-th peak is assumed to be zero. A change in irregularity changes the perceived timbre of a sound.

Spectral Spread: Spectral spread defines the dispersion of the spectrum around its centroid, i.e., it measures the standard deviation of a spectrum. So it can be computed as:

$$Spread, \sigma = \sqrt{\sum_{i=1}^{N} [(f_i - \mu)^2 \cdot w_i]} \tag{3.7}$$

where w_i represents the weight of the i-th frequency component obtained from Equation (3.4) and μ represents the centroid of the spectrum obtained from Equation (3.3).

Spectral Skewness: Spectral skewness computes the coefficient of skewness of a spectrum. Skewness (third central moment) measures the symmetry of the distribution. A distribution can be positively skewed in which case it has a long tail to the right while a negatively-skewed distribution has a longer tail to the left. A symmetrical distribution has a skewness of zero. The coefficient of skewness is the ratio of the skewness to the standard deviation raised to the third power.

$$Skewness = \frac{\sum_{i=1}^{N} [(f_i - \mu)^3 \cdot w_i]}{\sigma^3} \tag{3.8}$$

Spectral Kurtosis: Spectral Kurtosis gives a measure of the flatness or spikiness of a distribution relative to a normal distribution. It is computed from the fourth central moment using the following function:

$$Kurtosis = \frac{\sum_{i=1}^{N} [(f_i - \mu)^4 \cdot w_i]}{\sigma^4} \tag{3.9}$$

A kurtosis value of 3 means the distribution is similar to a normal distribution whereas values less than 3 refer to flatter distributions and values greater than 3 refer to steeper distributions.

Spectral Rolloff: The spectral rolloff is defined as the frequency below

which 85% of the distribution magnitude is concentrated [76]

$$\arg\min_{f_c \in \{1,\ldots,N\}} \sum_{i=1}^{f_c} m_i \geq 0.85 \cdot \sum_{i=1}^{N} m_i \tag{3.10}$$

where f_c is the rolloff frequency and m_i is the magnitude of the i-th frequency component of the spectrum. The rolloff is another measure of spectral shape that is correlated to the noise cutting frequency [100].

Spectral Brightness: Spectral brightness calculates the amount of spectral energy corresponding to frequencies higher than a given cut-off threshold. This metric correlates to the perceived timbre of a sound. Increase of higher frequency energy in the spectrum yields a sharper timbre, whereas a decrease yields a softer timbre [101]. Spectral brightness can be computed using the following equation:

$$Brightness_{f_c} = \sum_{i=f_c}^{N} m_i \tag{3.11}$$

where f_c is the cut-off frequency (set to 1500 Hz) and m_i is the magnitude of the i-th frequency component of the spectrum.

Spectral Flatness: Spectral flatness measures how energy is spread across the spectrum, giving a high value when energy is equally distributed and a low value when energy is concentrated in a few narrow frequency bands. The spectral flatness is calculated by dividing the geometric mean of the power spectrum by the arithmetic mean of the power spectrum [102]:

$$Flatness = \frac{\left[\prod_{i=1}^{N} m_i\right]^{1/N}}{\frac{1}{N}\sum_{i=1}^{N} m_i} \tag{3.12}$$

where m_i represents the magnitude of bin number i. Spectral flatness provides a way to quantify the noise-like or tone-like nature of the signal. One advantage of using spectral flatness is that it is not affected by the amplitude of the signal, meaning spectral flatness virtually remains unchanged when the distance between the sound source and microphone fluctuates during recording.

Mel-Frequency Cepstrum Coefficients (MFCCs): MFCCs are short-term spectral features and are widely used in the area of audio and speech

processing [103, 76]. Their success has been due to their capability of compactly representing spectrum amplitudes. Figure 3.7 highlights the procedure for extracting MFCCs from audio signals. The first step is to divide the signal into fixed size frames (typically 50 `ms` chunks) by applying a windowing function at fixed intervals. The next step is to take Discrete Fourier Transform (DFT) of each frame. After taking the log-amplitude of the magnitude spectrum, the DFT bins are grouped and smoothed according to the perceptually motivated Mel-frequency scaling.[1] Finally, in order to decorrelate the resulting feature vectors a discrete cosine transform is performed. We use the first 13 coefficients for our experiments.

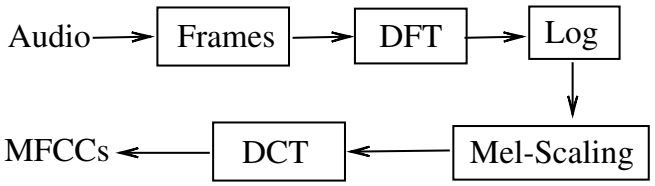


Figure 3.7: Procedure for extracting MFCCs from audio signals.

Chromagram: A chromagram (also known as harmonic pitch class profile) is a 12-dimensional vector representation of an audio signal showing the distribution of energy along the 12 distinct semitones or pitch classes. First a DFT of the audio signal is taken and then the spectral frequencies are mapped onto a limited set of 12 chroma values in a many-to-one fashion [104]. In general, chromagram is robust to noise (e.g., ambient noise or percussive sounds) and independent of timbre change.

Tonal Centroids: Tonal centroid introduced by Harte et al. [105] maps a chromagram onto a 6-dimensional Hypertorus structure. The resulting representation wraps around the surface of a Hypertorus, and can be visualized as a set of three circles of harmonic pitch intervals: fifths, major thirds, and minor thirds. Tonal centroids are efficient in detecting changes in harmonic contents.

[1]Mel-scale approximates the human auditory response more closely than the linearly-spaced frequency bands. http://en.wikipedia.org/wiki/Mel_scale

3.5 Classification Algorithms and Evaluation Metrics

Before we dig deep into the evaluation section, let us briefly describe the classification algorithms and metrics that we used to determine how well we can fingerprint smartphones using onboard microphones and speakers.

3.5.1 Classification Algorithms

We need some way to leverage the set of features to perform device identification. To achieve this, we leverage a *supervised* classification algorithm, which takes observations (features) from the observed device as input, and attempts to classify the device into one of several previously-observed devices. To do this, our approach works as follows. First, we perform a training step, by collecting a number of observations from a set of devices. Each observation (data point) corresponds to a set of features observed from that device, represented as a tuple with one dimension per feature. As such, data points can be thought of as existing in a hyper-dimensional space, with each axis corresponding to the observed value of a corresponding feature. Our approach then applies a classification algorithm to build a representation of these data points, which can later be used to associate new observations with device types. When a new observation is collected, the classification algorithm returns the most likely device that caused the observation.

To do this effectively, we need an efficient classification algorithm. In our work, we compare the performance of two alternate approaches described below: *k-nearest neighbors* (associates an incoming data point with the device corresponding to the nearest "learned" data points), and *Gaussian mixture models* (computes a probability distribution for each device, and determines the maximally-likely association).

k-NN: k-nearest neighbor algorithm (k-NN) is a non-parametric lazy learning algorithm. The term "non-parametric" means that the k-NN algorithm does not make any assumptions about the underlying data distribution, which is useful in analyzing real world data with complex underlying distribution. The term "lazy learning" means that the k-NN algorithm does not use the training data to make any generalization, rather all the training data are used in the testing phase making it computationally expensive

(however, different optimizations are possible). k-NN algorithm works by first computing the distance from the input data point to all training data points and then classifies the input data point by taking a majority vote of the k closest training records in the feature space [106]. The best choice of k depends upon the data; generally, larger values of k reduce the effect of noise on the classification, but make boundaries between classes less distinct.

GMM: A Gaussian mixture model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. The unknown patterns and mixture weights are estimated from training samples using an *expectation–maximization* (EM) algorithm [107]. During the matching phase the fingerprint for an unknown recording is first compared with a database of precomputed GMMs and then the class label of the GMM that gives the highest likelihood is returned as the expected class for the unknown fingerprint. GMMs are often used in biometric systems, most notably in human speaker recognition systems, due to their capability of representing a large class of sample distributions [108, 76].

For analyzing and matching fingerprints we use a desktop machine with the following configuration: Intel i7-2600 3.4 GHz processor with 12 GiB RAM. We found that the average time required to match a new fingerprint was around 5–10 `ms` for k-NN classifier and around 0.5–1 `ms` for GMM classifier.

3.5.2 Evaluation Metrics:

We use standard multiclass classification metrics such as *precision*, *recall*, and *F-score* [109] in our evaluation. Assuming there are fingerprints from n classes (e.g., n different devices), we first compute the true positive (TP) rate for each class, i.e., the number of traces from the class that are classified correctly. Similarly, we compute the false positive (FP) and false negative (FN), as the number of wrongly accepted and wrongly rejected traces, respectively, for each class i ($1 \leq i \leq n$). We then compute precision, recall,

and F-score for each class using the following equations:

$$\text{Precision, } Pr_i = \frac{TP_i}{TP_i + FP_i} \tag{3.13}$$

$$\text{Recall, } Re_i = \frac{TP_i}{TP_i + FN_i} \tag{3.14}$$

$$\text{F-Score, } F_i = \frac{2 \times Pr_i \times Re_i}{Pr_i + Re_i} \tag{3.15}$$

F-score is the harmonic mean of precision and recall; it provides a good measure of overall classification performance, since precision and recall represent a tradeoff: a more conservative classifier that rejects more instances will have higher precision but lower recall, and vice versa. To obtain the overall performance of the system we compute average values in the following way:

$$\text{Avg. Precision, } AvgPr = \frac{\sum_{i=1}^{n} Pr_i}{n} \tag{3.16}$$

$$\text{Avg. Recall, } AvgRe = \frac{\sum_{i=1}^{n} Re_i}{n} \tag{3.17}$$

$$\text{Avg. F-Score, } AvgF = \frac{2 \times AvgPr \times AvgRe}{AvgPr + AvgRe} \tag{3.18}$$

Each audio excerpt is recorded/played 10 times, 50% of which is used for training and the remaining 50% is used for testing. The selection of training and testing sample is done in *random*. To prevent any bias in the selection of the training and testing set we rerun our experiments 10 times and report the average F-score. We report the maximum evaluation obtained by varying the number of nearest-neighbors (k) from 1 to 5 for k-NN classifier and considering 1 to 5 Gaussian distributions per class for GMM classifier. Since GMM parameters are produced by the randomized EM algorithm, we perform 10 parameter-generation runs for each instance and report the average classification performance. We also compute the 95% confidence interval, but we found it to be less than 1% and therefore, do not report it in the rest of the chapter.

3.6 Feature Exploration

At first glance, it might seem that we should use all features at our disposal to identify device types. However, including too many features can *worsen*

performance in practice, due to their varying accuracies and potentially-conflicting signatures. Hence, in this section, we provide a framework to explore all the 15 audio features described in Table 3.3 and identify the *dominating subset* of all features, i.e., which combination of features should be used. For this purpose we adopt a well known machine learning strategy known as *feature selection* [110, 111]. Feature selection is the process of reducing dimensionality of data by selecting only a subset of the relevant features for use in model construction. The main assumption in using feature selection technique is that the data may contain redundant features. Redundant features are those which provide no additional benefit than the currently selected features. Feature selection techniques are a subset of the more general field of *feature extraction*, however, in practice they are quite different from each other. Feature extraction creates new features as functions of the original features, whereas feature selection returns a subset of the original features. It is preferable to use feature selection over feature extraction when the original units and meaning of features are important and the modeling goal is to identify an influential subset. When the features themselves have different dimensionality, and numerical transformations are inappropriate, feature selection becomes the primary means of dimension reduction.

Feature selection involves the maximization of an objective function as it searches through the possible candidate subsets. Since exhaustive evaluation of all possible subsets are often infeasible (2^N for a total of N features) different heuristics are employed. We use a greedy search strategy known as *sequential forward selection* (SFS) where we start off with an empty set and sequentially add the features that maximize our objective function. The pseudo code of our feature selection algorithm is described in Algorithm 1.

The algorithm works as follows. First, we compute the F-score that can be achieved by each feature individually. Next, we sort the features based on their achieved F-score in descending order. Then, we iteratively add features starting from the most dominant one and compute the F-score of the combined feature subset. If adding a feature increases the F-score seen so far we move on to the next feature, else we remove the feature under inspection. Having traversed through the entire set of features, we return the subset of features that maximizes our device classification task. Note that this is a greedy approach, therefore, the generated subset might not

Algorithm 1 Sequential Feature Selection (SFS).

Input: Input feature set F
Output: Dominant feature subset D
$F\_score \leftarrow [\ ]$
for $f \in F$ **do**
  $F\_score[f] \leftarrow Classify(f)$
end for
$F' \leftarrow sort(F, F\_score)$ #In descending order
$max\_score \leftarrow 0$
$D \leftarrow \varnothing$
for $f \in F'$ **do**
  $D \leftarrow D \cup f$
  $temp \leftarrow Classify(D)$
  if $temp > max\_score$ **then**
    $max\_score \leftarrow temp$
  else
    $D \leftarrow D - \{f\}$
  end if
end for
return D

always provide the optimal F-score. However, for our purpose, we found this approach to perform well, as we demonstrate in our evaluation section. We test our feature selection algorithm for all three types of audio excerpts listed in Table 3.2. We report the maximum F-score obtained by varying k from 1 to 5 for k-NN classifier and also considering 1 to 5 Gaussian distributions per class for GMM classifier.

3.6.1 Feature Exploration for Different Make and Model

First, we look at features obtained from smartphones manufactured by five different vendors. We take one representative smartphone from each row of Table 3.1 giving us a total of 7 different phones. Each type of audio is recorded 10 times giving us a total of 70 samples from the 7 representative handsets; 50% of which (i.e., 5 samples per handset) is randomly selected for training and the remaining 50% is used for testing. All the training samples are labeled with their corresponding handset identifier. Both classifiers return the class label for each audio clip in the test set and from that we compute F-score.

Table 3.4: Feature exploration for different make and model smartphones using only speaker.

#	Feature	Fingerprinting Speakers Maximum F-Score (%)					
		Instrumental		Human Speech		Song	
		k-NN	GMM	k-NN	GMM	k-NN	GMM
1	RMS	**97.4**	98	80.8	78.3	88.8	86.2
2	ZCR	57.6	52.1	63.7	48.6	77	77
3	Low-Energy-Rate	69.2	51.8	52.8	38.2	59.6	58.8
4	Spectral Centroid	91.5	88.4	59.4	55.8	88.1	87.7
5	Spectral Entropy	84	80.5	59.4	46.5	91.5	91
6	Spectral Irregularity	31.3	51.4	37.5	46.8	43.2	50.3
7	Spectral Spread	90.2	89.2	56.7	56.7	91	85.7
8	Spectral Skewness	68.3	82.1	69	62	82.9	79.9
9	Spectral Kurtosis	76.7	79.5	68.1	60.2	88.6	86.9
10	Spectral Rolloff	86.6	86.4	85.8	66.7	74.8	76.1
11	Spectral Brightness	87.5	85.2	70.9	87.7	85.5	77.1
12	Spectral Flatness	84.5	84	61.6	61.3	95.1	97.4
13	MFCCs	**97.4**	**100**	**100**	**100**	94.8	**100**
14	Chromagram	84.3	79.4	81.1	**100**	97.4	**100**
15	Tonal Centroid	86.4	88.4	80.3	98.2	**100**	**100**
Sequential Feature Selection		[1,7]	[13]	[13]	[13]	[15]	[13]
Max F-Score		100	100	100	100	100	100

Table 3.4 highlights the subset of features selected by our sequential feature selection algorithm for features obtained from different brands of smartphone speakers. We find that most of the time *MFCCs* are the dominant features for all categories of audio excerpt. We see similar outcomes for features obtained from different brands of smartphone microphones in Table 3.5. And when we combine both speaker and microphone (Table 3.6), we see that both *MFCCs* and *Tonal Centroids* provide high F-scores.

3.6.2 Feature Exploration for Same Make and Model

Next, we look at features obtained from phones manufactured by the same vendor and are of the same model. From Table 3.1 we see that we have 15 Motorola Droid A855 handsets, which is the largest number among all the other types of phones in our collection. We, therefore, use these 15 devices for all the experiments in this section. Again, each type of audio is recorded 10 times giving us a total of 150 samples from the 15 handsets; 50% of which

Table 3.5: Feature exploration for different make and model smartphones using only microphone.

#	Feature	Fingerprinting Microphones Maximum F-Score (%)					
		Instrumental		Human Speech		Song	
		k-NN	GMM	k-NN	GMM	k-NN	GMM
1	RMS	87.2	84.4	62.7	70.3	80.2	82.8
2	ZCR	76.4	74.3	75.7	76.8	72.4	71.3
3	Low-Energy-Rate	47	42.2	26	19.5	38.8	36.6
4	Spectral Centroid	73.7	69.7	70	77.9	73.7	76
5	Spectral Entropy	52.1	56.5	68	59.8	68.8	58.8
6	Spectral Irregularity	49.3	55	52.6	49.7	53	48.7
7	Spectral Spread	90.4	86.2	50.5	54.8	81.6	76.5
8	Spectral Skewness	71.4	63.4	65.6	63.6	61.8	49.7
9	Spectral Kurtosis	63.9	61.2	65.2	65.1	85.4	58
10	Spectral Rolloff	37.8	42.7	82.9	83.8	67.7	73
11	Spectral Brightness	66.4	64.8	60.3	60.2	63.7	67.4
12	Spectral Flatness	92.5	92.5	66.7	68.9	74.8	74.4
13	MFCCs	**94.8**	**94.8**	79.9	**92.1**	**90.4**	**95.4**
14	Chromagram	88.4	77.7	88.7	86	78.4	87.7
15	Tonal Centroid	91.9	92.7	**92.1**	86.4	89.6	92.5
Sequential Feature Selection		[13,1]	[13,1,7]	[15,9,1]	[13,15,11]	[13,1,12]	[13,1,9]
Max F-Score		97.4	100	92.1	93	92.5	97.4

is randomly selected for training and the remaining 50% is used for testing.

Table 3.7 shows the maximum F-score achieved by each acoustic feature for the three different types of audio excerpt. The table also highlights the dominating subset of features selected by our sequential feature selection algorithm. We again find that *MFCCs* are the dominant features for all categories of audio excerpt.

We observe similar results for same make and model microphones as shown in Table 3.8. Even when we test audio segments that combine features from both the phone's built-in speaker and microphone we see that *MFCCs* are still the dominant features among all the acoustic features (shown in Table 3.9).

One thing that is noticeable — in general the F-score for same make and model devices is lower compared to what we get for different make and model devices. This is understandable as same make and model devices contain the same brand of speakers and microphones whereas different make and model devices might contain different brands of speakers and microphones. Thus, it is only trivial that fingerprinting same brand of speakers and/or microphones is going to be a harder problem.

To get a better understanding of why *MFCCs* are the dominant acoustic

Table 3.6: Feature exploration for different make and model smartphones using both speaker and microphone.

#	Feature	Fingerprinting Speakers and Microphones Maximum F-Score (%)					
		Instrumental		Human Speech		Song	
		k-NN	GMM	k-NN	GMM	k-NN	GMM
1	RMS	93.1	92.7	89	80.8	93.1	96.3
2	ZCR	83.3	83.3	84.6	78.8	93.1	96.3
3	Low-Energy-Rate	93.1	90	81.7	77.7	96.3	96.3
4	Spectral Centroid	82.3	85.1	40.5	42.7	74.2	76.1
5	Spectral Entropy	78.3	72.4	81	67.9	96.3	96.3
6	Spectral Irregularity	74.6	72.7	55.4	53.6	90.1	81.1
7	Spectral Spread	86.5	86.5	92.7	92.4	93.1	93.1
8	Spectral Skewness	92.3	89.7	84.6	86.2	96.3	96.3
9	Spectral Kurtosis	89.5	86.3	60.7	59.5	81.6	85.3
10	Spectral Rolloff	**100**	96.3	92.7	92.7	**100**	96
11	Spectral Brightness	80.4	79.4	81.6	67.6	87.4	90.2
12	Spectral Flatness	85.1	85.1	**96.3**	92.7	**100**	96.3
13	MFCCs	93.1	**100**	**96.3**	**96.3**	92.7	**100**
14	Chromagram	96.3	93.1	88.6	**96.3**	86.5	**100**
15	Tonal Centroid	96.3	**100**	**96.3**	**96.3**	**100**	**100**
Sequential Feature Selection		[10]	[13]	[12]	[13]	[10]	[13]
Max F-Score		100	100	96.3	96.3	100	100

features we plot the MFCCs of a given audio excerpt from three different handsets on Figure 3.8. All the coefficients are ranked in the same order for the three handsets. We can see that the magnitude of the coefficients vary across the handsets. For example, coefficient 3 and 5 vary significantly across the three handsets. Hence, MFCCs contain high degree of variability making it the dominant feature for fingerprinting smartphones.

3.6.3 Feature Exploration for Large Pool of Devices

Lastly, we look at features from all the devices in our collection. In this case we combine microphone and speaker to generate the auditory fingerprint for smartphones. We do so because in the previous sections we found that combining speaker and microphone yields the highest accuracy. To collectively fingerprint smartphones using both the embedded microphone and speaker we use our android app to play and record audio clips simultaneously. We, therefore, limit ourself to only android devices for this experiment. In our

Table 3.7: Feature exploration for same make and model smartphones using only speaker.

#	Feature	Fingerprinting Speakers Maximum F-Score (%)					
		Instrumental		Human Speech		Song	
		k-NN	GMM	k-NN	GMM	k-NN	GMM
1	RMS	34.9	33.8	16.6	12.3	20	25.7
2	ZCR	29.7	26.5	12.2	14.4	13	7.1
3	Low-Energy-Rate	12.5	14.8	15	5.7	21.8	18.7
4	Spectral Centroid	28	30.5	12.2	19	39.9	40.3
5	Spectral Entropy	20.9	19.8	14.2	16.6	33.9	26.3
6	Spectral Irregularity	14.5	11.7	7.4	14.7	11.8	17.5
7	Spectral Spread	36.4	43.7	11.3	14.3	35.2	38.4
8	Spectral Skewness	33.9	29.1	13.3	15.5	31.5	40.3
9	Spectral Kurtosis	30.5	29.1	11.6	16	31.1	36.8
10	Spectral Rolloff	40.4	39	14.9	14.3	38.7	41.1
11	Spectral Brightness	32.1	31.6	18.9	21.8	18.5	17.9
12	Spectral Flatness	34.9	31	19.8	13.3	32.4	30
13	MFCCs	**90.4**	**96.5**	**91.3**	**97.5**	**90**	**91.4**
14	Chromagram	79.1	70.6	72.9	66	80.6	80
15	Tonal Centroid	77	60	65.4	53.4	63.6	53.8
Sequential Feature Selection		[13,14]	[13,14]	[13]	[13,14]	[13,7]	[13,14]
Max F-Score		97.5	97.7	93.7	98.2	91.5	92.9

collection we had a total of 50 android devices (iphone5 and Sony Ericsson W518 were the two non-android devices in our collection). Table 3.10 highlights our findings. We see that again *MFCCs* are the dominant features for all categories of audio excerpt. This is expected as we saw similar outcomes in Table 3.6 and Table 3.9.

3.7 Experimental Evaluations

We perform a series of experiments to evaluate how well we can fingerprint smartphones by exploiting the manufacturing imperfections of microphones and speakers embedded in them. We look at fingerprinting devices, first, of different make and model, followed by devices of same make and model, and finally a combination of both with multiple units of different models. Note that the audio excerpts used for feature exploration and the ones used for evaluating our fingerprinting approach in this section are not identical. We use different audio excerpts belonging to the same three categories listed in

Table 3.8: Feature exploration for same make and model smartphones using only microphone.

#	Feature	Fingerprinting Microphones Maximum F-Score (%)					
		Instrumental		Human Speech		Song	
		k-NN	GMM	k-NN	GMM	k-NN	GMM
1	RMS	40.2	36.9	19.6	20.1	23.5	28.6
2	ZCR	22.7	29.6	26.2	22.6	44.5	41.9
3	Low-Energy-Rate	22.6	24.8	5.2	7.4	10.7	13.4
4	Spectral Centroid	17.3	24.8	16.6	12.9	33.7	35.7
5	Spectral Entropy	29.1	22.2	15.2	15.1	40.3	36
6	Spectral Irregularity	12.6	16.3	13.2	17.9	15.8	18.6
7	Spectral Spread	17.2	22.6	16.4	14.9	36.2	34.8
8	Spectral Skewness	31.8	28.1	20.8	13.7	38	43.1
9	Spectral Kurtosis	28.5	26.1	20.3	14	45.8	39.2
10	Spectral Rolloff	30	32.8	15.1	11.6	46.1	44
11	Spectral Brightness	22.5	20.3	12.6	16	33.1	27.4
12	Spectral Flatness	24.6	23.8	17.2	12.2	39.2	35.5
13	MFCCs	**89**	**93.5**	**98.8**	**96.2**	**94.1**	**97.5**
14	Chromagram	71.5	55.3	75	88.7	87.3	85.3
15	Tonal Centroid	67.8	51.3	70	70.8	83.1	79.4
Sequential Feature Selection		[13,8,12]	[13,8,12]	[13]	[13,14,2]	[13,14,10]	[13,14]
Max F-Score		93	96.7	98.8	97.5	96.3	97.9

Table 3.2, so as to *not bias* our evaluations. All the evaluations are done with 50% of the samples (randomly chosen) being used for training and the remaining 50% for testing.

3.7.1 Fingerprinting Different Make and Model Devices

First, we look at fingerprinting devices of different make and model. So, we take one representative smartphone from each row of Table 3.1 giving us a total of 7 different phones. We test our fingerprinting approach using all three types of audio excerpt. To generate fingerprints we only use the acoustics features obtained from our sequential feature selection algorithm as listed in Tables 3.4, 3.5 and 3.6. Table 3.11 summarizes our findings. From Table 3.11 we see that we can successfully, with an F-score of 100%, identify which audio clip originated from which smartphone. Similar to speaker, we also find that microphone properties differ quite substantially across vendors. We see that by using only the microphone to fingerprint smartphones we can achieve an F-score of over 97%. Combining microphone with speaker the

Table 3.9: Feature exploration for same make and model smartphones using both speaker and microphone.

#	Feature	Fingerprinting Speakers and Microphones Maximum F-Score (%)					
		Instrumental		Human Speech		Song	
		k-NN	GMM	k-NN	GMM	k-NN	GMM
1	RMS	87.2	83	92.1	93	89.5	89.2
2	ZCR	59.8	60.5	56.8	58.4	67.6	75.2
3	Low-Energy-Rate	67.4	70.9	30.1	36.7	69.7	64.5
4	Spectral Centroid	30.1	27.5	25.7	30.1	35.1	32.7
5	Spectral Entropy	78.5	70.3	52.9	54.8	81.8	81.8
6	Spectral Irregularity	63.9	54.6	32.5	33.6	62.5	67.1
7	Spectral Spread	84.6	81.3	67.6	62.8	87	87.4
8	Spectral Skewness	85.7	88.3	58.7	54.4	70.9	68.9
9	Spectral Kurtosis	80.3	80.6	51.9	49.9	82.2	76.2
10	Spectral Rolloff	79.4	73.4	46.9	51.5	77.2	71.8
11	Spectral Brightness	86	88	75.2	69.2	87.5	79.5
12	Spectral Flatness	79.8	79	45.5	45.4	86.4	87.2
13	MFCCs	**100**	**100**	**98.7**	**100**	**100**	**100**
14	Chromagram	98.8	95.8	97.6	**100**	**100**	96.5
15	Tonal Centroid	98.8	94.8	95.2	92.7	**100**	98.8
Sequential Feature Selection		[13]	[13]	[13]	[13]	[13]	[13]
Max F-Score		100	100	98.7	100	100	100

F-score bumps back up to 100%.[2] Thus, a malicious app having access to speaker and/or microphone can successfully fingerprint smartphones using only a few acoustic features.

3.7.2 Fingerprinting Same Make and Model Devices

We now look at fingerprinting the 15 Motorola Droid A855 handsets. Table 3.12 highlights our findings. We test our fingerprinting approach against three different forms of audio excerpt. We use the acoustic features obtained from our sequential feature selection algorithm as listed in Tables 3.7, 3.8 and 3.9. From Table 3.12, we see that we can achieve an F-score of over 94% in identifying which audio clip originated from which handset using only the speaker as our source for generating the fingerprints. Using only microphones we can bump up the F-score to 95%. However, we obtain the best

[2]For fingerprinting smartphones through both microphone and speaker we use our android app for data collection. And as a result we exclude iPhone5 and Sony Ericsson W518 handset from this experiment, reducing the pool of handsets to 5 devices.

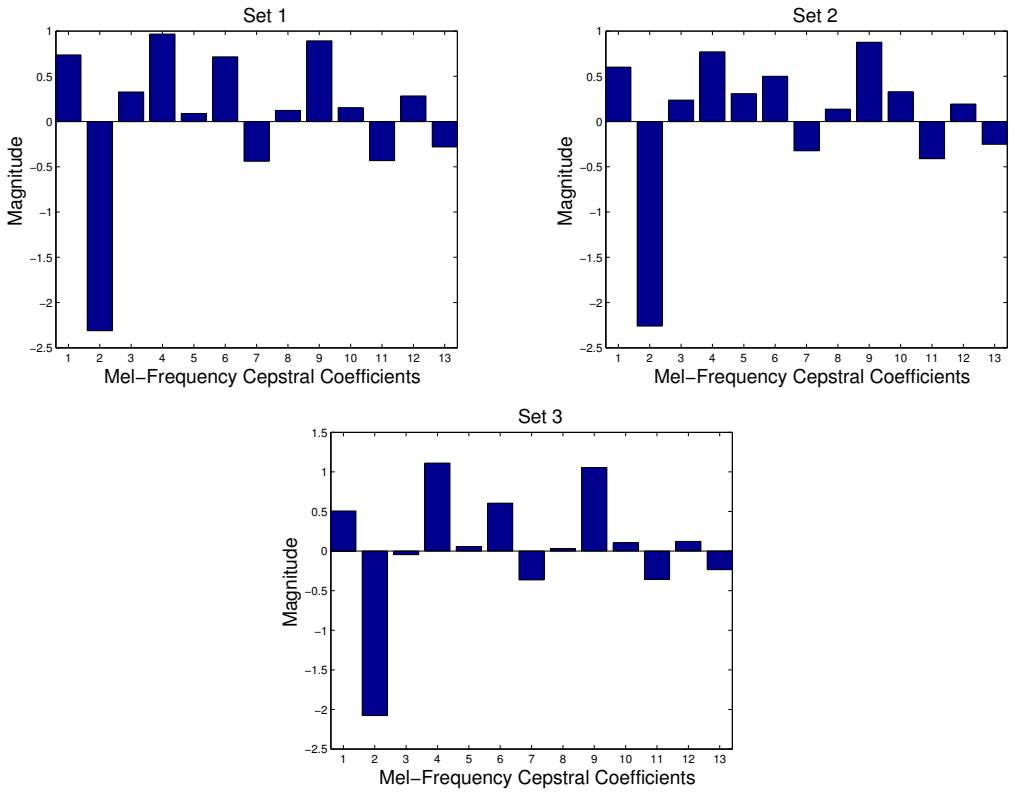


Figure 3.8: MFCCs for a given audio sample taken from three different handsets manufactured by the same vendor. We can see that some of the coefficients vary significantly, thus enabling us to exploit this feature to fingerprint smartphones.

result when we consider fingerprinting both the speaker and microphone. In that case we achieve an F-score of 100%. In other words we were able to fingerprint all test samples correctly when we combined anomalies from both the embedded microphone and speaker. So, if a malicious app can get access to the speaker (which does not require explicit permission) and microphone (which may require explicit permission, but many games nowadays require access to microphone anyway) it can successfully track individual devices.

3.7.3 Fingerprinting All Make and Model Devices

Lastly, we evaluate how effectively we can fingerprint the 50 android smartphones in our collection. The setting is similar to all the previous experiments where each audio clip is recorded 10 times, 50% of which is used for training and the remaining 50% for testing. We use our android app to collect all

Table 3.10: Feature exploration for 50 android smartphones.

#	Feature	Maximum F-Score (%)					
		Instrumental		Human Speech		Song	
		k-NN	GMM	k-NN	GMM	k-NN	GMM
1	RMS	82.7	80	87.3	84	78.7	76.8
2	ZCR	51.3	48.2	50.3	45.9	48.5	45.9
3	Low-Energy-Rate	45.2	40.6	19.4	15.4	31.9	33.8
4	Spectral Centroid	35.6	34.7	23.7	25.8	25.7	30.1
5	Spectral Entropy	56.2	60.8	46.3	48.1	67.7	67.7
6	Spectral Irregularity	46.1	47	25.9	23.6	26.9	35.3
7	Spectral Spread	57.4	57	54.2	49.7	70.9	74.1
8	Spectral Skewness	50.3	53.9	34.5	32.5	52.7	59.9
9	Spectral Kurtosis	45	47.7	37.1	38.6	51.5	54.2
10	Spectral Rolloff	49.5	53.5	48.4	45.9	59.1	62.8
11	Spectral Brightness	52.1	54.5	38.1	35.3	59.2	61.7
12	Spectral Flatness	61	60.1	61.6	63.4	67.3	68.3
13	MFCCs	**100**	**100**	**100**	**99.6**	**100**	**99.6**
14	Chromagram	96.2	93.4	98.9	95.8	99.6	98.2
15	Tonal Centroid	96	89	95.5	91.8	98.5	98.5
Sequential Feature Selection		[13]	[13]	[13]	[13]	[13]	[13]
Max F-Score		100	100	100	99.6	100	99.6

Table 3.11: Fingerprinting different make and model devices.

Hardware	Avg. F-Score (%)					
	Instrumental		Human Speech		Song	
	k-NN	GMM	k-NN	GMM	k-NN	GMM
Speaker	97.4	100	94.8	100	97.4	100
Microphone	94.8	100	94.8	97.4	97.4	100
Speaker+Microphone	96.3	100	96.3	100	96.3	100

Table 3.12: Fingerprinting same make and model devices.

Hardware	Avg. F-Score (%)					
	Instrumental		Human Speech		Song	
	k-NN	GMM	k-NN	GMM	k-NN	GMM
Speaker	96.3	98.3	98.8	98.8	92.6	94.5
Microphone	95.3	95.3	98.8	100	96.2	96.1
Speaker+Microphone	100	100	100	100	100	100

the audio samples. Table 3.13 summarizes our fingerprinting results. We see that we can obtain an F-score of over 98% in fingerprinting all the 50 smartphones. This result suggests that fingerprinting smartphones via microphones and speakers is truly feasible.

Table 3.13: Fingerprinting heterogeneous devices.

Hardware	Avg. F-Score (%)					
	Instrumental		Human Speech		Song	
	k-NN	GMM	k-NN	GMM	k-NN	GMM
Speaker+Microphone	99	98.3	99.6	99.3	99.6	100

3.8 Sensitivity Analysis

In this section we investigate how different factors such as audio sampling rate, training set size, the distance between audio-source and recorder, and background noise impact our fingerprinting accuracy. Such investigations will help us determine the conditions under which our fingerprinting approach will be feasible, specially if the attacker is tracking devices in public locations. For the following set of experiments we only focus on fingerprinting similar model smartphones from the same vendor (as this has been shown to be a tougher problem in the previous sections) and consider only fingerprinting speakers as this is applicable to the scenario where the attacker is tracking devices in public locations. We also consider recording only ringtones (i.e., audio clips belonging to our defined 'Instrumental' category in Table 3.2) for the following experiments. Since we are recording ringtones, we use the features highlighted in Table 3.7 under the 'Instrumental' category.

3.8.1 Impact of Sampling Rate

First, we investigate how the sampling rate of audio signals impacts our fingerprinting precision. To do this, we record a ringtone at the following three frequencies: 8 kHz, 22.05 kHz and 44.1 kHz. Each sample is recorded 10 times with half of them being used for training and the other half for testing. Figure 3.9 shows the average precision and recall obtained under different sampling rates. As we can see from the figure, as sampling frequency decreases, the precision/recall also goes down. This is understandable, because the higher the sampling frequency the more fine-tuned information we have about the audio sample. However, the default sampling frequency on most hand-held devices today is 44.1 kHz [112], with some of the latest models adopting even higher sampling rates [113]. We, therefore, believe sampling rate will not impose any obstacles for our fingerprinting approach, and in

future we will be able to capture more fine grained variations with the use of higher sampling rates.

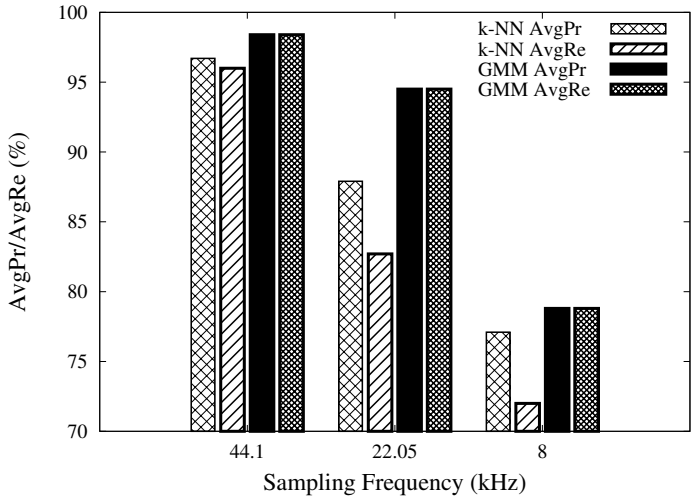


Figure 3.9: Impact of sampling frequency on precision/recall.

3.8.2 Impact Training Set Size

Next, we consider performance of the classifiers in the presence of limited training data. For this experiment we vary the training set size from 10% to 50% (i.e., from 1 to 5 samples per device) of all available samples. Figure 3.10 shows the evolution of the F-score as training set size is increased. We see that as the training set size increases the F-score also rises which is expected. However, we see that even with only three samples per device we can achieve an F-score of over 90%. This suggests that we do not need too many training samples to construct a good predictive model.

3.8.3 Impact of Distance between Speaker and Recorder

Now, we inspect how fingerprinting accuracy degrades as the distance between the audio source (i.e., smartphone) and recorder (i.e., laptop/PC) is varied. For this experiment we use a separate external microphone as the signal capturing capacity of the microphone embedded inside a laptop degrades drastically as distance increases. We use the relatively inexpensive ($44.79) Audio-Technica ATR-6550 shotgun microphone for this experiment

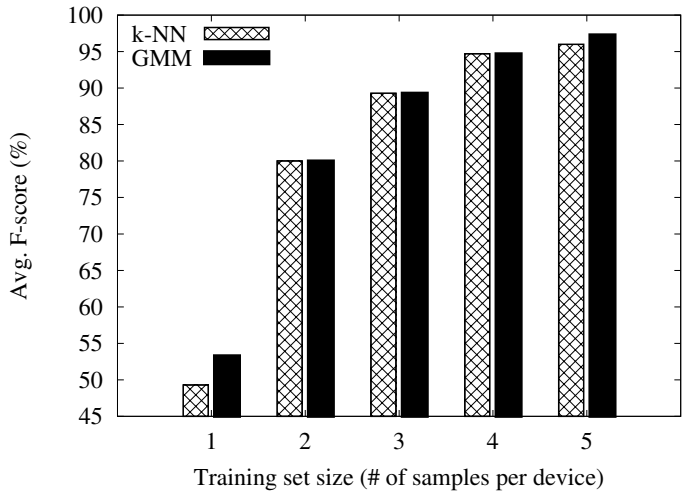


Figure 3.10: Impact of varying training set size on accuracy.

and vary the distance between the external microphone and smartphone from 0.1 meter to 5 meters. Figure 3.11 summarizes how F-scores changes as the distance between the smartphone and microphone is varied. We see that as distance increases, F-score decreases. This is expected, because the longer the distance between the smartphone and microphone, the harder it becomes to capture the minuscule deviations between audio samples. However, we see that even up to two meters we can achieve an F-score of around 93%. This suggests that our device fingerprinting approach works only up to a certain distance using off-the-shelf inexpensive commercial microphones. However, using specialized microphones, such as parabolic microphones (usually used in capturing animal sounds from a far distance), could help increase the fingerprinting precision even at longer distances.

3.8.4 Impact of Ambient Background Noise

In this section we investigate how ambient background noise impacts the performance of our fingerprinting technique. For this experiment we consider scenarios where there is a crowd of people using their smartphones and we are trying to fingerprint those devices by capturing audio signals (in this case ringtones) from the surrounding environment. Table 3.14 highlights the four different scenarios that we consider. To emulate such environment, external speakers (2 pieces) are placed between the smartphone and microphone while

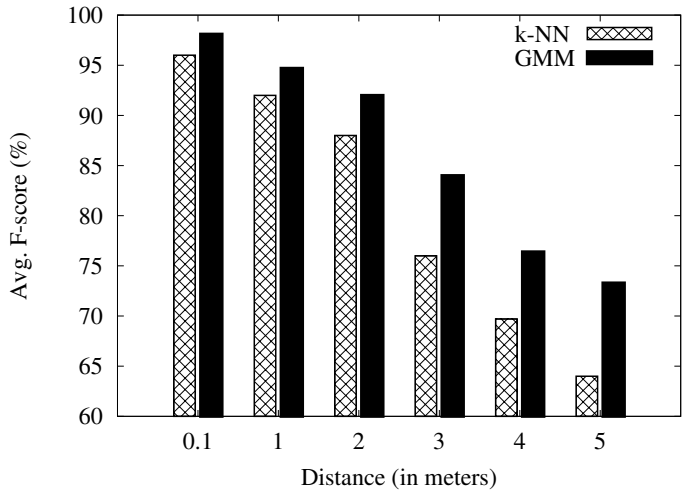


Figure 3.11: Impact of varying the distance between smartphone and microphone.

recording is taking place. The external speakers are constantly replaying the respective ambient noise in the background. We consider a distance of two meters from the audio source to recorder as shown in Figure 3.12.

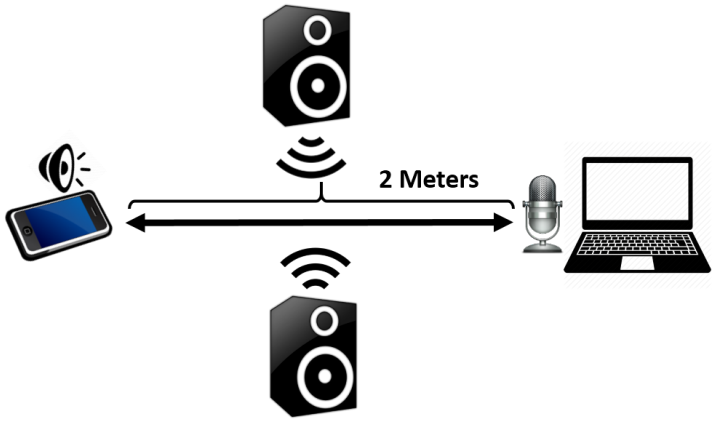


Figure 3.12: Experimental setup for determining the impact of ambient background noise.

The ambient background sounds were obtained from PacDV [114] and SoundJay [115]. We also compute the signal-to-noise (SNR) ratio between the original ringtone and the different ambient background noise. The RMS (root-mean-square) value of the different background noise varied from approximately 13% (17.77 dB) to 18% (14.92 dB) of the RMS value of the

ringtone under consideration. Table 3.14 shows our findings (values are reported as percentages). We can see that even in the presence of various background noise we can achieve an F-score of over 91%.

Table 3.14: Impact of ambient background noise.

Environments	SNR (dB)	k-NN			GMM		
		Features [13,14]*			Features [13,14]*		
		$AvgPr$	$AvgRe$	$AvgF1$	$AvgPr$	$AvgRe$	$AvgF1$
Shopping Mall	15.85	88.8	85.3	87	95.1	93.3	94.2
Restaurant/Cafe	17.77	90.5	89.7	90.1	92.5	90.7	91.6
City Park	15.43	91.7	90	90.8	95.2	94.1	94.6
Airport Gate	14.92	91.3	89.5	90.4	94.5	93.3	93.9

∗ Feature numbers taken from Table 3.7

3.9 Limitations

Our approach has a few limitations. First, we experimented with 52 devices manufactured by different vendors; it is possible that a larger target device pool would result in lower accuracy. That said, distinctions across different device types are more clear; additionally, audio fingerprints may be used in tandem with other techniques, such as accelerometer fingerprinting [83], to better discriminate between devices. Secondly, most of the experiments took place in a lab setting. However, we studied the impact of ambient background noise and still found our approach to be applicable. Lastly, all the phones used in our experiments were not in mint condition and some of the idiosyncrasies of individual microphones and speakers may have been the result of uneven wear and tear on each device; we believe, however, that this is likely to occur in the real world as well.

3.10 Summary

In this chapter we show that it is feasible to fingerprint smartphones through onboard acoustic components like microphones and speakers. As microphone and speaker are one of the most standard components present in almost all smartphones available today, this creates a key privacy concern for users. To demonstrate the feasibility of this approach, we collect fingerprints from

52 different phones covering a total of five different brands of smartphones. Our studies show that it is possible to successfully fingerprint smartphones through microphones and speakers, not only under controlled environments, but also in the presence of ambient noise. We believe our findings are important steps towards understanding the full consequences of fingerprinting smartphones through acoustic channels.

CHAPTER 4

FINGERPRINTING SMARTPHONES VIA MOTION SENSORS

Motion sensors play a critical role in making smartphones smart. It is because of motion sensors that users can enjoy sophisticated applications such as 3-D gaming, augmented reality and fitness monitoring. Motions sensors provide consumers with a more interactive user experience. Gesture and activity based applications are quickly gaining popularity among consumers. However, these same sensors can be used as side-channels to uniquely track smartphones. Disturbingly, access to motions sensors is deemed non-sensitive and thus requires no explicit user permission for accessing them. As a result, JavaScript embedded in any public web page can easily and surreptitiously access these sensors while the user is browsing the web page. This would enable any website to fingerprint the manufacturing imperfections of these sensors and thereby track physical devices across multiple visits.

4.1 Overview

First, we start with an overview of our approach and describe the attack scenario. Motion sensors such as accelerometers and gyroscopes are available not only to installed applications but also to HTML5 [116]. And interestingly, websites do not require any explicit permission to access these motion sensors. So our attack scenario consists of setting up a web page to surreptitiously collect accelerometer and gyroscope data as shown in Figure 4.1. All that is needed is a small block of JavaScript to access and transmit the sensor data to our server. From the collected data we then aim to "pull out" the imperfections in sensor circuitry. Manufacturing imperfections result in each sensor having unique characteristics in their produced signal. These characteristics can be captured in the form of a fingerprint and be used to track users across multiple websites. However, practical fingerprinting faces

several challenges. During a typical web browsing session, a smartphone is either held in a user's hand, resulting in noisy motion inputs, or is resting on a flat surface, minimizing the amount of sensor input. Additionally, web APIs for accessing motion sensor data have significantly lower resolution than what is available to the operating system and native applications. We show that, using machine learning techniques, it is possible to combine a large number of features from both the accelerometer and gyroscope sensor and produce highly accurate classification despite these challenges. In some cases, we can improve the classifier accuracy by using an inaudible sound, played through the speakers, to stimulate the motion sensors. We evaluate our techniques in a variety of lab settings; additionally, we collected data from volunteer participants over the web, capturing a wide variety of smartphone models and operating systems. In our experiments, a web browsing session lasting in the orders of 25–30 seconds is sufficient to generate a fingerprint that can be used to recognize the phone in the future with only 5–6 seconds worth of web browsing session.

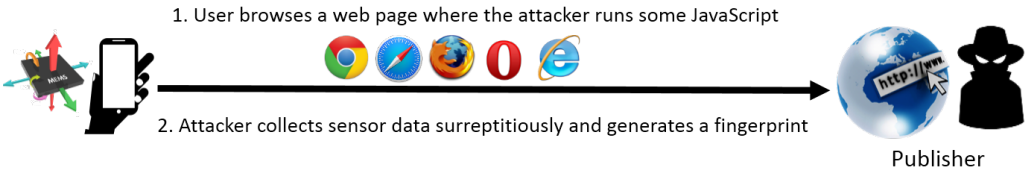


Figure 4.1: Fingerprinting motion sensors through HTML5.

4.2 Data Collection Setup and Data Processing

In this section we will depict our data collection process. We will also describe our data processing steps.

4.2.1 Data Collection Setup

Given that mobile accounts for a third of all global web pages served [117], our data collection process consists of developing our own web page to collect sensor data.[1] We obtain IRB approval for collecting sensor data. Our web

[1]http://datarepo.cs.illinois.edu/DataCollectionHowPlaced.html

page contains a JavaScript to access motion sensors like accelerometer and gyroscope. We create an event listener for device motion in the following manner:

```
window.addEventListener('devicemotion',motionHandler)
```

Once the event listener is registered, the `motionHandler` function can access accelerometer and gyroscope data in the following manner:

```
function motionHandler(event){
  // Access Accelerometer Data
  ax = event.accelerationIncludingGravity.x;
  ay = event.accelerationIncludingGravity.y;
  az = event.accelerationIncludingGravity.z;
  // Access Gyroscope Data
  rR = event.rotationRate;
  if (rR != null){
    gx = rR.alpha;
    gy = rR.beta ;
    gz = rR.gamma;
  }
}
```

However, since we collect data through the browser the maximum obtainable sampling frequency is lower than the available hardware sampling frequency (restricted by the underlying OS). Table 4.1 summarizes the sampling frequencies obtained from the top 5 mobile browsers [118].[2] We use a Samsung Galaxy S3 and iPhone 5 to test the sampling frequency of the different browsers. Table 4.1 also highlights the motion sensors that are accessible from the different browsers. We see that Chrome provides the best sampling frequency on both platforms while the default Android browser is the most restrictive browser in terms of not only sampling frequency but also access to different motion sensors. Chrome being the most popular mobile browser [119], we collect data using the Chrome browser. One thing to remember is that the sample rate available at any instance of time depends on multiple factors such as the current battery life and the number of applications running in the background.

[2]Computed the average time to obtain 100 samples. http://datarepo.cs.illinois.edu/SamplingFreq.html

Table 4.1: Sampling frequency from different browsers.

OS	Browser	Sampling Frequency (∼Hz)	Accessible Sensors*
Android 4.4	Chrome	100	A,G
	Android	20	A
	Opera	100	A,G
	UC Browser	20	A,G
	Standalone App [120]	200	A,G
iOS 8.1.3	Safari	100	A,G
	Chrome	100	A,G
	Standalone App [121]	100	A,G

* **'A'** means accelerometer and **'G'** refers to gyroscope

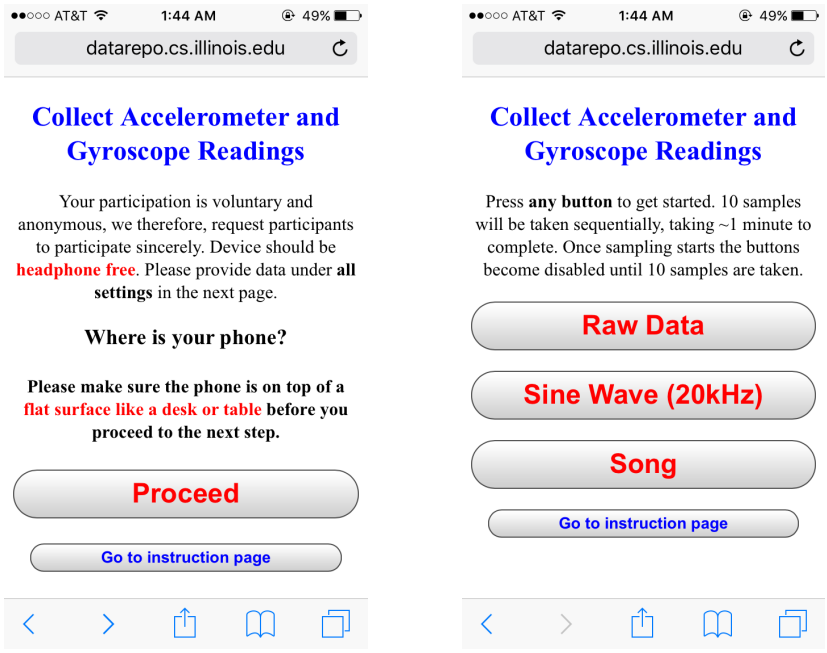
Now, since our fingerprinting approach aims to capture the inherent imperfections of motion sensors, we need to keep the sensors stationary while collecting data. Therefore, by default, we have the phone placed flat on a surface while data is being collected, unless explicitly stated otherwise. This mimics the scenario where the user has placed his/her smartphone on a desk while browsing a web page. We, however, do test our approach for the scenario where the user is holding the smartphone in his/her *hand while sitting down quietly*. Also, as gyroscopes react to audio stimulation we collect data under three different background audio-settings. Table 4.2 describes the three types of audio stimulation. For the latter two audio stimulations the corresponding audio file is played in the background of the browser while data is being collected. Under each setting we collect 10 samples where each sample is about 5 to 6 seconds worth of data (so, total data collection time is in the range of 1 minute per setting). Our web page collects all sensor data is in the background without interfering with the user's browsing experience and then sends the data to our back-end server for analysis. For the purpose of labeling our data we plant a unique random number inside the cookie. This provides us with ground truth data, thus, making it possible to correlate data samples coming from the same physical device.[3]

Screenshots of our data collection website is provided in Figure 4.2. As you can see from the figure users are first asked to place the device on a flat surface before proceeding to the next step of the data collection process.

[3]It is possible that users cleared this cookie, but we do not expect this to happen with enough frequency to significantly affect our data.

Table 4.2: Types of background audio stimulation.

Type	Description
No-audio	No audio stimulation present
Sine	20 kHz sine wave played through the speaker
Song	A popular song played through the speaker



(a) Instruction page (b) Settings for data collection

Figure 4.2: Screenshots of our data collection website. Users are first asked to place the device on a flat surface before selecting a specific background audio-stimulation.

4.2.2 Data Processing

Data from motion sensors can be thought of as a stream of timestamped real values. For both accelerometer and gyroscope we obtain values along three axes. So, for a given timestamp, t, we have two vectors of the following form: $\vec{a}(t) = (a_x, a_y, a_z)$ and $\vec{\omega}(t) = (\omega_x, \omega_y, \omega_z)$. The accelerometer values include gravity, i.e., when the device is stationary lying flat on top of a surface we get a value of 9.81 $\mathtt{ms}^{-2}$ along the z-axis. We convert the acceleration vector into a scalar by taking its magnitude: $|\vec{a}(t)| = \sqrt{a_x^2 + a_y^2 + a_z^2}$. This technique discards some information, but has the advantage of making the accelerometer data independent of device orientation; i.e., if the device is stationary the acceleration magnitude will always be around 9.81 $\mathtt{ms}^{-2}$, whereas

the reading on each individual axis will vary greatly (by +/- $1g$) depending on how the device is held. For the gyroscope we consider data from each axis as a separate stream, since there is no corresponding baseline rotational speed. In other words, if the device is stationary the rotation rate along all three axes should be close to 0 $\texttt{rads}^{-1}$, irrespective of the orientation of the device. Thus, our model considers four streams of sensor data in the form of $\{|\vec{a}(t)|, \omega_x(t), \omega_y(t), \omega_z(t)\}$.

For all data streams, we also look at frequency domain characteristics. But since the browser, running as one of many applications inside the phone, makes API calls to collect sensor data the OS might not necessarily respond in a synchronized manner.[4] This results in non-equally spaced data points. We, therefore, use cubic-spline interpolation [122] to construct new data points such that $\{|\vec{a}(t)|, \omega_x(t), \omega_y(t), \omega_z(t)\}$ become equally-spaced.

4.3 Temporal and Spectral Features

To summarize the characteristics of a sensor data stream, we explore a total of 25 features consisting of 10 temporal and 15 spectral features (listed in Table 4.3). As we have four data streams, we have a total of 100 features to summarize the unique characteristics of the motion sensors. A brief description of each feature follows, but since many of the features overlap with the features described in Section 3.4, we refrain ourselves from describing the overlapping features.

Mean Signal Value: This feature computes the arithmetic mean of a signal amplitude. In the case of a set of N values $\{x_1, x_2, \ldots, x_N\}$, the mean value is given by the following formula:

$$\mu = \frac{1}{N}\left(x_1 + x_2 + \cdots + x_N\right) \tag{4.1}$$

The mean value provides an approximation of the average signal strength.

Signal Standard Deviation: This feature computes the dispersion in signal strength. For a set of N values $\{x_1, x_2, \ldots, x_N\}$, the standard deviation

[4]Depending on current load and priority of other running applications, OS might prioritize such API calls differently.

Table 4.3: Explored temporal and spectral features.

#	Domain	Feature
1	Time	Mean
2		Standard Deviation
3		Average Deviation
4		Skewness
5		Kurtosis
6		RMS
7		Max
8		Min
9		ZCR
10		Non-Negative count
11	Frequency	Spectral RMS
12		Low-Energy-Rate
13		Spectral Centroid
14		Spectral Entropy
15		Spectral Irregularity
16		Spectral Spread
17		Spectral Skewness
18		Spectral Kurtosis
19		Spectral Rolloff
20		Spectral Brightness
21		Spectral Flatness
22		Spectral Roughness
23		Spectral Flux
24		Spectral Attack Time
25		Spectral Attack Slope

is given by the following formula:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2} \tag{4.2}$$

where μ refers to the mean signal strength. Standard deviation measures the spread of a signal strength.

Average Deviation: This feature measures the average distance from mean. In the case of a set of N values $\{x_1, x_2, \ldots, x_N\}$, the average deviation is computes using the following formula:

$$AvgDev = \frac{1}{N}\sum_{i=1}^{N}|x_i - \mu| \tag{4.3}$$

where μ refers to the mean signal strength.

Skewness: This feature measures asymmetry around mean. For a set of N values $\{x_1, x_2, \ldots, x_N\}$, the skewness is computed as:

$$\gamma_1 = \frac{1}{N}\left(\sum_{i=1}^{N}(\frac{x_i - \mu}{\sigma})^3\right) \tag{4.4}$$

where μ and σ respectively represents the mean and standard deviation of signal strength.

Kurtosis: This feature measures the flatness or spikiness of a distribution. For a set of N values $\{x_1, x_2, \ldots, x_N\}$, the kurtosis is computed as:

$$\beta_1 = \frac{1}{N}\left(\sum_{i=1}^{N}(\frac{x_i - \mu}{\sigma})^4\right) \tag{4.5}$$

where μ and σ respectively represents the mean and standard deviation of signal strength.

Spectral Roughness: Spectral roughness computes the average of the dissonance between all possible pairs of peaks in a spectrum [123, 124].

Spectral Flux: Spectral flux is a measure of how quickly the power spectrum of a signal changes. It is calculated by taking the average Euclidean distance between the power spectrum of two contiguous frames [124].

Spectral Attack Time: This features computes the average rise time to spectral attacks where spectral attacks are local maxima in the spectrum [124].

Spectral Attack Slope: This features computes the average slope to spectral attacks where spectral attacks are local maxima in the spectrum [124].

4.4 Classification Algorithms and Evaluation Metrics

Let us briefly discuss the classification algorithms and evaluation metrics used for fingerprinting smartphones before we proceed to the evaluation section.

4.4.1 Classification Algorithms

Once we have features extracted from the sensor data, we use supervised learning algorithms to identify the source sensor. Any supervised learning classifier has two main phases: training phase and testing phase. During training, features from all smartphones (i.e., labeled data) are used to train the classifier. In the test phase, the classifier predicts the most probable class for a given (unseen) feature vector. We evaluate the performance of the following classifiers — Support Vector Machine (SVM), Naive-Bayes classifier, Multiclass Decision Tree, k-Nearest Neighbor (k-NN), Quadratic Discriminant Analysis classifier and Random Forest (MATLAB's Treebagger model [125]). In general, we found that ensemble based approaches like random forest outperforms other classifiers. We report the maximum achievable accuracies from these classifiers.

For training and testing the classifiers we *randomly* split the dataset in such a way that 50% of data from each device goes to the training set while the remaining 50% goes to the test set. To prevent any bias in the selection of the training and testing set, we randomize the training and testing set 10 times and report the average F-score. We also compute the 95% confidence interval, but we found it to be less than 1% in most cases and hence do not report them in such cases. For analyzing and matching fingerprints we use a desktop machine with an Intel i7-2600 3.4 GHz processor with 12 GiB RAM. We found that the average time required to match a new fingerprint was around 10–100 `ms`.

4.4.2 Evaluation Metrics

For evaluation metrics we use the same set of standard multiclass classification metrics as described in Section 3.5.2. That is we first compute *precision* and *recall*, and then take the harmonic mean of precision and recall to compute *F-score*. We use the F-score to report the effectiveness of our fingerprinting approach.

4.5 Experimental Setup

Our experimental setup consists of collecting data from both lab setting and public setting. In the lab setting we collect data from lab phones in our office. For the public setting we request participants to visit our web page with their smartphone from wherever they desire.

4.5.1 Lab Setting

We kick off our data collection by gathering data from 30 lab phones. Table 4.4 lists the distribution of the different smartphones from which we collect sensor data.

Table 4.4: Types of lab phones used.

Maker	Model	Quantity
Apple	iPhone 5	4
	iPhone 5s	3
Samsung	Nexus S	14
	Galaxy S3	4
	Galaxy S4	5
Total		30

For the lab setting we collect data under all three audio stimulations (as described in Table 4.2). Also, data is collected for both when the device in placed flat on top of a surface as well as when it is placed in the hand of the user while sitting down.

4.5.2 Public Setting

Next, we expand our data collection process to cover real world public settings. We invite people to voluntarily participate in our study. Participants are asked to visit our web page and follow a few simple steps to provide us with sensor data. We recruit participants through institutional mass email and online social networks. We asked participants to provide data under two settings: no-audio setting and the inaudible sine-wave setting (we avoid the background song setting to make the experience less bothersome for the user). Each setting collected sensor data for about one minute, requiring a total of two minutes of participation. Over the course of two weeks, we

received data from a total of 76 devices. However, some participants did not follow all the steps and as a result we were able to use only 63 of the 76 submissions. Figure 4.3 shows the distribution of the different devices that participated in our study.[5] We can see from Figure 4.3 that our public dataset covers a diverse set of smartphones covering majoring vendors like Apple and Samsung.

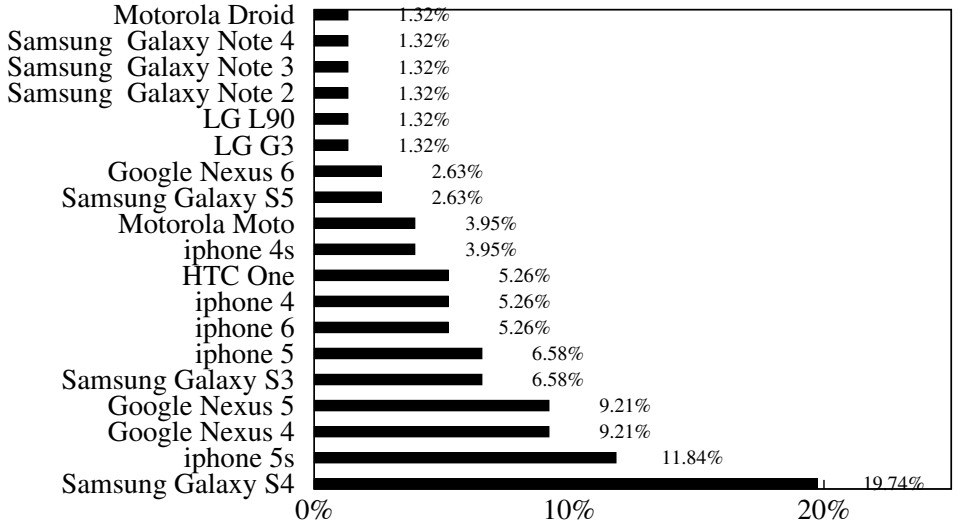


Figure 4.3: Distribution of participant device models.

4.5.3 Analytic Tools

To extract spectral features we use the following signal-analytic tools and modules: *MIRtoolbox* [91] and *Libxtract* [126]. For feature selection we use the FEAST toolbox [127]. Lastly, for classifiers we use MATLAB's Machine Learning Toolbox [125].

4.6 Feature Exploration

We perform feature exploration to determine if inclusion of too many features worsens performance due to their varying accuracies and potentially conflicting signatures. We, therefore, explore all the features and determine

[5]We used https://web.wurfl.io/ to obtain the make and model of a smartphone.

the subset of features that optimize our fingerprinting accuracy. For temporal features, no transformation of the data stream is required, but for spectral features we first convert the non-equally spaced data stream into a fixed-spaced data stream using cubic spline interpolation. We interpolate at a sampling rate of 8 kHz.[6] Then, we use *MIRtoolbox* and *Libxtract* to extract spectral features. Once feature extraction is complete, we look at feature selection where we explore different combinations of features to maximize our fingerprinting accuracy. We use the FEAST toolbox [127] and utilize the *Joint Mutual Information* criterion (JMI criterion is known to provide the best tradeoff in terms of accuracy, stability, and flexibility with small data samples [128]) for ranking the features.

Figure 4.4 shows the results of our feature exploration for the 30 lab smartphones. We see that when using only accelerometer data the F-score seems to flatten after considering the top 10 features. For gyroscope data we see that using all 75 features (25 per data stream) achieves the best result. And finally when we combine both accelerometer and gyroscope features, we see that the F-score plateaus after considering the top 70 features (from a total of 100 features). Among these top 70 features we found that 21 of them came from accelerometer features and the remaining 49 came from gyroscope features. In terms of the distribution between temporal and spectral features, we found that spectral features dominated with 44 of the top 70 features being spectral features. We use this subset of features in all our later evaluations.

4.7 Experimental Evaluations

In this setting we evaluate how well we can fingerprint smartphones using onboard motion sensors. We will first look at fingerprinting smartphones from our lab setting, followed by smartphones from our public setting and finally we will look at fingerprinting all smartphones in our collection.

[6] Although up-sampling the signal from ∼100 Hz to 8 kHz does not increase the accuracy of the signal, it does make direct application of standard signal processing tools more convenient.

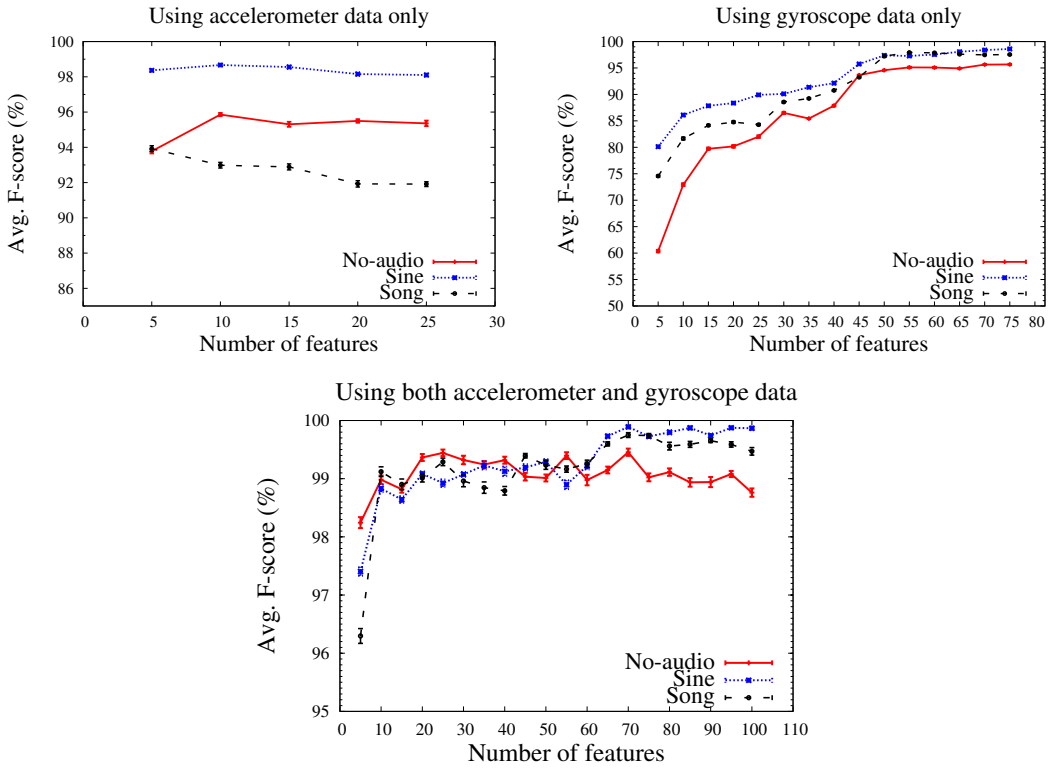


Figure 4.4: Exploring the number optimal features for different sensors. a) For accelerometer using more than the top 10 features leads to diminished returns, b) For gyroscope all 75 features contribute to obtaining improved accuracy, c) For the combined sensor stream using more than 70 features leads to diminished returns.

4.7.1 Results from Lab Setting

First, we look at fingerprinting smartphones under lab setting to demonstrate the basic viability of the attack. For this purpose we keep smartphones stationary on top of a flat surface. Table 4.5 summarizes our results. We see that we can almost correctly identify all 30 smartphones under all three scenarios by combining accelerometer and gyroscope features. Even when devices are kept in the hand of the user, we can successfully identify devices with an F-score of greater than 93%. While the benefit of the background audio stimulation is not clear from this table, we will later on show that audio stimulation do in fact enhance fingerprinting accuracy in the presence of countermeasure techniques like sensor calibration and data obfuscation (more on this in Chapter 7). Overall these results indicate that it is indeed possible to fingerprint smartphones through motion sensors.

Table 4.5: Average F-score under lab setting.

Device Placed	Stimulation	Avg. F-score (%)		
		Accelerometer	Gyroscope	Accelerometer+Gyroscope
On Desk	No-audio	96	95	99
	Sine	98	99	100
	Song	93	98	100
In Hand	No-audio	88	83	93
	Sine	88	94	98
	Song	84	89	95

4.7.2 Results from Public Setting

Next, we apply our fingerprinting approach on the public dataset. Table 4.6 shows our findings. Compared to the results from our lab setting, we see a slight decrease in F-score but even then we were able to obtain an F-score of 95%. Again, the benefit of the audio stimulation is not evident from these results, however, their benefits will become more visible in the Chapter 7 when we discuss countermeasure techniques.

Table 4.6: Average F-score under public setting where smartphones are kept *on top of a desk*.

Stimulation	Avg. F-score (%)		
	Accelerometer	Gyroscope	Accelerometer+Gyroscope
No-audio	86	87	95
Sine	85	87	92

4.7.3 Results from Combined Setting

Finally, we combine our lab data with the publicly collected data to give us a combined dataset containing 93 different smartphones. We apply the same set of evaluations on this combined dataset. Table 4.7 highlights our findings. Again, we see that combining features from both sensors provides the best result. In this case we obtained an F-score of 96%. All these results suggest that smartphones can be successfully fingerprinted through motion sensors.

Table 4.7: Average F-score under both lab and public setting where smartphones are kept *on top of a desk*.

Stimulation	Avg. F-score (%)		
	Accelerometer	Gyroscope	Accelerometer+Gyroscope
No-audio	85	89	96
Sine	89	89	95

4.8 Sensitivity Analysis

In this section we will look at how robust our fingerprints are as we vary the size of the device population and training set. We also study how our sensor fingerprints react to temperature change. Finally, we investigate temporal stability of motion sensors.

4.8.1 Impact of Device Number

We evaluate the accuracy of our classifier while varying the number of devices. We pick a subset of n devices from our dataset and perform the training and testing steps for this subset. For each value of n, we repeat the experiment 10 times, using a different random subset of n devices each time. In this experiment we only consider the use of both accelerometer and gyroscope features, since they produce the best performance (as evident from our previous results), and focus on the no-audio and sine wave background scenarios. Figure 4.5 shows that the F-score generally decreases with large number of devices, which is expected because an increased number of unique labels makes classification more difficult. But even then scaling from 10 devices to 93 devices the F-score decreases by only 4%. Extrapolating from the graph, we expect classification to remain accurate even for significantly larger datasets.

4.8.2 Impact of Training Set Size

We also consider how varying the training set size impacts the fingerprinting accuracy. For this experiment we vary the ratio of training and testing set size. For this experiment we only look at data from our lab setting as many of the devices from our public setting did not have exactly 10 samples.

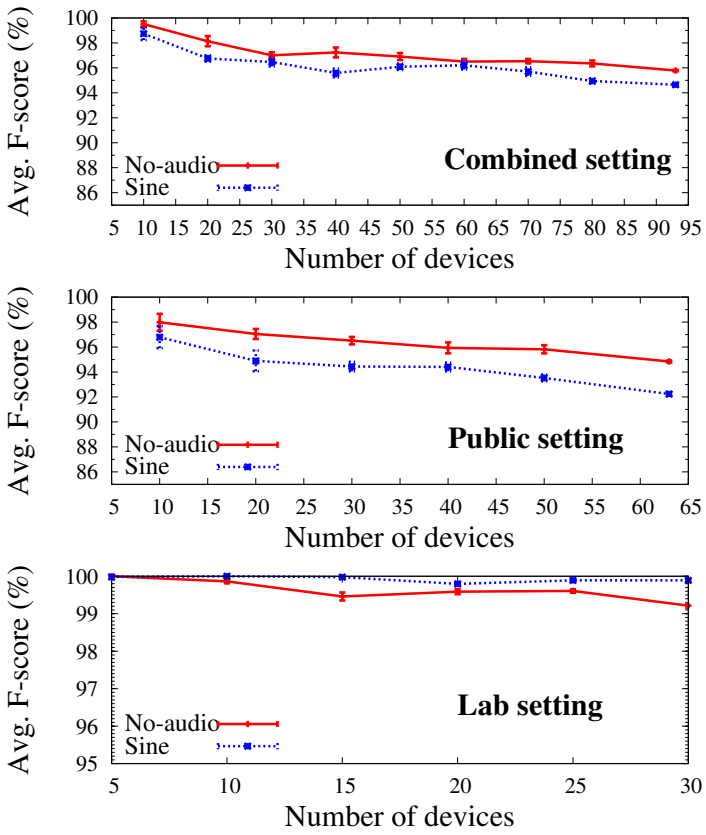


Figure 4.5: Average F-score for different numbers of smartphones. F-score generally tends to decrease slightly as more devices are considered.

We also consider the setting where there is no background audio stimulation and use the combined features of accelerometer and gyroscope. Figure 4.6 shows our findings. While an increased training-set size improves classification accuracy, even with mere two training samples (each consisting of 5–6 seconds worth of sensor data) we can achieve an F-score of 98%; with increased training set sizes producing an F-score of over 99%.

4.8.3 Impact of Temperature

Here we analyze how temperature impacts the fingerprint of smartphone sensors. For this purpose we collect sensor data under different temperatures. We took one set of readings outside our office building on September 03, 2015 (with temperatures in the range of $91°F$ to $93°F$) while we took another set of readings on October 9, 2015 (with temperatures in the range of $61°F$ to $63°F$). In both cases we also took readings inside the office where tempera-

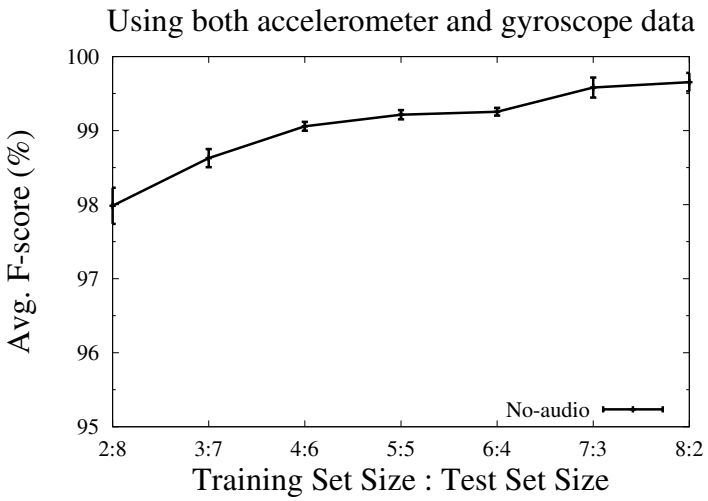


Figure 4.6: Average F-score for different ratio of training and testing data. With only two training data we achieved an F-score of 98%.

ture was set to around $74°F$ on the thermostat. As these set of experiments were conducted at a later period of time compared to our other experiments, we were only able to collect data from 17 smartphones (as described in Table 4.8).[7] Therefore, the results described in this section are in the context of the smartphones specified in Table 4.8.

Table 4.8: Types of phones used for analyzing temperature effect.

Maker	Model	Quantity
Apple	iPhone 5	4
	iPhone 5s	3
Samsung	Nexus S	3
	Galaxy S3	2
	Galaxy S4	5
Total		17

Table 4.9 summarizes our findings. We refer to September 03, 2015 as a *hot* day and October 09, 2015 as a *cold* day. From Table 4.9 we see that temperatures do lower F-score where warmer temperatures cause more discrepancies in the generated fingerprints compared to colder temperatures (as indicated by the red and blue blocks in the table).

[7]We only had access to these 17 smartphones at the time of conducting this experiment.

Table 4.9: Impact of temperature on motion sensor fingerprinting.

No-audio		Test (Avg. F-score in %)			
		Inside (hot)	Outside (hot)	Inside (cold)	Outside (cold)
Train	Inside (hot)	**100***	89	90	92
	Outside (hot)	90	**100***	81	75
	Inside (cold)	89	77	**100***	97
	Outside (cold)	86	82	99	**100***

Sine wave		Test (Avg. F-score in %)			
		Inside (hot)	Outside (hot)	Inside (cold)	Outside (cold)
Train	Inside (hot)	**100***	80	92	91
	Outside (hot)	83	**99***	82	72
	Inside (cold)	88	72	**100***	90
	Outside(cold)	85	69	92	**100***

* 50% of the dataset is used for training and remaining 50% for testing

4.8.4 Temporal Stability

We now take a closer look at how the fingerprints evolve over time. For this purpose we reuse data collected from Section 4.8.3. As we collected data inside our lab in two different dates (one on September 03, 2015 and the other on October 09, 2015) we can analyze how sensor fingerprints change over time and how they impact our F-score. Table 4.10 summarizes our findings. We see that over time fingerprints do change to some extent, but even then we can achieve an F-score in the range of 88–92%.

Table 4.10: Fingerprinting sensors at different dates.

No-audio		Test (Avg. F-score in %)	
		Sept. 03, 2015	Oct. 09,2015
Train	Sept. 03, 2015	**100***	90
	Oct. 09,2015	89	**100***

Sine wave		Test (Avg. F-score in %)	
		Sept. 03, 2015	Oct. 09,2015
Train	Sept. 03, 2015	**100***	92
	Oct. 09,2015	88	**100***

* 50% of the dataset is used for training and remaining 50% for testing

4.9 Summary

In this chapter, we show that motion sensors such as accelerometer and gyroscope can be used to uniquely identify smartphones. The more concerning matter is that these sensors can be surreptitiously accessed by a web page publisher without users' awareness. We do, however, experiment with only 93 devices; a larger target device pool could lower our accuracy. In the next chapter, we will look at how our fingerprinting approach scales with large number of smartphones.

CHAPTER 5

LARGE-SCALE MOTION SENSOR FINGERPRINTING

In this chapter we extend our work on motion sensor fingerprinting for a large pool of smartphones. We also develop a generic model to estimate prediction accuracies for larger-scale user populations.

5.1 Overview

An important question that we want to address is whether our fingerprinting approach can be effective at scale. To answer this question we first perform a larger-scale evaluation of our approach by collecting motion sensor data from a total of 610 devices. We rerun our classifications on this large dataset and show that high classification accuracy is still feasible. We then use the data we collected to develop a model to predict classification accuracies for even larger datasets, by fitting a parametric distribution to model inter- and intra-cluster distances. These distributions are then used to predict the accuracy of a k-NN classifier, used with state-of-the-art distance metric learning techniques. Our evaluation reveals that even with 100 000 devices an accuracy of 10–16% can be achieved depending on training set size, which suggests that motion sensor fingerprinting can be effective when combined with even a weak browser fingerprint. Note that because k-NN underperforms other classifiers, such as a random forest (or bagged trees), our estimate of accuracy is quite conservative.

5.2 Data Collection

After our initial recruitment of users through institutional mass email and social media like Facebook and Twitter (as described in Section 4.5.2), we extend our data collection through Amazon's Mechanical Turk [129]. Users

are asked to visit our web page while placing their smartphone on a flat surface. Thus, mimicking the scenario where the user has placed his/her smartphone on a desk while browsing a web page. We only collect data for the *no-audio* setting (as described in Table 4.2), so our web page collects 10 samples consecutively where each sample is 5–6 seconds worth of sensor data; total participation time is in the range of 1 minute. In total, we had a total of 610 participants over a period of three months.[1] We obtained data from 108 different brands (i.e., make and model) of smartphones with different models of iPhone comprising nearly half of the total devices as shown in Appendix B.

Since some participation was voluntary for users not using Mechanical Turk, we had many devices for which we had fewer than 10 samples. Figure 5.1 shows the distribution of the different number of samples per device. Also, we received data from participants at various sampling rates ranging from 20 Hz to 120 Hz. This maybe caused by various factors such as the current battery life or the number of applications running in the background.

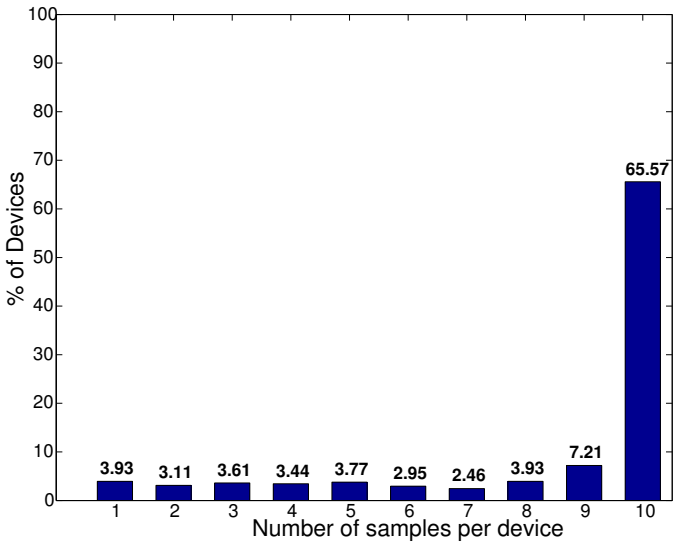


Figure 5.1: Distribution of the number of data samples per smartphone.

[1]This 610 devices included 30 lab phones, 63 public phones and 517 Mechanical Turk participant phones.

5.3 Classification Algorithms and Metrics

Classification Algorithms: In Chapter 4 we saw that ensemble based approaches such as random forest (i.e., bagged decision trees) out perform other multiclass classifiers. We, therefore, only explore the performance of the following two classifiers in this chapter: k-Nearest Neighbor (k-NN) and random forest (MATLAB's Treebagger model) [125].

Evaluation metrics: We use *F-score* (described in Section 3.5.2) to report real world large-scale evaluation results. To evaluate large-scale *simulation* results we use *Accuracy* as our evaluation metric.[2] Accuracy is defined as the portion of test traces that are correctly classified.

$$\text{Accuracy, } Acc = \frac{\text{\# of samples correctly classified}}{\text{Total test samples}} \tag{5.1}$$

5.4 Fingerprinting Large Number of Smartphones

We had a total of 610 participants in our data collection study. To evaluate the performance of the classifiers, we first split our dataset into training and testing set. As we have devices with different number of data samples (see Figure 5.1), we evaluate F-score for different number of training samples. To prevent any bias in the selection of training and testing set, we *randomly* select training and testing samples and rerun our experiments 10 times to report the average F-score.[3] Table 5.1 summarizes the average F-score for different number of training samples per device.

From Table 5.1 we see that we can achieve high classification accuracy even for this larger dataset. With five training samples, which correspond to about 25 seconds of data, accuracy is 86%, increasing to 90% with 9 training samples. Even with a single 5 seconds worth of data sample, we can obtain 33% accuracy, which may be sufficient if a small amount of extra information can be obtained through other browser fingerprinting techniques, however weak.

[2] *Accuracy* can be thought of as a relaxed version of *F-score*.

[3] We also compute the 95% confidence interval for F-score, but we found it to be less than 1% in most cases.

Table 5.1: Average F-score for different size of training set.

Training samples per device	Number of devices	Avg. F-score (%)
		Random Forest*
1	586	33
2	567	65
3	545	78
4	524	83
5	501	86
6	483	88
7	468	89
8	444	89
9	400	90

* 100 bagged decision trees

5.5 Large-Scale Simulation

Although we have shown that we can reliably fingerprint up to a few hundred devices, in real world scenarios the fingerprinted population will be much larger. It is not feasible for us to collect data on much larger datasets; instead, we develop a model to predict how well a classifier will perform as the number of devices grows. However, although random forest provides the best classification performance, on our dataset, its operation is hard to model, as different trees use a different random sample of features. We therefore base our analysis on nearest-neighbor classifier (k-NN), which uses a distance metric that we can model parametrically. Note that k-NN does not perform as well as random forest; as a result, our estimates are a *conservative* measure of the actual attainable classification accuracies.

5.5.1 Distance Metric Learning

The k-NN algorithm relies on a distance metric to identify neighboring points. It is possible to compute simple Euclidean distance between feature vectors; however, this is unlikely to yield optimal results as some features will tend to dominate. Learning a better distance (or similarity) metric between data points has received much attention in the field of machine learning, pattern recognition and data mining for the past decade [130]. Handcrafting a good distance metric for a specific problem is generally difficult and this has led

to the emergence of *metric learning*. The goal of a distance metric learning algorithm is to take advantage of prior information, in form of labels, to automatically learn a transformation for the input feature space. A particular class of distance function that exhibits good generalization performance for distance-based classifiers such as k-NN, is *Mahalanobis metric learning* [131]. The aim is to find a global, linear transformation of the feature space such that relevant dimensions are emphasized while irrelevant ones are discarded. The linear transformation performs arbitrary rotations and scalings to conform to the desired geometry. After projection, Euclidean distance between data points is measured.

State-of-the-art Mahalanobis metric learning algorithms include *Large Margin Nearest Neighbor* (LMNN) [132], *Information Theoretic Metric Learning* (ITML) [133] and *Logistic Discriminant Metric Learning* (LDML) [134]. A brief description of these metric learning algorithms is provided by Köstinger et al. [131]. To understand how these metric learning algorithms improve the performance k-NN classifier, we first plot the *mutual information* (MI) of each feature before and after each transformation. Figure 5.2 shows the amount of mutual information per feature under both untransformed and transformed settings. Figure 5.2 shows a clear benefit of the distance metric learning algorithms. All the transformations provide higher degree of mutual information compared to the original untransformed data. Among the three transformations we see that LDML on average provides slightly higher amount mutual information per feature. Distribution of the top 12 features for both original and LDML-transformed feature space is provided in Appendix A.

To show that LDML provides the best transformation on our dataset we rerun the k-NN classifier on the transformed feature space. Table 5.2 highlights the average F-score for different metric learning algorithms. We see that for our dataset, LDML seems to be the best choice. We, therefore, use LDML algorithm to transform our feature space before applying k-NN for the rest of this chapter.

However, even with LDML, k-NN generally underperforms random forest, as seen in Table 5.3: our F-score drops from 78% to 50% with 3 training samples and from 86% to 54% with 5 training samples. The only exception occurs for one training sample where k-NN with LDML performs slightly better than random forest.

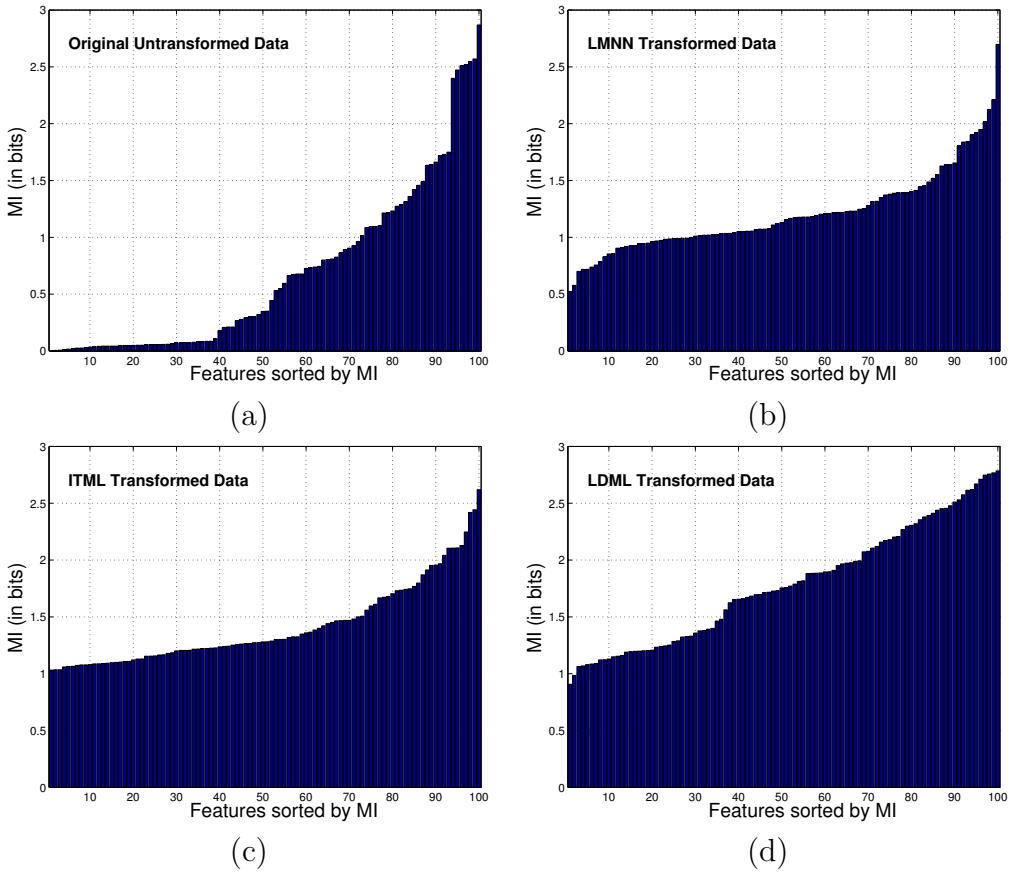


Figure 5.2: Comparing mutual information for different metric learning algorithms. Mutual information per feature for (a) untransformed data (b) LMNN transformation (c) ITML transformation, and (d) LDML transformation.

Table 5.2: Performance of different metric learning algorithms.

Avg. F-score for k-NN*			
Untransformed	LMNN	ITML	LDML
35	41	46	50

* $k = 1$ with 3 training samples per device

5.5.2 Intra- and Inter-Device Distance Modeling

To predict how k-NN will operate on larger datasets, we proceed to derive a distribution for distances between samples from different devices (inter-device), and a second distribution for distances between different samples from the same device (intra-device), after applying the LDML transformation to the feature space. Since each data sample is a point in an n-dimensional feature space, we compute the Euclidean distance between any two data

Table 5.3: Average F-score of k-NN after LDML.

Training samples per device	Number of devices	Avg. F-score (%)		
		k-NN*	k-NN+LDML*	Random Forest+
1	586	24	38	33
2	567	31	43	65
3	545	35	50	78
4	524	36	52	83
5	501	38	54	86
6	483	38	54	88
7	468	38	53	89
8	444	37	52	89
9	400	35	50	90

* $k = 1$
+ 100 bagged decision trees

samples using the following equation:

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \tag{5.2}$$

where p and q represent two feature vectors in an n-dimensional space defined as follows, $p = (p_1, p_2, ..., p_n)$ and $q = (q_1, q_2, ..., q_n)$. We then group distances between feature vectors from the same device into one class C_{intra} and distances between feature vectors from different devices into another class C_{inter}. Class C_{intra} and C_{inter} can be defined as follows:

$$C_{intra} = \{x : x = d(p,q), p \in D_i, q \in D_i, \forall i \in D\}$$
$$C_{inter} = \{x : x = d(p,q), p \in D_i, q \in D_j, i \neq j, \forall i, j \in D\}$$

where D refers to the set of all devices; we consider only devices with at least two training samples, we have 567 such devices. We can now fit an individual distribution for each class. To do this we utilize MATLAB's *fitdist* function [135]. To avoid overfitting, we split our devices into four equal subsets. We then fit and compare distributions from each subset. Figure 5.3 shows the top five estimated inter-device distance (C_{inter}) distributions for each subset of devices. Here, the distributions are ranked based on *Akaike Information Criterion* (AIC) [136]. From Figure 5.3 we can see that the top

five distributions are more or less consistent across all four subsets.

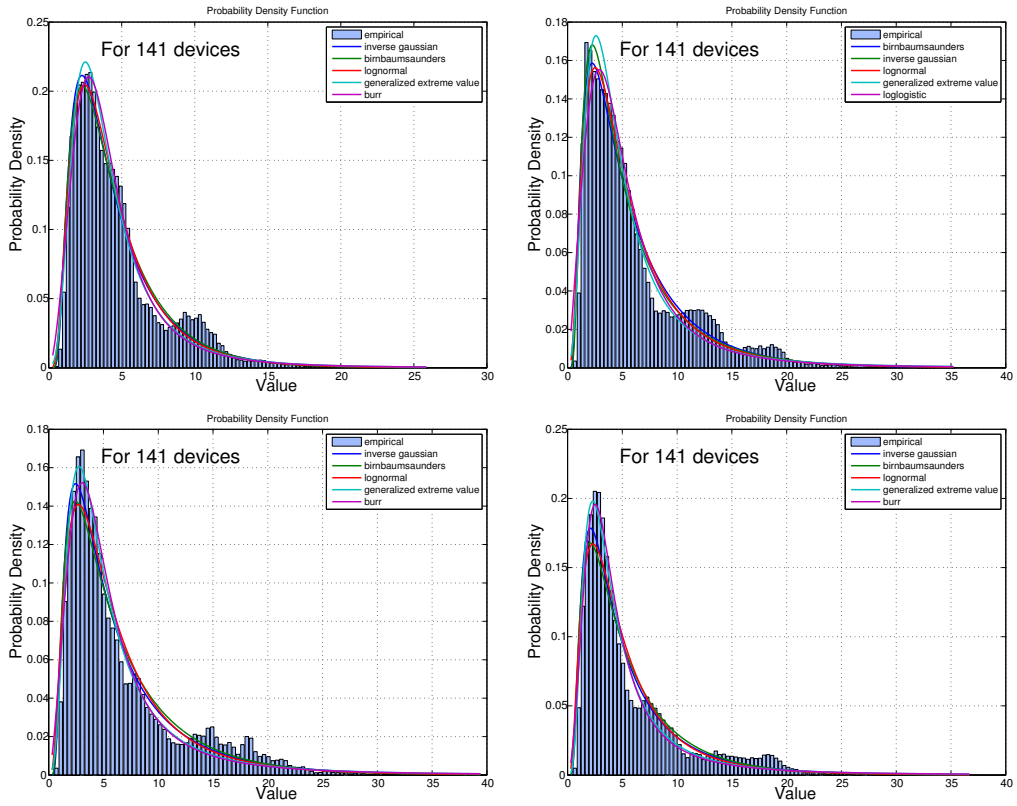


Figure 5.3: Estimated inter-device distance distributions for 4 subsets of devices where each subset contains 141 devices.

We see similar outcomes when we plot the intra-device distance (C_{intra}) distributions. Figure 5.4 shows the top five estimated distributions for each subset of devices. Again we can see that certain parametric distributions are present across all four subsets.

Next, we plot the same inter-device distance distribution but this time we consider data from all 567 devices. Figure 5.5(a) highlights the top five distributions. Comparing Figure 5.3 and Figure 5.5(a), we see that the most representative inter-device distance distribution is an *Inverse Gaussian* distribution. Similarly, we find that the most likely intra-device distance distribution (C_{intra}) is a *Generalized extreme value* distribution as shown in Figure 5.5(b). Figure 5.5(c) shows the difference between intra- and inter-device distance distribution.

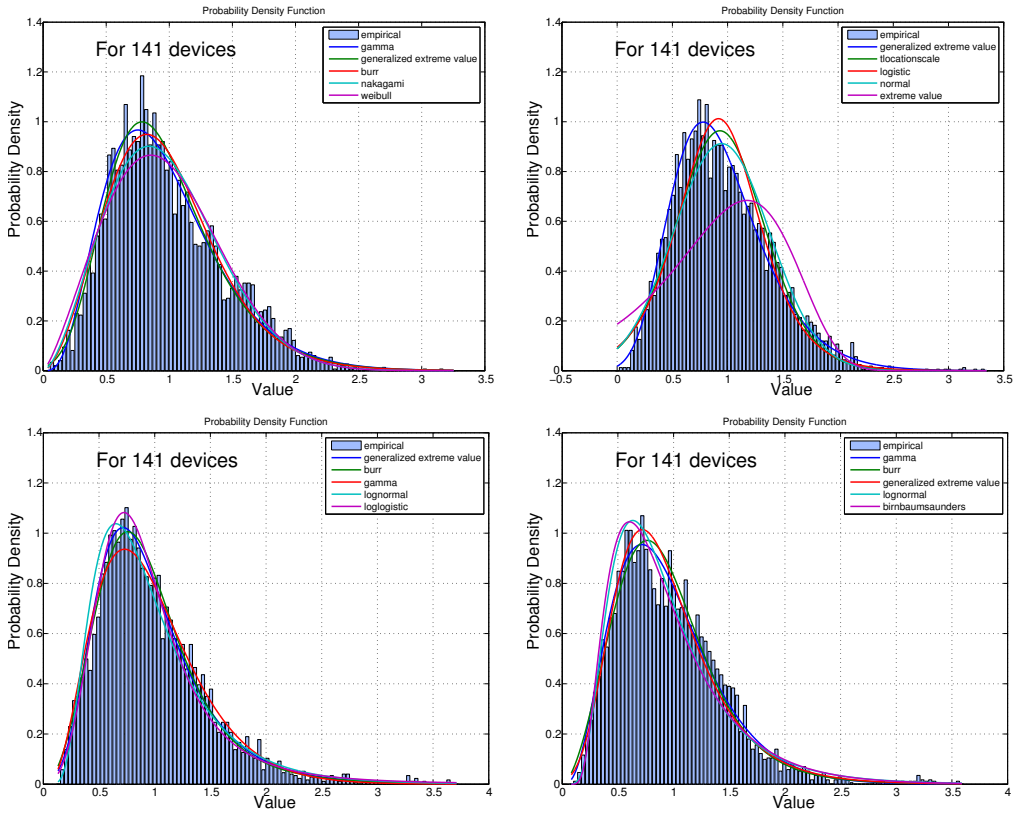


Figure 5.4: Estimated intra-device distance distributions for 4 subsets of devices where each subset contains 141 devices.

5.5.3 Simulating k-NN for Large Number of Devices

Now that we have representative distributions for intra- and inter-device distance, we can simulate a k-NN classifier. The pseudo code for simulating k-NN classifier is provided in Algorithm 2. The algorithm works as follows. Let us assume that there are D devices and for each device we have N training samples. Now, for any given test sample, a k-NN classifier, first computes $N \times D$ distances of which N distances are with samples from the same device and $N \times (D-1)$ distances are with all samples belonging to $(D-1)$ other devices. We emulate these distances by drawing N and $N \times (D-1)$ distances from our representative intra- and inter-device distance distributions, respectively. k-NN classifier then inspects the class label for the k nearest neighbors. We can emulate this step by sorting the distances and picking the k lowest distances. Lastly, k-NN classifier outputs the class label with the majority vote. To emulate this step we assign each distance a label of either 0 (meaning distance from same device) or 1 (meaning distance from different

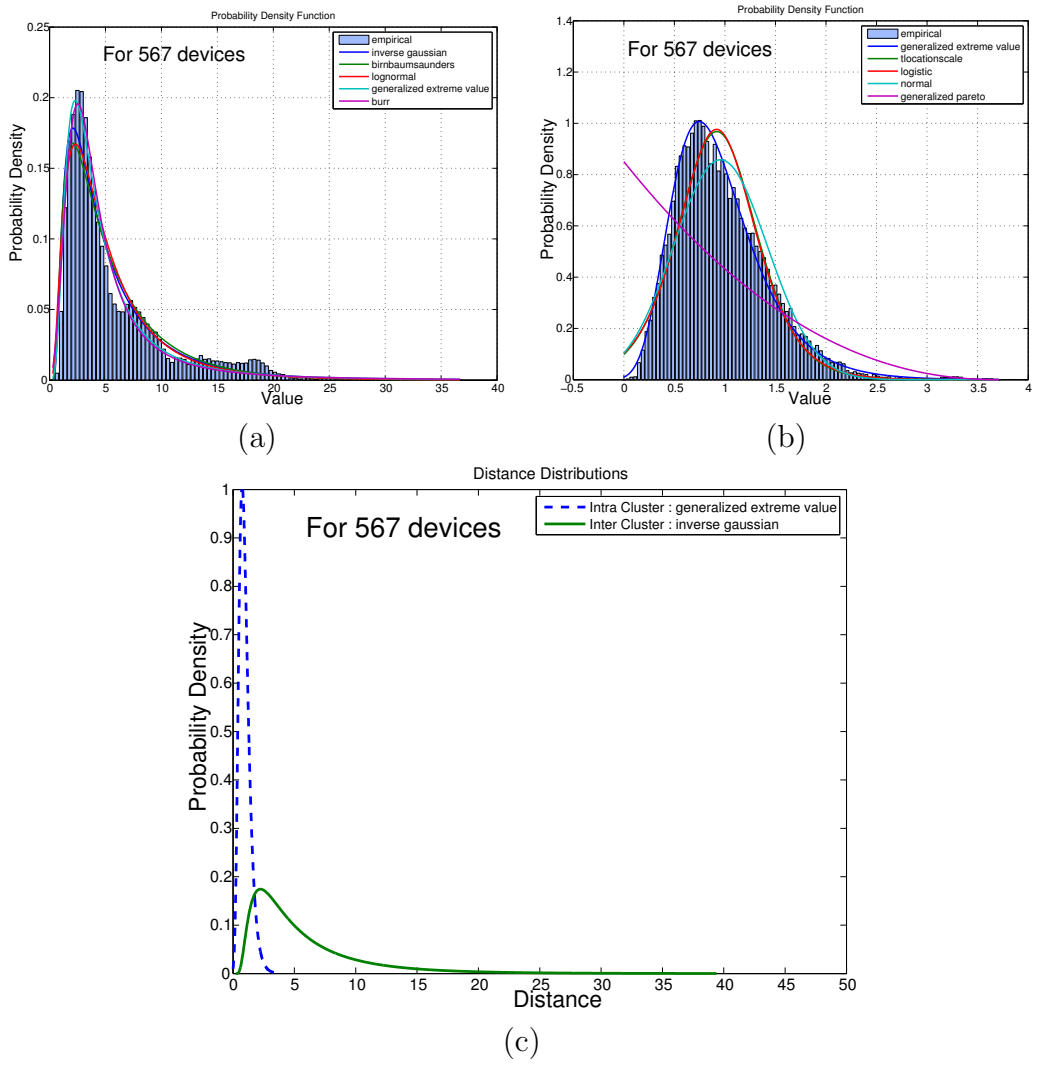


Figure 5.5: Estimated distributions for (a) inter-device distance (C_{inter}) (b) intra-device distance (C_{intra}). (c) Difference between intra- and inter-device distance distribution.

device). We then check if label-0 dominates over label-1, if so we count that as a successful classification. This whole process repeats multiple times to provide us with an average classification accuracy.

Next, we run our k-NN simulator for a large number of devices. Given that we have our intra- and inter-device distance distributions ($Distr_{intra}$ and $Distr_{inter}$) we can simulate the fingerprinting accuracy for any number of devices (D). The only two parameters that we need to explore are – the number of training samples per device (N) and number of nearest neighbors (k). Given that a user spends on average anywhere between 10 to 20 seconds

Algorithm 2 Simulating k-NN classifier.

Input: k, N, D, $Distr_{intra}$, $Distr_{inter}$, $Runs$
k – number of nearest neighbors (odd integer)
N – number of training samples per device
D – number of devices
$Distr_{intra}$ – intra-device distance distribution
$Distr_{inter}$ – inter-device distance distribution
$Runs$ – number of runs
Output: Acc
Acc – Average classification accuracy

$d \leftarrow \{\}$ #list of (distance,label) tuple
$Acc \leftarrow 0$
for $i := 1$ to $Runs$ **do**
  #add N intra-distances and label each with 0
  for $j := 1$ to N **do**
    $d \leftarrow d + \{(random(Distr_{intra}), 0)\}$
  end for
  #add $N \times (D-1)$ inter-distances and label each with 1
  for $j := 1$ to $N \times (D-1)$ **do**
    $d \leftarrow d + \{(random(Distr_{inter}), 1)\}$
  end for
  $d \leftarrow sort(d)$ #in ascending order of distance
  $l \leftarrow label(d, k)$ #return label for top k elements
  $imposters \leftarrow sum(l)$ #sum top k labels
  if $imposters < k/2$ **then**
    $Acc \leftarrow Acc + 1$ #correct decision
  end if
end for
$Acc \leftarrow Acc/Runs$
return Acc

on a web page [137, 138] values of $N \leq 5$ seem most realistic (each of our data sample is around 5 seconds worth of web session). We, therefore, vary N from 2 to 5 in our experiments. Also, we explore all odd integer values of $k \leq N$ (in practice a rule of thumb in machine learning is to limit k to the square root of the size of the training set [139], i.e., $1 \leq k \leq \sqrt{N}$). Figure 5.6 shows how our simulation results compare with our real world results for different values of N and k.

From Figure 5.6 we can see that as k increases the simulation results also start to deviate from real world results. However, setting $k = 1$ provides the

best overlap between real world and simulation results.[4] Also, we can see that for $k = 1$ the average classification accuracy is in the range of 10–16% when we scale up to 100 000 devices. This accuracy is unlikely to be sufficient if motion sensors are the unique source of a fingerprint, but it suggests that combining motion sensor data with even a weak browser-based fingerprint is likely to be effective at distinguishing users in large populations. Additionally, these classification accuracies are conservative and potentially provide a lower bound on performance, as random forests provide significantly better performance.

5.6 Summary

We demonstrated that sensor fingerprinting is feasible on a much larger scale than what we previously studied. We show that 90% accuracy can be achieved for up to 400 devices, and at least 10–16% accuracy can be realized with 100 000 devices, as predicted according to our model. While this accuracy alone might not be sufficient to uniquely identify a smartphone but combining our approach with any other browser-based fingerprints is likely to provide sufficient discrimination among a large pool of smartphones.

[4]Differences between our k-NN model and the actual k-NN classifier on real data arise from an imperfect fit of the distribution as well as the fact that our model makes an assumption that intra- and inter-phone distances are identically and independently distributed.

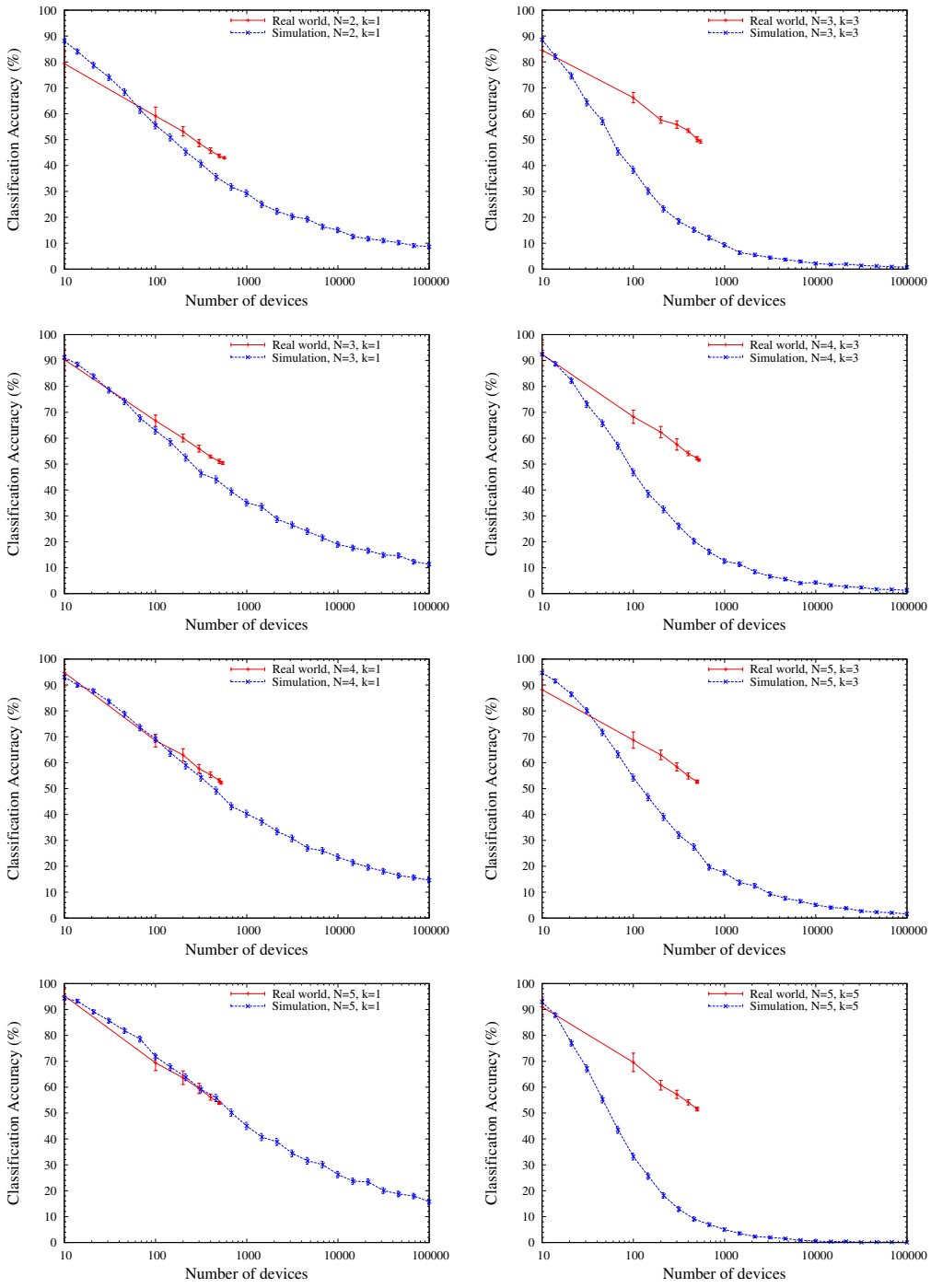


Figure 5.6: Comparing real world results with simulation results. Simulation results closely match real world results for $k = 1$.

CHAPTER 6

USAGE PATTERNS OF MOTION SENSORS IN THE WILD

In this chapter we look at how many of the top websites access motion sensors. We also cluster the usage patterns into broad groups to determine how motion sensors are being utilized in the web today.

6.1 Data Collection Framework

Figure 6.1 provides an overview of our methodology to automatically capture and cluster JavaScripts accessing motion sensor data through mobile browsers. To automate this process, we use Selenium Web Driver [140] to run an instance of Chrome browser with a *user agent* set for a smartphone client. In order to collect unfolded JavaScripts, we attach a debugger between the V8 JavaScript engine [141] and the web page. Specifically, we observe `script.parsed` function, which is invoked when new code is added with `<iframe>` or `<script>` tag. We implement the debugger as a Chrome extension and monitor all JavaScript snippets parsed on a web page. The debugger collects script snippets that access sensor data, i.e., scripts that invoke sensor APIs.

6.2 Feature Extraction

Once scripts are collected, we aim to cluster them into a broad groups to identify their usage pattern. To analyze and quantify the similarity between JavaScript snippets, we parse them to produce *Abstract Syntax Trees* (ASTs). ASTs have been used in prior literatures for JavaScript malware detection [142]. ASTs allow us to retain the structural and logical properties of the code while ignoring fine details like variable names, which are not

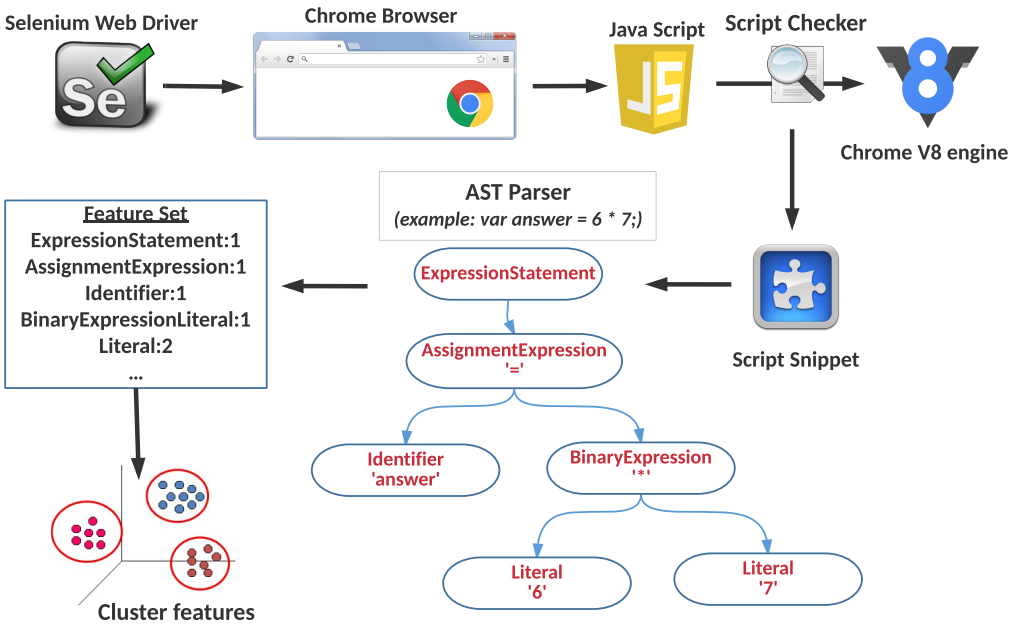


Figure 6.1: Overview of our JavaScript analysis setup.

useful for our analysis. We use the Esprima JavaScript parser [143] to visualize AST for each JavaScript snippet. We transform ASTs into normalized node sequences by performing *pre-order traversal* on each tree. It should be mentioned that we start parsing each AST from the point where sensor data is first accessed. Each variable length sequence is composed of node types that appear in the tree. Since there are 88 distinct node types in JavaScript language, we transform the variable length normalized node sequences into 88-dimensional summary vectors. In other words, each JavaScript snippet is represented as a point in a 88-dimensional space, where each dimension corresponds to a node type. Finally, we attempt to perform unsupervised clustering on these summary vectors (as shown in Figure 6.1).

6.3 Measurement Study Results

We provide real world results in terms of how many websites access motion sensor data and also in terms of what are the main use cases for accessing motion sensors.

6.3.1 Websites Accessing Motion Sensors

We run our experiment for the top 100 000 Alexa websites [21]. Among these websites we find that 1130 websites contain some form of JavaScript code that accesses at least one of the motion sensors. It is worth mentioning that a few of the scripts were downloaded from *ad networks* as the web pages were loaded. Table 6.1 shows a breakdown of the detected websites into their corresponding ranking groups. We see that majority (1022 out of the 1130) of our detected websites come from the top 10 000–100 000 websites. However, even 6 of the top 100 websites seem to access motion sensors.

Table 6.1: Top websites accessing motion sensors.

Rank	# of sites
1–100	6
101–1000	12
1001–10000	90
10001–100000	1022

6.3.2 Types of Usage for Motion Sensors

Our next goal is to cluster these 1130 websites into individual groups based on their usage of sensor data, so that we can identify the major reasons as to why websites access motion sensors. To cluster the JavaScript snippets into a small number of groups we first perform feature reduction to remove irrelevant features. Many of the 88 features had a value of zero for all Javascript snippets, so we first throw out these features. This reduces the size of the feature vector to 31. We then use the MATLAB Toolbox provided by Laurens van der Maaten [144] to further map the features into a low dimensional space. We find that *Stochastic Proximity Embedding* (SPE) method [145] provides the best outcome in terms of both reducing dimensionality and providing good clusters. Our final reduced feature space had three dimensions. Figure 6.2 shows a scatter plot along the three dimensions for all the JavaScripts. We can clearly see that the JavaScripts form clusters.

To determine the number of clusters that is a good fit for our data we run k-means clustering algorithm [146] for different number of clusters and perform *Silhouette* analysis [147]. Silhouette analysis can be used to study the separation distance between the resulting clusters. Silhouette coefficient

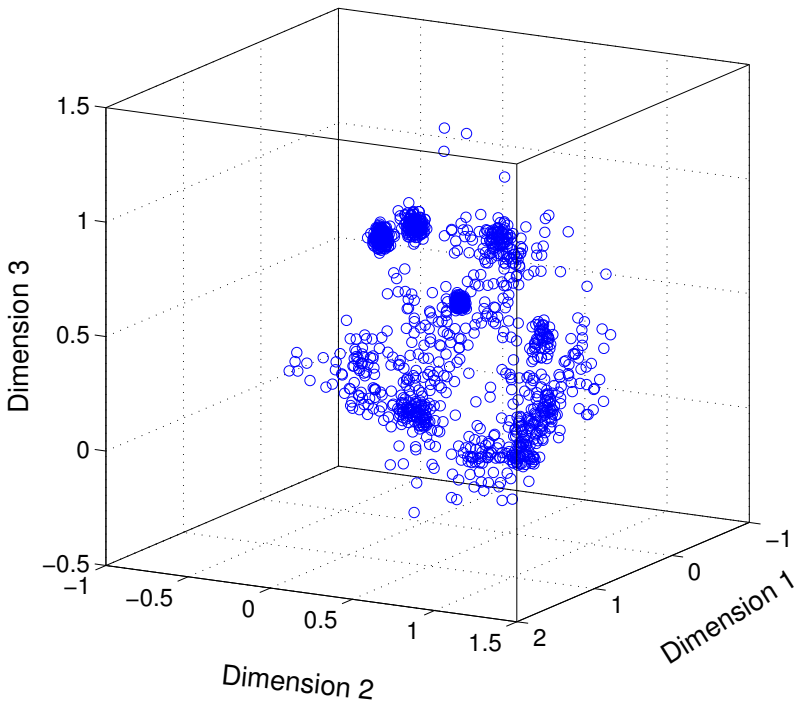


Figure 6.2: Scatter plot for JavaScript snippets accessing motion sensors along reduced dimensions.

Table 6.2: Silhouette coefficient for different number of clusters.

Clusters	3	4	5	6	7	8	9	10
$C_{silhouette}$	0.51	0.59	0.59	0.62	0.63	0.65	0.64	0.38

ranges from +1, indicating point are very distant from neighboring clusters, through 0, indicating points are very close to the decision boundary between two neighboring clusters, to -1, indicating points are probably assigned to the wrong cluster. Table 6.2 summarizes the average silhouette coefficients ($C_{silhouette}$) for different number of clusters. We see that silhouette coefficient peaks for 8 clusters. The corresponding silhouette plot for 8 clusters in given in Figure 6.3. We see that on average samples in cluster 1,2,4,6 and 7 have silhouette coefficient value greater than 0.6 while the samples in cluster 3,5 and 8 have silhouette coefficient close to 0.5. We also see some samples with negative silhouette coefficients and this is likely caused by JavaScripts reusing code snippets belonging to different libraries. Here, our goal is not to generate a perfect clustering of all the JavaScripts rather to broadly cluster them to identify the major usage patterns for accessing motion sensors.

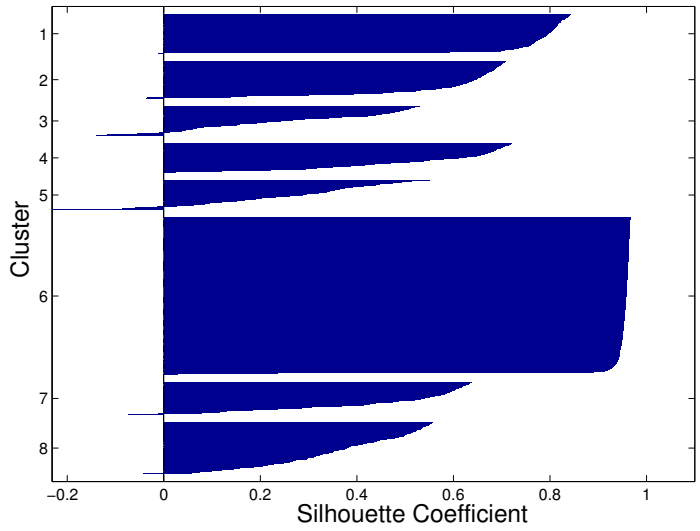


Figure 6.3: Silhouette plot for the estimated 8 clusters.

Once we have the general clusters, we then go back to the JavaScripts to understand their usage of motion sensor data. This part of the analysis was carried out manually. However, since we generated 8 clusters we randomly sampled multiple JavaScripts from each cluster to verify if they were performing similar functionality with the sensor data.[1] We were able to identify 8 generic use cases for the motion sensors. Table 6.3 and 6.4 summarizes our findings. We see that majority of the detected scripts periodically send sensor data to some third party sites. We were not able to pinpoint the exact use case for sending motion sensor data to third party sites as we did not have access to third party code. The next big usage for motion sensor data is that they are used in generating random numbers. Other use cases include — parallax viewing, gesture detection, motion captcha, specific ad generation and orientation detection. We were not able to concretely identify the use case for cluster 3 as we found that it contains multiple scripts all performing different tasks; some were doing touch analytics using accelerometer to detect tilt while others were doing something similar to parallax scrolling. We intend to perform a more thorough in-depth analysis of this usage patterns in the future.

[1]We randomly sampled around 15 JavaScripts per cluster.

Table 6.3: Generic use cases for accessing motion sensor data.

Cluster #	% of scripts	Use Case
6	40.5	Transmit sensor data
4	16.6	Random number generator
8	9.7	Detect device orientation
5	8.9	Parallax scrolling/viewing
2	7.1	Gesture detections
1	7.0	Motion captcha
3	6.0	Miscellaneous
7	4.2	Specific Ad generation

Table 6.4: Description of use cases for accessing motion sensor data.

Clus. #	Use Case Description
6	Periodically sends motion sensor data to third party sites (can be marked suspicious)
4	Crypto libraries use sensor data to add entropy to random numbers [148]
8	Detects device orientation periodically to readjust components in the website
5	Parallax Engine that reacts to the orientation of a smart device [149]
2	A *jQuery* plug-in for gesture events such as 'pinch', 'rotate', 'swipe', 'tap' and 'shake' [150]
1	A *jQuery* CAPTCHA plug-in based on the HTML5 Canvas element [151]
3	We were not able to point the exact use case for this cluster.
7	Checks to see if accelerometer is present so that certain ad URLs can be requested

6.4 Summary

Our measurement study reveals that motion sensors are already being used by over 1% of the top 100 000 websites, and distressingly, sensor data is often sent to servers for storage and analysis, which could serve as a vehicle for fingerprinting. Thus, we can conclude that motion sensor fingerprinting is a realistic threat to mobile users' privacy.

CHAPTER 7

COUNTERMEASURES FOR MOTION SENSOR FINGERPRINTING

Up until now we have focused on showing how easy it is to fingerprint smartphones through onboard sensors. We now shift our focus on providing a systematic approach to defend against such fingerprinting techniques. Accessing the microphone requires explicit user permission; moreover, users are aware of the obvious privacy threats associated with providing access to microphone. We, therefore, feel promoting general awareness about not giving applications and/or websites access to microphone, unless it is deemed really useful, seems to be the most practical line of defense against fingerprinting smartphones through acoustic side-channels.

In this chapter, we only focus on developing countermeasures for fingerprinting motion sensors like accelerometers and gyroscopes as accessing them does not require any explicit user permission. In the absence of explicit user permission websites can surreptitiously access the motion sensors without interfering with the user's browsing experience. As a result users might not be aware that websites maybe fingerprinting their smartphones in the background while they are browsing certain web pages. Such surreptitious nature poses a greater threat to user privacy. We start off by investigating the following three countermeasure techniques: *sensor calibration* and *data obfuscation* and *sensor quantization*. We first study the effectiveness of the countermeasures in lowering fingerprinting accuracy and then study how such mitigation techniques impact the utility of the motion sensors.

7.1 Sensor Calibration

Bojinov et al. [73] observe that accelerometers have calibration errors, and use these calibration differences as a mechanism to distinguish between them. In particular, they consider an affine error model: $a^M = g \cdot a + o$, where

a is the true acceleration along an axis and a^M is the measured value of the sensor. The two error parameters are the offset o (bias away from 0) and the gain g which magnifies or diminishes the acceleration value. Our classification uses many features, but we find that the *mean* signal value is the most discriminating feature for each of the sensor streams, which is closely related to the offset. We therefore explore whether calibrating the sensors will make them more difficult to fingerprint. Note that calibration has a side effect of improving the accuracy of sensor readings and is therefore of independent value. We perform calibration only on the sensors in our lab phones (30 smartphones as described in Table 4.4) because we felt that the calibration process is too time consuming for the volunteers.[1] Moreover, we could better control the quality of the calibration process when carried out in the lab.

First, let us briefly describe the sensor coordinate system where the sensor framework uses a standard 3-axis coordinate system to express data values. For most sensors, the coordinate system is defined relative to the device's screen when the device is held in its default orientation (shown in Figure 7.1). When the device is held in its default orientation, the positive x-axis is horizontal and points to the right, the positive y-axis is vertical and points up, and the positive z-axis points toward the outside of the screen face.[2] We compute offset and gain error in all three axes.

7.1.1 Calibrating Accelerometers

Considering both offset and gain error, the measured output of the accelerometer ($a^M = [a_x^M, a_y^M, a_z^M]$) can be expressed as:

$$\begin{bmatrix} a_x^M \\ a_y^M \\ a_z^M \end{bmatrix} = \begin{bmatrix} O_x \\ O_y \\ O_z \end{bmatrix} + \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{bmatrix} \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \tag{7.1}$$

where $S = [S_x, S_y, S_z]$ and $O = [O_x, O_y, O_z]$ respectively represents the gain and offset errors along all three axes ($a = [a_x, a_y, a_z]$ refers to the actual

[1] Requiring around 12 minutes in total for calibrating both the accelerometer and gyroscope.

[2] Android and iOS consider the positive and negative direction along an axis differently.

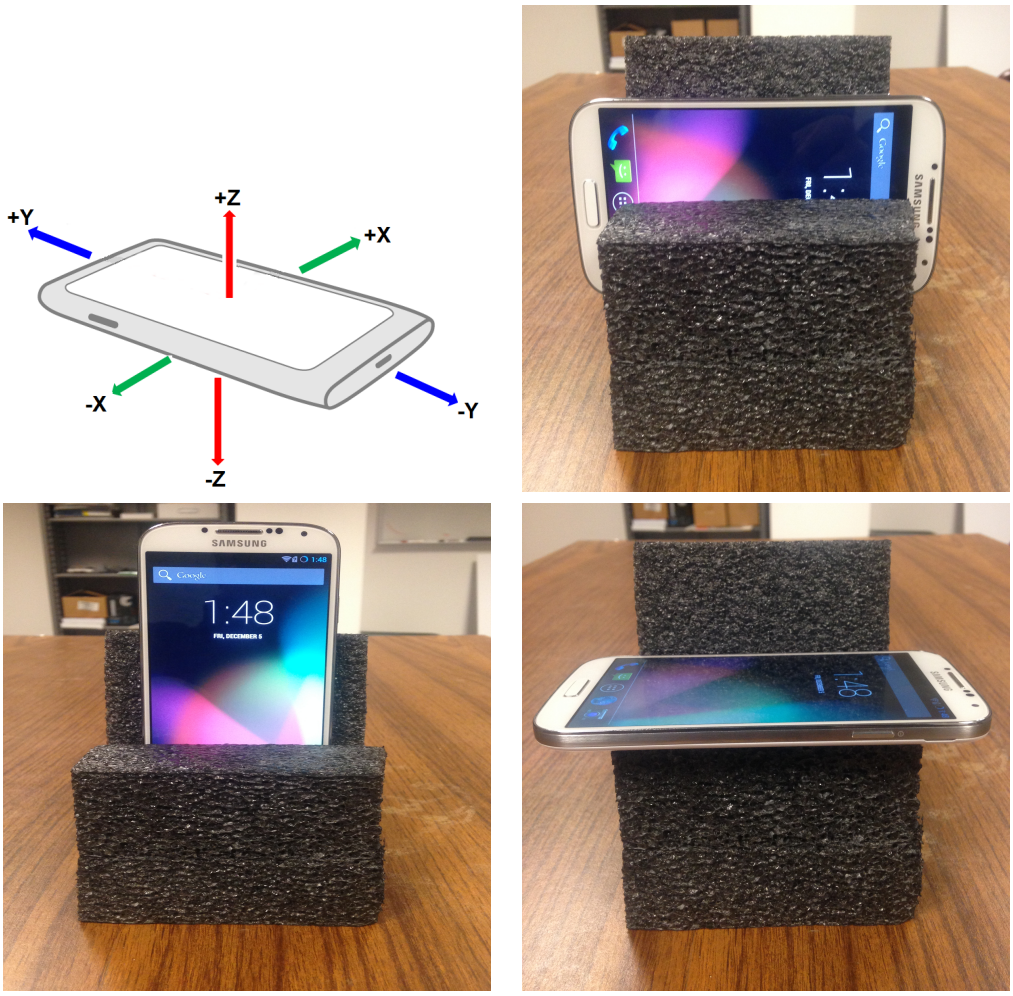


Figure 7.1: Calibrating accelerometer along three axes. We collect measurements along all 6 directions $(\pm x, \pm y, \pm z)$.

acceleration). In the ideal world $[S_x, S_y, S_z] = [1, 1, 1]$ and $[O_x, O_y, O_z] = [0, 0, 0]$, but in reality they differ from the desired values. To compute the offset and gain error of an axis, we need data along both the positive and negative direction of that axis (one measures positive $+g$ while the other measures negative $-g$). In other words, six different *static* positions are used where in each position one of the axes is aligned either along or opposite to earth's gravity. This causes $a = [a_x, a_y, a_z]$ vector to take one of the following six possible values $\{[\pm g, 0, 0], [0, \pm g, 0], [0, 0, \pm g]\}$. For example, if a^M_{z+} and a^M_{z-} are two values of accelerometer reading along the positive and negative z-axis, then we can compute the offset (O_z) and gain (S_z) error using the

following equations:

$$S_z = \frac{a^M_{z+} - a^M_{z-}}{2g}, \qquad O_z = \frac{a^M_{z+} + a^M_{z-}}{2} \tag{7.2}$$

We take 10 measurements along all six directions $(\pm x, \pm y, \pm z)$ from all our lab devices as shown in Figure 7.1. From these measurements we compute the average offset and gain error along all three axes using equation (7.2). Figure 7.2 shows a scatter-plot of the errors along $\texttt{z}-\texttt{axis}$ for 30 smartphones (each color code represents a certain make and model). We can see that the devices are scattered around all over the plot which signifies that different devices have different amount of offset and gain error. Such unique distinction makes fingerprinting feasible.

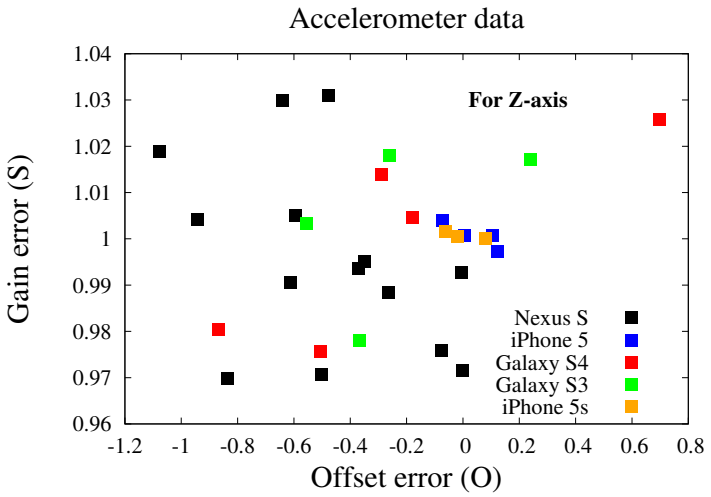


Figure 7.2: Accelerometer offset and gain error from 30 smartphones.

7.1.2 Calibrating Gyroscopes

Calibrating gyroscope is a harder problem as we need to induce a fixed angular change to determine the gain error even though the offset error can be computed while keeping the device stationary.[3] Similar to accelerometer we can also represent the measured output of the gyroscope (ω^M =

[3]However, we found that a gyroscope's offset was impacted by orientation.

$[\omega_x^M, \omega_y^M, \omega_z^M]$) using the following equation:

$$\begin{bmatrix} \omega_x^M \\ \omega_y^M \\ \omega_z^M \end{bmatrix} = \begin{bmatrix} O_x \\ O_y \\ O_z \end{bmatrix} + \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{7.3}$$

where again $S = [S_x, S_y, S_z]$ and $O = [O_x, O_y, O_z]$ respectively represents the gain and offset errors along all three axes. Here, $\omega = [\omega_x, \omega_y, \omega_z]$ represents the ideal/actual angular velocity. Ideally all gain and offset errors should be equal to 1 and 0 respectively. But in the real world when the device is rotated by a fixed amount of angle, the measured angle tends to deviate from the actual angular displacement (shown in Figure 7.3(a)). This impacts any system that uses gyroscope for angular-displacement measurements.

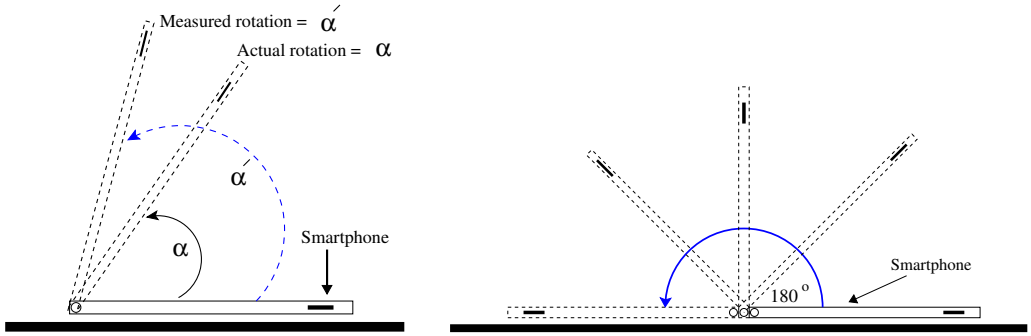


Figure 7.3: a) Offset and gain error in gyroscope impact systems that use them for angular-displacement measurements, b) Calibrating the gyroscope by rotating the device 180° in the positive x-axis direction.

To calibrate gyroscope we again need to collect data along all six different directions $(\pm x, \pm y, \pm z)$ individually, but this time instead of keeping the device stationary we need to *rotate* the device by a fixed amount of angle (θ). In our setting, we set $\theta = 180°$ (or π `rad`). For example, Figure 7.3(b) shows how we rotate the smartphone by 180° around the positive x-axis. The angular displacement along any direction can be computed from gyroscope

data in the following manner:

$$\begin{aligned} \omega_i^M &= O_i + S_i\omega, \quad i \in \{\pm x, \pm y, \pm z\} \\ \int_0^t \omega_i^M \, \mathrm{d}t &= \int_0^t O_i \, \mathrm{d}t + S_i \int_0^t \omega \, \mathrm{d}t \\ \theta_i^M &= O_i t + S_i \theta \end{aligned} \tag{7.4}$$

where t refers to the time it takes to rotate the device by θ angle with a fixed angular velocity of ω. Now, for any two measurements along the opposite directions of an axis we can compute the offset and gain error using the following equations:

$$O_i = \frac{\theta_{i+}^M + \theta_{i-}^M}{t_1 + t_2}, \; S_i = \frac{\theta_{i+}^M - \theta_{i-}^M - O_i(t_1 - t_2)}{2\pi} \tag{7.5}$$

where $i \in \{x, y, z\}$ and t_1 and t_2 represents the timespan of the positive and negative measurement respectively. We take 10 measurements along all six directions $(\pm x, \pm y, \pm z)$ and compute the average offset and gain error along all three axes. However, since it is practically impossible to manually rotate the device at a fixed angular velocity, the integration in equation (7.4) will introduce noise and therefore, the calculated errors will at best be approximations of the real errors. We also approximate the integral using *trapezoidal rule* [152] which will introduce more error.

We next visualize the offset and gain errors obtained from the gyroscopes of 30 smartphones (only showing for $\texttt{z} - \texttt{axis}$ where each color code represents a certain make and model). Figure 7.4 shows our findings. We see similar result compared to accelerometers where devices are scattered around at different regions of the plot. This suggests that gyroscopes exhibit different range of offset and gain error across different units.

7.1.3 Fingerprinting Calibrated Data

In this section we look at how calibrating sensors impact fingerprinting accuracy. For this setting, we first correct the raw values by removing the offset and gain errors before extracting features from them. That is, the calibrated value $a^C = (a^M - o)/g$. We then generate fingerprints on the *corrected data* and train the classifiers on the new fingerprints. Table 7.1 shows the average

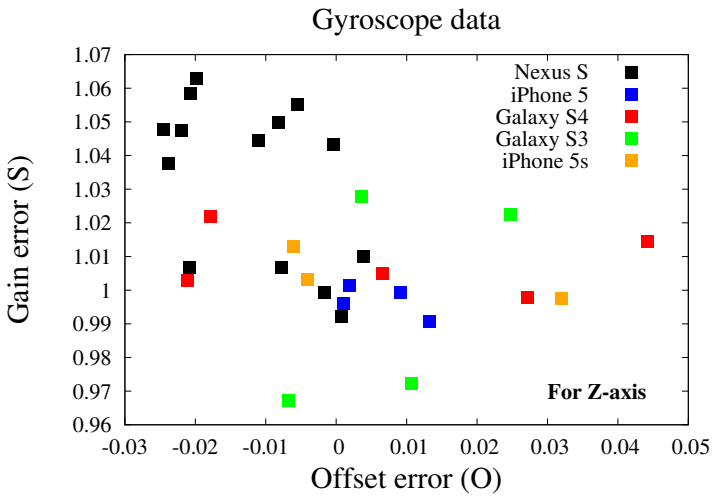


Figure 7.4: Gyroscope offset and gain error from 30 smartphones.

F-score for calibrated data under three scenarios (described in Table 4.2), considering both cases where the devices were kept on top of a desk and in the hand of a user. When we compare the results from uncalibrated data (Table 4.5) to those from calibrated data, we see that the F-score reduces by approximately 16–25% for accelerometer data but not as much for the gyroscope data. This suggests that we were able to calibrate the accelerometer much more precisely than the gyroscope, as expected given the more complex and error-prone manual calibration procedure for the gyroscope. Another interesting observation is that audio stimulation provides small improvement in classifier accuracy. This suggests that audio stimulation does not influence the dominant features removed by the calibration, but does significantly impact secondary features that come into play once calibration is carried out. Overall, our results demonstrate that calibration is a promising technique, especially if more precise measurements can be made. Manufacturers should be encouraged to perform better calibration to both improve the accuracy of their sensors and to help protect users' privacy.

7.2 Data Obfuscation

Rather than removing calibration errors, we can instead add extra noise to hide the miscalibration. This approach has the advantage of not requiring a calibration step, which requires user intervention and is particularly difficult

Table 7.1: Average F-score for calibrated data under lab setting.

Device Placed	Stimulation	Avg. F-score (%)		
		Accelerometer	Gyroscope	Accelerometer+Gyroscope
On Desk	No-audio	71	97	97
	Sine	75	98	98
	Song	77	99	99
In Hand	No-audio	69	85	91
	Sine	70	90	93
	Song	69	89	93

for the gyroscope sensor. As such, the obfuscation technique could be deployed with an operating system update. Obfuscation, however, adds extra noise and can therefore negatively impact the utility of the sensors (in contrast to calibration, which improves their utility). We explore the following techniques for adding noise –

- **Uniform noise**: highest entropy while having a bound.
- **Laplace noise**: highest entropy which is inspired by Differential Privacy [153, 154].
- **White noise**: affecting all aspects of a signal.

7.2.1 Uniform Noise

In this section we randomly choose offset and gain errors from a uniform range where we deduce the base ranges from our lab phones. We consider three variations of adding uniform noise to sensor data.

Basic Obfuscation: First, we consider small obfuscation values in the range that is similar to what we observed in the calibration errors in the previous section. Adding noise in this range is roughly equivalent to switching to a differently (mis)calibrated phone and therefore should cause minimal impact to the user. To add obfuscation noise, we compute $a^O = a^M \times g^O + o^O$, where g^O and o^O are the obfuscation gain and offset, respectively. Based on Figures 7.2 and 7.4, we choose a range of [-0.5,0.5] for the accelerometer offset, [-0.1,0.1] for the gyroscope offset, and [0.95,1.05] for the gain. For each session, we pick uniformly random obfuscation gain and offset values from the range; by varying the obfuscation values we make it difficult to fingerprint

repeated visits. Table 7.2 summarizes our findings when we apply obfuscation to all the sensor data obtained from our 30 lab smartphones. Compared to unaltered data (Table 4.5), data obfuscation seems to provide significant improvement in terms of reducing the average F-score. Depending on the type of audio stimulation, F-score reduces by almost 7–24% when smartphones are kept stationary on the desk and by 23–42% when smartphones are kept stationary in the hand of the user. The impact of audio stimulation in fingerprinting motion sensors is much more visible in these results. We see that F-score increases by almost 18–21% when a song is being played in the background (compared to the no-audio scenario); again, we expect this to be a consequence of audio-stimulation significantly impacting secondary features that come into play once primary features are obfuscated.

Table 7.2: Average F-score for obfuscated data under lab setting.

Device Placed	Stimulation	Avg. F-score (%)		
		Accelerometer	Gyroscope	Accelerometer+Gyroscope
On Desk	No-audio	43	73	75
	Sine	49	76	76
	Song	71	88	93
In Hand	No-audio	46	46	51
	Sine	42	49	57
	Song	55	63	72

Next, we apply similar techniques to the public (see Section 4.5.2 for more detail) and combined dataset (combining data from sections 4.5.1 and 4.5.2). We apply the same range of offset and gain errors to the raw sensor values before generating fingerprints. Table 7.3 and Table 7.4 summarizes our results for both presence and absence of audio stimulation. We see that F-score reduces by approximately 20–41% (compared to Table 4.6 and Table 4.7). We expect one of the reasons for the lower accuracy is the usage of a larger dataset, suggesting that for even larger datasets the impact of obfuscation is likely to be even more pronounced. In Section 7.5 we analyze the impact of obfuscation on a larger dataset.

Increasing Obfuscation Range: We now look at how the fingerprinting technique reacts to different ranges of obfuscation. Starting with our base ranges of $[-0.5, 0.5]$ and $[-0.1, 0.1]$ for the accelerometer and gyroscope offsets, respectively, and $[0.95, 1.05]$ for the gain, we linearly scale the ranges

Table 7.3: Average F-score for obfuscated data under public setting (63 phones) where smartphones were kept *on top of a desk*.

Stimulation	Avg. F-score (%)		
	Accelerometer	Gyroscope	Accelerometer+Gyroscope
No-audio	27	52	57
Sine	40	65	66

Table 7.4: Average F-score for obfuscated data under both lab and public setting (93 phones) where smartphones were kept *on top of a desk*.

Stimulation	Avg. F-score (%)		
	Accelerometer	Gyroscope	Accelerometer+Gyroscope
No-audio	26	50	55
Sine	41	69	75

and observe the impact on F-score. We scale all ranges by the same amount, increasing the ranges symmetrically on both sides of the interval midpoint.

For this experimental setup we only consider the combined dataset as this contains the most number of devices (93 in total). We also restrict ourselves to the setting where we combine both the accelerometer and gyroscope features because this provides the best result (as evident from all our past results). Figure 7.5 highlights our findings. As we can see increasing the obfuscation range does reduce F-score but it has a *diminishing return*. For 10x increment, the F-score drops down to approximately 40% and 55% for no-audio and audio stimulation respectively. Beyond 10x increment (not shown) the reduction in F-score is minimal (at most 10% reduction at 50x increment). This result suggests that simply obfuscating the raw values is not sufficient to hide all unique characteristics of the sensors. So far we have only manipulated the signal value but did not alter any of the frequency features and as a result the classifier is still able to utilize the spectral features to uniquely distinguish individual devices.

Enhanced Obfuscation: Given that we know that the spectral features are not impacted by our obfuscation techniques, we now focus on adding noise to the frequency of the sensor signal. Our data injection procedure is described in Algorithm 3. The main idea is to probabilistically insert a modified version of the current data point in between the past and current timestamp where the timestamp itself is randomly selected. Doing so will influence cubic interpolation of the data stream which in turn will impact

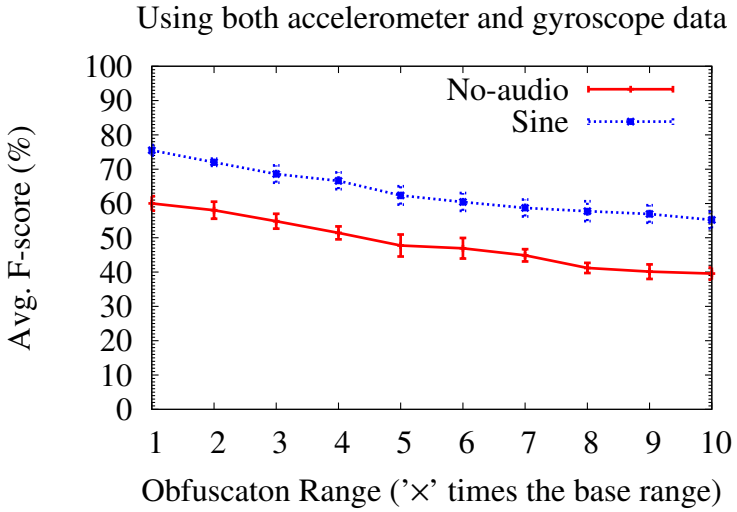


Figure 7.5: Impact of obfuscation range as the range is linearly scaled up from 1x to 10x of the base range.

the spectral features extracted from the data stream.

Algorithm 3 Obfuscated Data Injection.

Input: Time series Data (D, T), Probability Pr, Offset O, Gain G, Offset Range O_{range}, Gain Range G_{range}
Output: Modified time series Data (MD, MT)
$offset \leftarrow Null$
$gain \leftarrow Null$
$\#$ $Random(range)$: randomly selects a value in range
$j \leftarrow 1$
for $i = 1$ to $length(D)$ **do**
  #New data insertion
  if $i > 1$ and $Random([0,1]) < Pr$ **then**
    $offset \leftarrow Random(O_{range})$
    $gain \leftarrow Random(G_{range})$
    $MT[j] \leftarrow Random([T[i], MT[j-1])$
    $MD[j] \leftarrow D[i] \times gain + offset$
    $j \leftarrow j + 1$
  end if
  #Original Data
  $MD[j] \leftarrow (D[i] \times G + O$
  $MT[j] \leftarrow T[i]$
  $j \leftarrow j + 1$
end for
return (MD, MT)

To evaluate our approach we first fix an obfuscation range. We choose 10x of the base range from the previous section as our fixed obfuscation range. We then vary the probability of data injection from [0,1]. Figure 7.6 shows our findings. We can see that even with relatively small amount of data injection (in the order of 20–40%) we can reduce the average F-score to approximately 15–20% depending on the type of background stimulation applied.

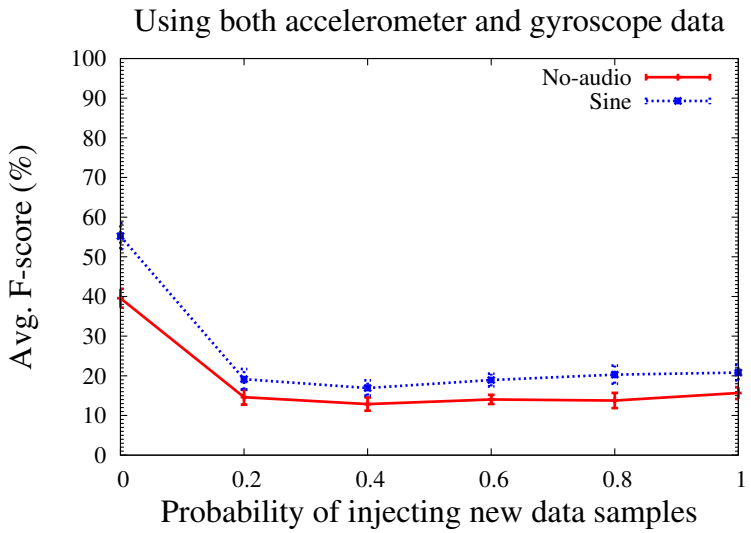


Figure 7.6: Impact of randomly inserting new data points.

7.2.2 Laplace Noise

Next, we adopted an approach inspired by differential privacy [153, 154] where we randomly selected offset and gain error from a Laplace distribution. From the definition of differential privacy [153], we know that a randomized function K gives ϵ-differential privacy if for all datasets D_1 and D_2 differing on at most one element, and all $S \subseteq Range(K)$,

$$\Pr[K(D_1) \in S] \leq e^{\epsilon} \Pr[K(D_2) \in S] \tag{7.6}$$

We can remap this setting into our own problem where we can think of each device as a single dataset, and K as the process of selecting random offset and gain error. S then becomes the outcome of applying random noise to raw sensor data. By changing ϵ we can control to what extent two device-output

distributions are alike. If we assume all features are affected by changing only the offset and gain errors along all three axes for both accelerometer and gyroscope then the probability of a randomly drawn sample belonging to a *particular device* (P) can be approximated as below:

$$\begin{aligned} P + (D-1) \times P' &= 1 \\ P + (D-1) \times P/e^{\epsilon} &\leq 1 \\ P(D - 1 + e^{\epsilon}) &\leq e^{\epsilon} \\ P &\leq \frac{e^{\epsilon}}{D - 1 + e^{\epsilon}} \end{aligned} \tag{7.7}$$

where D refers to the total number of devices and P' refers to the probability of a randomly drawn sample belonging to any other device. We can think of P as the probability of correct classification. Figure 7.7 shows how the upper bound of the probability of correct classification varies for different values of ϵ (setting $D = 93$ as we had 93 devices). We can see that as we increase the privacy budget, the maximum attainable correct classification also increases. This is excepted because increasing ϵ means the probability distributions from different devices vary more from each other which potentially increases the chance for distinguishing devices.

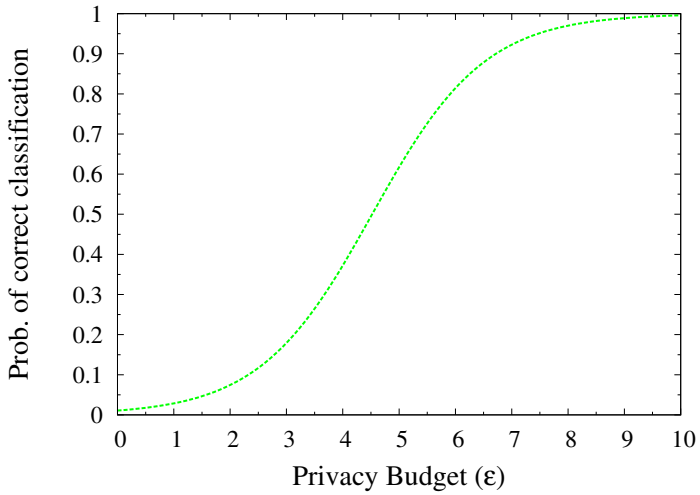


Figure 7.7: Approximation of the probability of correct classification under differential privacy approach where noise is modeled through only offset and gain errors along all three axes for both accelerometer and gyroscope.

In our setting we have offset and gain errors along 6 axes (xyz-axes for

both accelerometer and gyroscope), giving us a total of 12 dimensions. We equally distribute our privacy budget ϵ along all 12 dimensions and select noise along the $i-th$ dimension using the following Laplace distribution: $Lap(0, \beta_i)$ where $\beta_i = S_i/(\epsilon/12)$ and $S_i = \max(i-th$ Dimensional values$) - \min(i-th$ Dimensional values$)$, $i \in \{1, 2, ..., 12\}$. Figure 7.8 shows that as we increase ϵ (i.e., as we lower the scale parameter of the Laplace distribution), F-score also increases. But even with a relatively high privacy budget of $\epsilon = 10$ we see that F-score reduces from around 96% to 47–65% depending on the type of background stimulation we apply. Interestingly, if we compare Figure 7.8 with Figure 7.7 we can see that the upper bound for classification accuracy holds for all $\epsilon \geq 4$ but not for lower values of ϵ (i.e., $\epsilon < 4$). This suggests that assuming all features are influenced by only changing the offset and gain errors along 6 axes does not hold and there are certain features that are unaffected by any change in offset and gain errors.

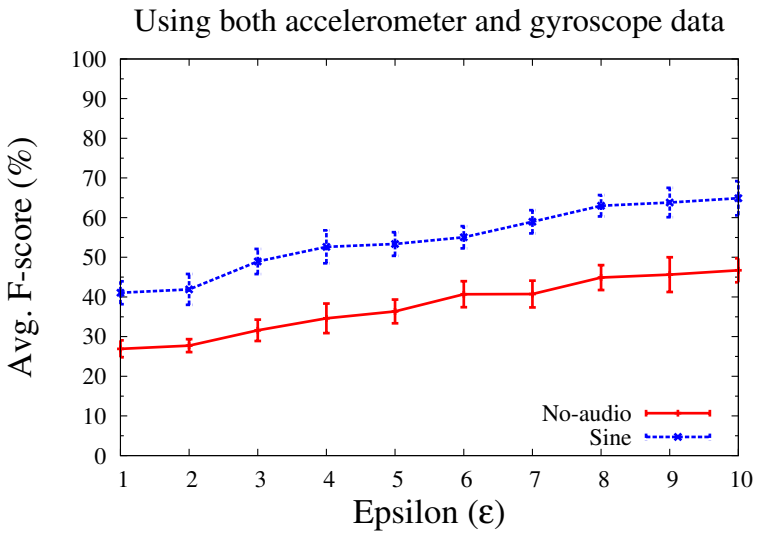


Figure 7.8: Impact of randomly selecting offset and gain error from a Laplace distribution inspired by differential privacy.

7.2.3 White Noise

From Figure 7.8 we see that even when $\epsilon = 1$ we can still achieve an F-score of 26–41%. We looked at the dominant features after applying random offset and gain error, and found that spectral features like *spectral irregularity*, *spectral attack slope* and *spectral entropy* are the top features in terms of

mutual information. This is understandable because changing the offset and gain have minimal impact on spectral features; we therefore next add Gaussian white noise to the signal after applying random offset and gain error from a Laplace distribution. For this experimental setup we fixed $\epsilon = 6$ (as this provides at least 0.5 privacy budget along all dimensions) and varied the signal-to-noise ratio (SNR) from 0.5 to 10. Figure 7.9 highlights F-scores for different values of SNRs. We can see that F-score drops from 40–55% to 20–30% when Gaussian white noise is added to the signal for $\epsilon = 6$ and SNR= 10. For lower SNR we see that the F-score reduces even further but that comes at the cost to mixing a large amount of noise to the original signal which could adversely affect the utility of the motion sensors as we will in Section 7.4.

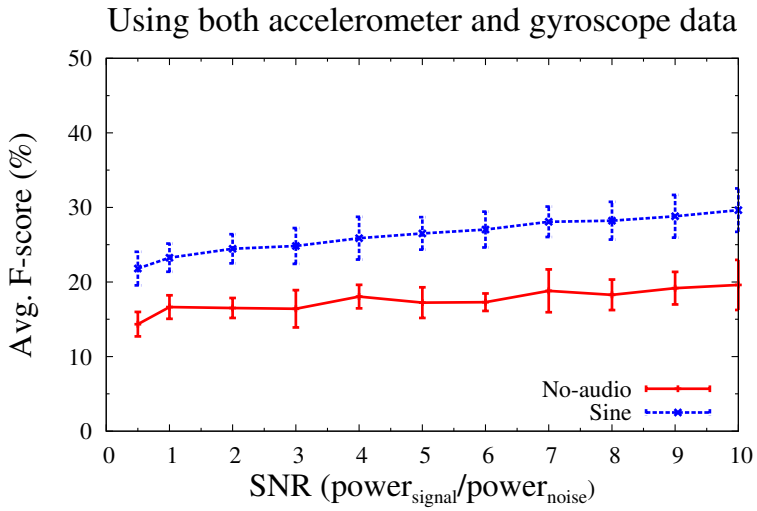


Figure 7.9: Impact of Gaussian white noise on F-score.

7.3 Sensor Quantization

The basic idea behind quantization is that human brain cannot discriminate minute changes in angle or magnitude. As a result if the raw values of a sensor are altered slightly, it should not adversely impact the functionality of the sensor. We perform quantization in the polar coordinate system as it is easy to perceive (shown in Figure 7.10).

So, our first task is to covert the accelerometer data into its equivalent

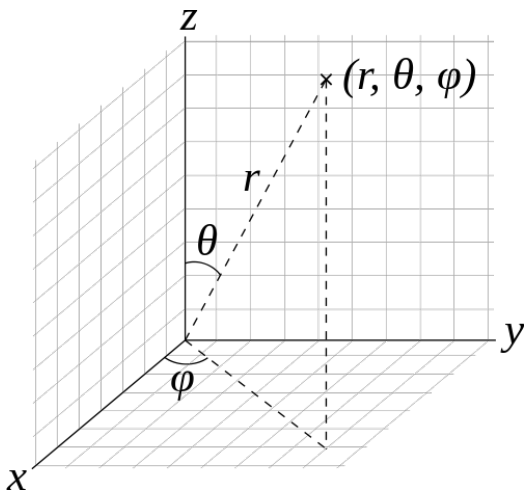


Figure 7.10: Convertion from Cartesian coordinate system (x, y, z) to Polar coordinate system (r, θ, ϕ).

polar vector form as shown below:

$$radius, r = \sqrt{a_x^2 + a_y^2 + a_z^2}$$
$$inclination, \theta = \cos^{-1} \frac{a_z}{r}$$
$$azimuth, \psi = \tan^{-1} \frac{a_y}{a_x}$$

where $< a_x, a_y, a_z >$ represent the accelerometer data in the Cartesian coordinate system. Since gyroscope provides rotational rate in $\texttt{rads}^{-1}$, we do not perform any conversion for gyroscope data. Next we pass our sensor data through the following *quantization* function:

```
function quatization(val,type,bin_size){
  // val: raw sensor value
  // type: data type (angle or magnitude)
  // bin_size: quantization size
  return round(val/bin_size)*bin_size;
}
```

For angle related data (θ,ψ and gyroscope data) we set $bin_{size} = 6^\circ$ while for magnitude (i.e., radius) we set $bin_{size} = 1\ \texttt{ms}^{-2}$. In other words, we place angles into 6 degree bins and for accelerometer magnitude we map it to the nearest integer. After performing quantization on the accelerometer data, we

remap it to the Cartesian coordinate system using the following equations:

$$a_x = r \sin\theta \cos\psi$$
$$a_y = r \sin\theta \sin\psi$$
$$a_z = r \cos\theta$$

7.3.1 Fingerprinting Quantized Data

We now evaluate how quantization defends against sensor fingerprinting. Figure 7.11 summarizes our findings for the *no-audio* setting where the devices are kept on top a desk while browsing our web page. We see that compared to original raw data (Tables 4.5, 4.6 and 4.7) F-score reduces by 32% for lab setting and around 50% for public and combined setting. Compared to basic obfuscation (Tables 7.2, 7.3 and 7.4) we see that quantization performs slightly better in lowering F-score. Quantization method has similar outcomes to Laplace noise when ϵ is set to 10 (see Figure 7.8 for more details). However, we see that quantization is not as effective as enhanced obfuscation (comparing with Figure 7.6) or white noise (comparing with Figure 7.9) in lowering F-score. This is understandable because quantization does not affect any of the spectral properties of the signal, whereas both enhanced obfuscation and white noise alter spectral properties of the signal.

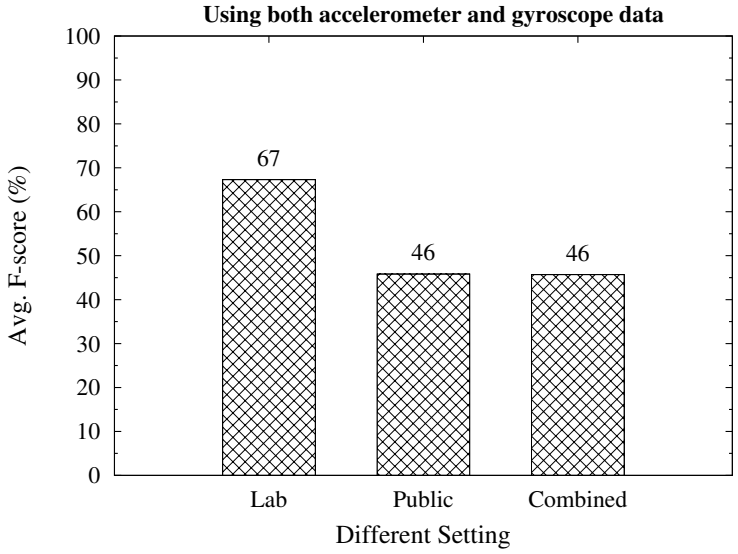


Figure 7.11: Impact of sensor quantization on F-score.

7.4 Determining Feasible Countermeasures

In the previous section we explored many possible countermeasures and showed that all the countermeasures are effective in lowering fingerprinting accuracy. However, as there are many legitimate application of motion sensors we would like to deploy a countermeasure that has negligible impact on the utility of the motion sensors. We, therefore, first explore how our countermeasures impact the utility of a simple yet popular accelerometer-based application known as *Step Counter* [155] (also known as *Pedometer*) that uses accelerometer readings to determine the number of steps taken by a user. Such analysis will enable us to shortlist the countermeasures that can be readily deployed without impacting the utility of the motion sensors. To carry out this analysis we prototype a *Step Counter* application where we use a web page to collect sensor data. In our experimental setting, we ask the participant to take 20 steps while holding the phone in his/her hand and this whole process in repeated 10 times.

To get an understanding of how the different countermeasure streams look we plot the magnitude of the accelerometer for the first three seconds in Figure 7.12. We can see from the figure that certain mitigation techniques such as *Enhanced obfuscation* and *White noise* disrupt the original signal significantly. We will later on see that these two schemes have significant adverse effect on the utility of the motion sensors. If we remove these two schemes the plot clears up as shown in Figure 7.13. We can see that all the renaming schemes retain the structural properties of the original signal and thus should not significantly impact the utility of the motion sensors.

To quantitatively analysis the impact of the different countermeasures we replay the step motion data 100 times under each countermeasure scheme and compute the average number of steps taken by the user. Table 7.5 highlights our findings for different forms of countermeasures. In the following sections we briefly discuss how each countermeasure scheme impacts the utility of the motion sensors in detail.

7.4.1 Utility under Calibration

From Table 7.5 we see that calibration does not have a significant effect on accuracy. In general, we would expect calibration to *improve* accuracy but

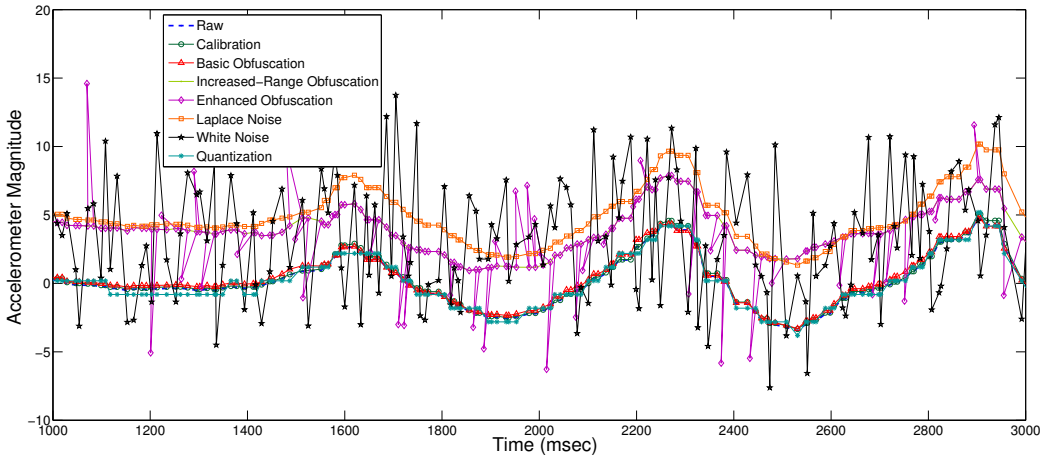


Figure 7.12: Accelerometer magnitude for different mitigation schemes.

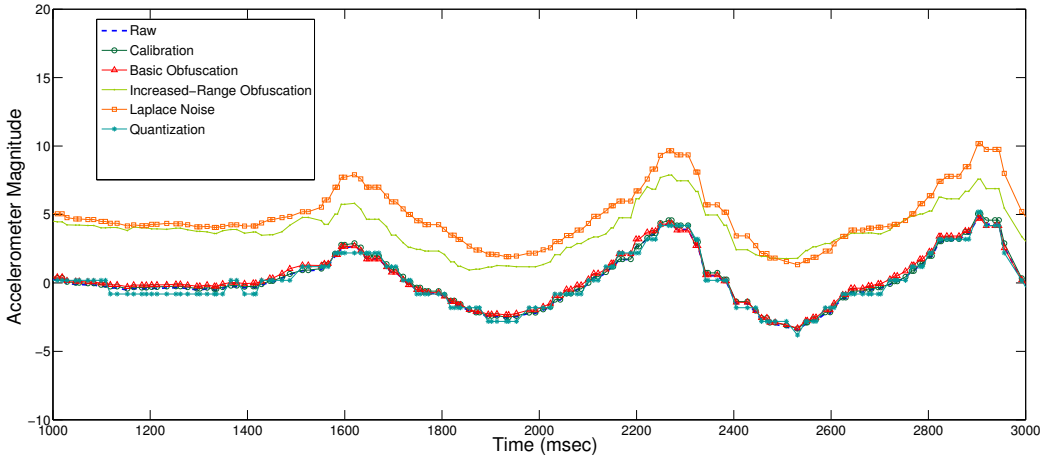


Figure 7.13: Accelerometer magnitude after removing disruptive countermeasures.

our calibration process is imperfect (manually done) and it is possible that it introduces very minor errors.

7.4.2 Utility under Obfuscation

Impact of Uniform Noise on Utility: Basic obfuscation introduces errors that are commensurate with calibration errors of actual devices and thus also has minimal impact on accuracy. Increasing the obfuscation range introduces errors that are still within acceptable range. However, introducing new data points makes the accelerometer readings significantly less reliable, and we observe this effect in the computed step counts.

Table 7.5: Privacy vs. Utility tradeoff for different countermeasures.

Stream Type		Step Count		Avg.
		Mean	Std. Deviation	F-score (%)
Original Stream		20	0	96[a]
Calibrated Stream		20.1	0.3	**97**[b]
Obfuscated Stream	Basic Obfuscation[c]	20.1	0.6	55[a]
	Increased-Range Obfuscation[d]	20.6	1.4	40[a]
	Enhanced Obfuscation[e]	**42.9**	**15.1**	15[a]
	Laplace Noise[f]	20.5	1.1	40[a]
	White Noise[g]	**73.9**	**9.1**	20[a]
Quantized Stream[h]		20.4	0.7	46[a]

[a] For 93 devices
[b] For 30 lab phone
[c] Base range: $offset = [-0.5, 0.5]$, $gain = [0.95, 1.05]$
[d] `10x` of base range [e] `10x` of base range with 0.4 injection probability
[f] For $\epsilon = 6.0$
[g] For $\epsilon = 6.0$ and $SNR = 5$
[h] Angles quantized to 6° bins and magnitudes rounded to the nearest integer

Impact of Laplace Noise on Utility: We rerun our step counter application on sensor data where we select offset and gain error from a Laplace distribution while varying ϵ. Figure 7.14 shows how step count evolves for different levels of privacy budget (ϵ). We see that as we increase ϵ, step count converges to the expected value with negligible deviation. For $\epsilon \geq 6$ the confidence interval is negligible, i.e., for $\epsilon \geq 6$ the impact of noise is minimal. Notably, on Figure 7.8, we can see that for $\epsilon = 6$, we get significantly lower classification accuracy than using low levels of uniform noise (see Figure 7.5). This suggests that Laplace noise may achieve a better tradeoff between privacy and utility; however, from Table 7.5 we see that Laplace noise performs slightly worse (slightly higher *standard deviation*) compared to basic obfuscation and quantization.

Impact of White Noise on Utility: Given that we see adding white noise provides low F-scores we wanted to see what kind on impact it would have on sensor utility. To evaluate this we rerun our step counter application on sensor data after applying Gaussian white noise. Figure 7.15 highlights the computed step counts for different SNRs. We see that adding white noise has drastic consequences as it increases the number of steps counted significantly, even at high signal-to-noise ratios.

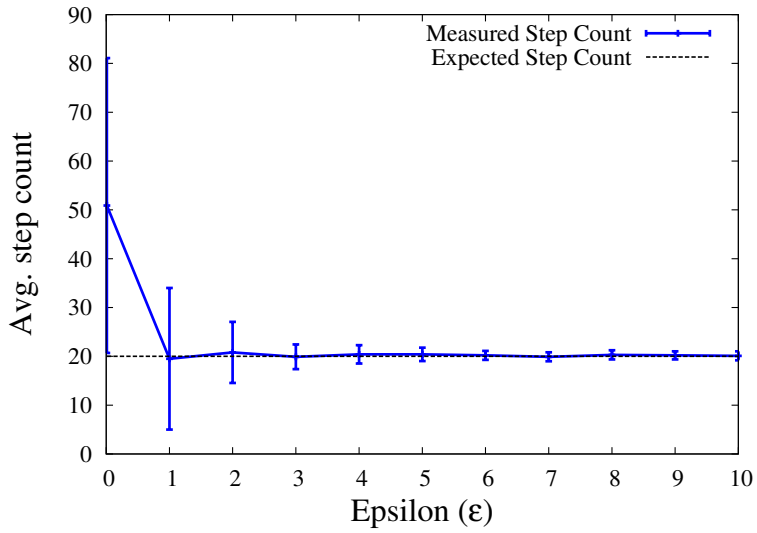


Figure 7.14: Impact of Laplace noise on utility.

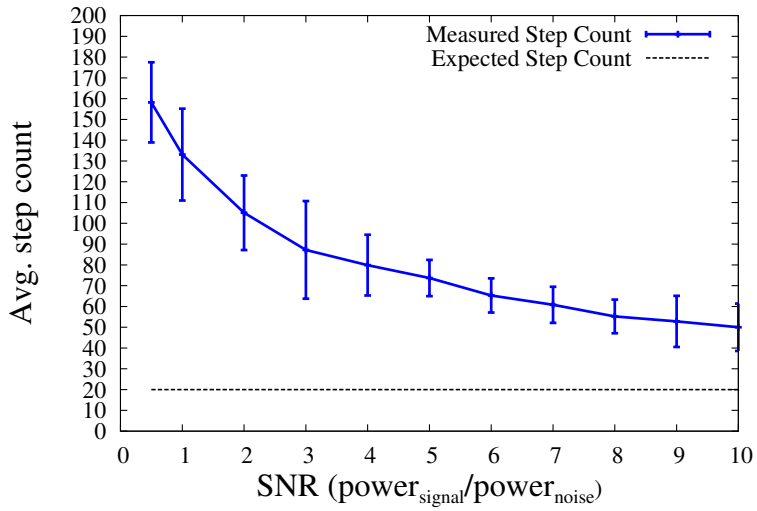


Figure 7.15: Impact of Gaussian white noise on sensor utility.

7.4.3 Utility under Quantization

From Table 7.5 we see that quantization has minimal impact on the utility of the step counter. This suggests even if we quantize the sensor data into bigger bins we can still obtain acceptable utility from the motion sensors. Thus, a simple yet effective defense against sensor fingerprinting is to lower the resolution of motion sensors.

7.4.4 Deployment Considerations

We envision our obfuscation technique as an update to the mobile operating system. From Table 7.5 we see that *calibration* represents one side of the tradeoff spectrum with high utility but low privacy; *enhanced obfuscation* and *white noise* provide the opposite side of the spectrum with low utility and high privacy. The remaining four techniques provide better tradeoff between privacy and utility. However, we see that *basic obfuscation* and *quantization* provide slightly better utility (i.e., closer to real *mean* with smaller *standard deviation*) compared to *Laplace noise* and *increased-range obfuscation*. Due to limited budget (both in terms of participation time and reward money) we give more emphasis on *basic obfuscation* and *quantization* in the following sections to conduct a large-scale user study under a more realistic web setting.

7.5 Effectiveness of Countermeasures at Large-Scale

Given that we have identified that *basic obfuscation* and *quantization* have minimal impact on the utility of the motion sensors, we now want to verify their effectiveness against fingerprinting large-scale smartphones in a real world setting. For this setup we run our fingerprinting technique under three setting: baseline, obfuscation and quantization. We merge our lab and public dataset with the data collected from Amazon's Mechanical Turk (i.e., merged dataset that we use in Chapter 5). For each setting we then evaluate F-score for both random forest and k-NN (with LDML). Table 7.6 shows our results for devices with at least 3 training samples (there are 545 such devices).

We can see that both countermeasure schemes significantly reduce the F-score with obfuscation performing slightly better than quantization. Next, we see how the countermeasure schemes react to different numbers of devices. To evaluate this we first model intra- and inter-device distances for both obfuscated and quantized datasets. We adopt techniques similar to Section 5.5.2 where we derive parametric distributions to model intra- and inter-device distances. We then use our proposed k-NN simulator (Algorithm 2) to predict classification accuracy for large pool of devices. Figure 7.16 highlights our findings. We only evaluate accuracy for $k = 1$ as this was shown to have the best overlap with real world results (see Section 5.5.3 for more details).

We also plot the corresponding real world results for a k-NN classifier. We see that irrespective of the device number classification accuracy reduces significantly under both countermeasure schemes. And the estimated accuracy reduces to almost zero for any number of devices greater than 1000. These results indicate that simple countermeasures can thwart device fingerprinting significantly.

Table 7.6: Comparing obfuscation and quantization with baseline for 545 devices.

Scheme	Avg. F-score(%)	
	k-NN with LDML	Random Forest
Baseline	50	78
Quantization	17	32
Obfuscation	7	26

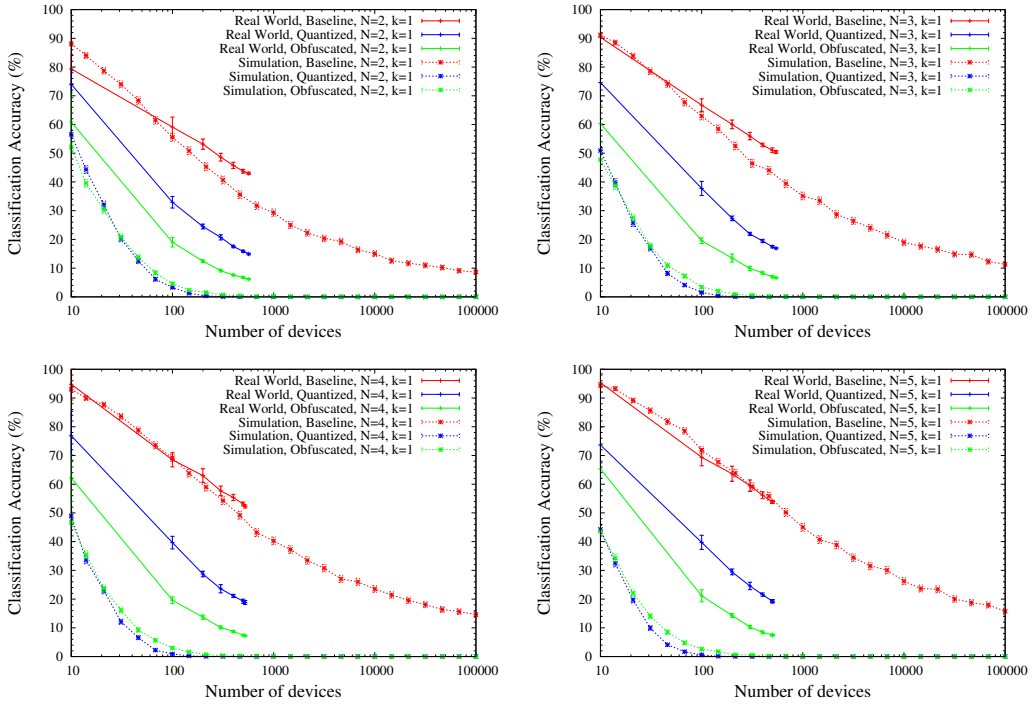


Figure 7.16: Comparing large-scale classification accuracy for obfuscation and quantization.

7.6 Large-Scale User Study of Privacy vs. Utility

The above countermeasures degrade the readings from the motion sensors somewhat and we wanted to better understand the impact of the countermeasures on the utility of the sensors to web applications, specially user-interactive applications. Of course, motion sensors have a wide range of uses, from simple orientation detection to activity classification, step counting, and other health metrics. Many of these, however, are deployed in application form, whereas we wanted to focus on the threat of fingerprinting by web pages. We performed a survey of web pages to identify how motions sensors are actually used. We found that one of the most common application of motion sensors was to detect orientation change in order to adjust page layout (see Section 6.3.2 for more detail); such a drastic change in the gravity vector will be minimally impacted by countermeasures. We did, however, find several instances where web pages used the motion sensors as a means of gesture recognition in the form of tilt-based input controlling a video game.

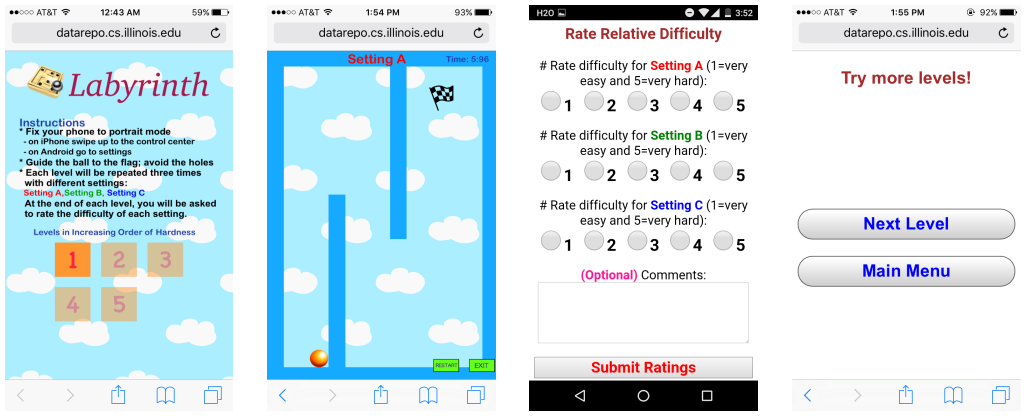
To study the impact of countermeasures on the utility of such tilt-based controls, we carried out a user study where participants were asked to play a game using tilt control while we applied privacy countermeasures to their motion sensor data. We then evaluated the impact of the countermeasures through both objective metrics of in-game performance, as well as subjective ratings given by the participants. Our study was approved by our institutional research board (IRB).

7.6.1 Study Design

After receiving some information about the study, our participants were invited to play a game using their personal smartphone (Figure 7.17(a)).[4] The objective of the game is to roll a ball to its destination through a maze, while avoiding traps (hitting a trap restarts the level from the beginning). The game had five levels, which the participants played in order of increasing difficulty. Each level was played three times with different privacy countermeasures applied: baseline (no countermeasures), obfuscation, and quantization. The order of countermeasure settings was randomized for each participant and for each level, and not revealed to the participants. After completing

[4]https://web.engr.illinois.edu/~das17/PrivacyVsUtility.html

a level three times, the participants were asked to rate each of the three settings in terms of difficulty of controlling the game on a scale of 1 to 5 (1 meaning 'very easy' whereas 5 referred to 'very hard'). Participants also were invited to provide free-form feedback (Figure 7.17(c)). Their ratings and feedback, along with the settings and metrics regarding the time spent on each game, and the number of times the game was restarted due to traps, were then sent to our server for analysis.



(a) Instruction page (b) Level 1 (c) Feedback form (d) Next Level form

(e) Level 2 (f) Level 3 (g) Level 4 (h) Level 5

Figure 7.17: Game interface. The object is to roll the ball to the flag while avoiding traps by tilting the smartphone. The user is then asked for feedback about the relative difficulty of each level using different privacy settings.

After completing a level, a user is invited to play the next level. Users were required to play levels in order of increasing difficulty, but participants were allowed to replay previous levels. We identified such repeat plays by setting a cookie in a user's browser and discarded repeat plays in our analysis.

7.6.2 Study Results

We recruited users through institutional mailing lists, social media, as well as Amazon's Mechanical Turk. We collected data from 201 users via Amazon's Mechanical Turk and 206 users that were recruited through other means over a period of one month, for a total of 407 users (covering 144 different device models as shown in Appendix C); several users' data had to be discarded due to irregularities in data collection. Note that not all users played through all five levels, as shown in Table 7.7. Note that Mechanical Turk users had to complete five levels to receive their reward, but in some cases we were not able to receive some of their data due to network congestion at our server.

Table 7.7: Number of users that completed the first n levels recruited through Amazon's Mechanical Turk and other means.

Levels completed	**MTurk**	**non-MTurk**	Total
1	0	26	26
1–2	1	14	15
1–3	0	34	34
1–4	91	67	158
1–5	107	63	170
Total	199	204	403

We found that, when considering the entire dataset, the choice of privacy protection method did not significantly influence the subjective ratings assigned to the level (χ^2 test, $p = 0.34$) nor the objective metrics of the game duration (pairwise t-tests, $p = 0.10$ and 0.75 comparing baseline to obfuscation and quantization, respectively) or the number of restarts due to traps (pairwise t-tests, $p = 0.11$ and 0.47). However, as expected, all difficulty metrics were significantly impacted by which level the person was playing, as shown in Figure 7.18.

Furthermore, we observed a significant training effect between the first and second time a user played the level (each level is played a total of 3 times using different privacy methods), as seen in Figure 7.19. Interestingly, this was not reflected in the subjective ratings (as verified by a χ^2 test for each level), suggesting that participants corrected for the training effect during their reporting. There was a smaller training effect between the second and third time a level was played; the improved performance was statistically

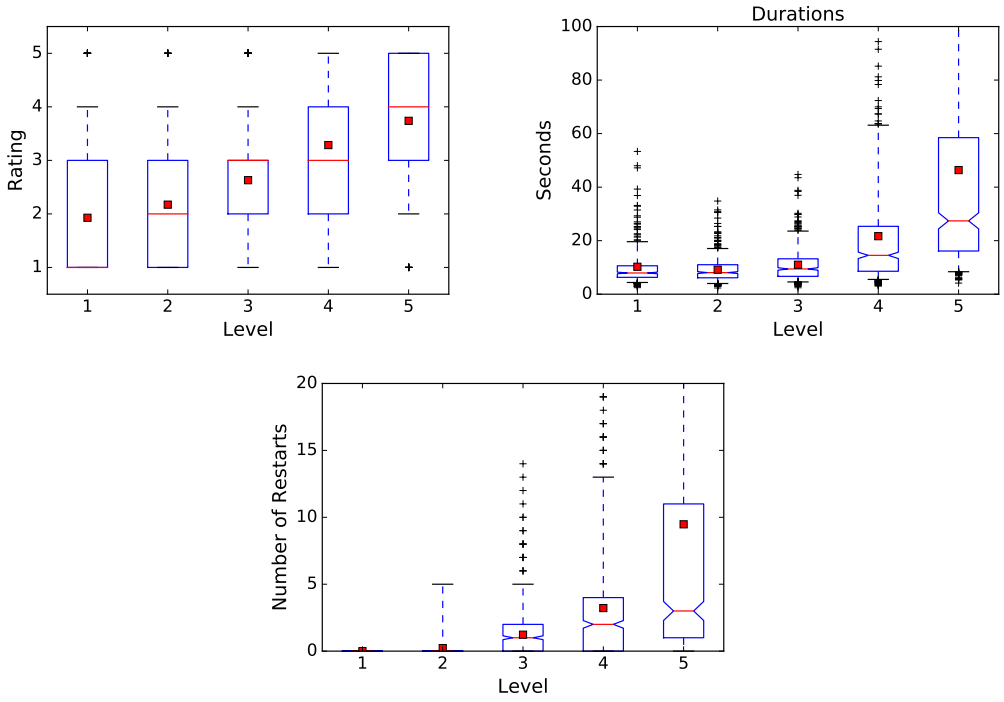


Figure 7.18: Subjective and objective difficulty metrics increase across game levels. Box plots show the median (horizontal line) and the interquartile range (box), while whiskers show the range between the 5th and 95th percentile, with the outliers being individually represented. The notch in the box denotes the 95% confidence interval around the median.

significant only for durations of levels 4 and 5 and for the number of restarts on level 5; which makes sense given the difficulty of these levels.

We therefore compared the difficulty of metrics for different privacy methods across only the second and third attempts at a level, discarding the first attempt as training. We show the results for all levels in Figure 7.20. Significance tests fail to detect any differences between the difficulty metrics when privacy methods are applied on any level.[5]

7.6.3 Limitations

Although the study failed to detect a significant impact of privacy methods on utility, it does not definitively show that no impact exists—failure to reject

[5]The raw p-value comparing the number of restarts on level 5 between baseline and obfuscated case is 0.025 but note that this is not significant at a $p < 0.05$ level after the Bonferroni correction is applied.

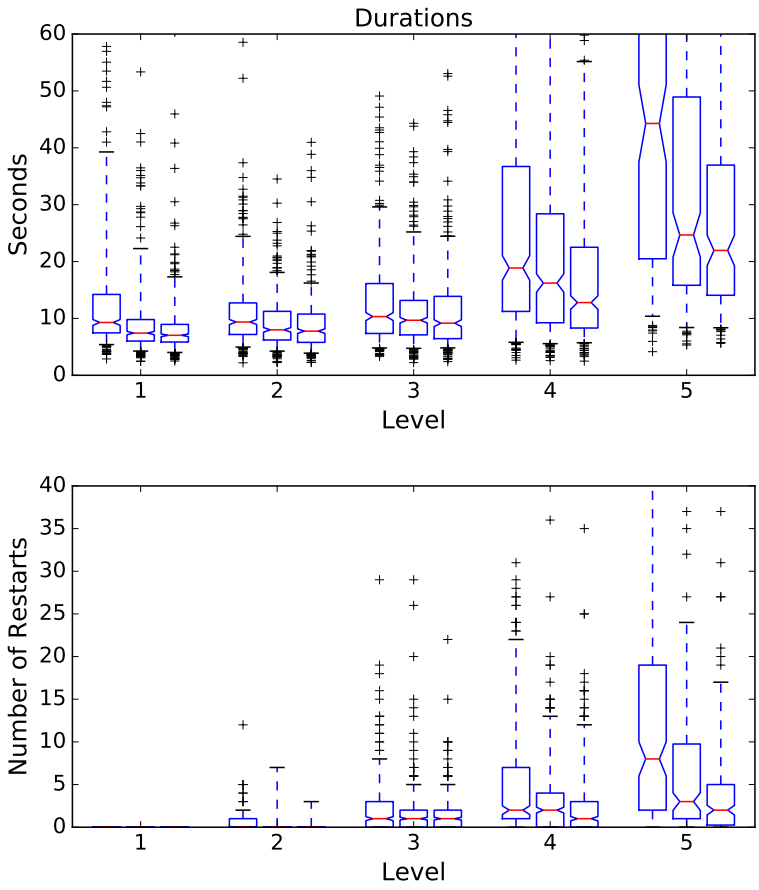


Figure 7.19: Game durations and number of restarts, as each level is played three times. A large training effect is observed between the first and second attempt, with a smaller effect between the second and third.

a null hypothesis does not demonstrate that the null hypothesis is true. In particular, given the large variance in game performance across users, as seen in, e.g., Figure 7.18, we would like to compare how different privacy methods change a single user's performance; however, given the low impact of privacy protection we have observed so far, we would need to modify our study to reduce or eliminate the training effect. Additionally, we tested our privacy methods in a short game, and perhaps in games with a longer duration some effects would materialize. However, we feel our results are promising in showing that users may not have to lose much utility to employ privacy protection methods.

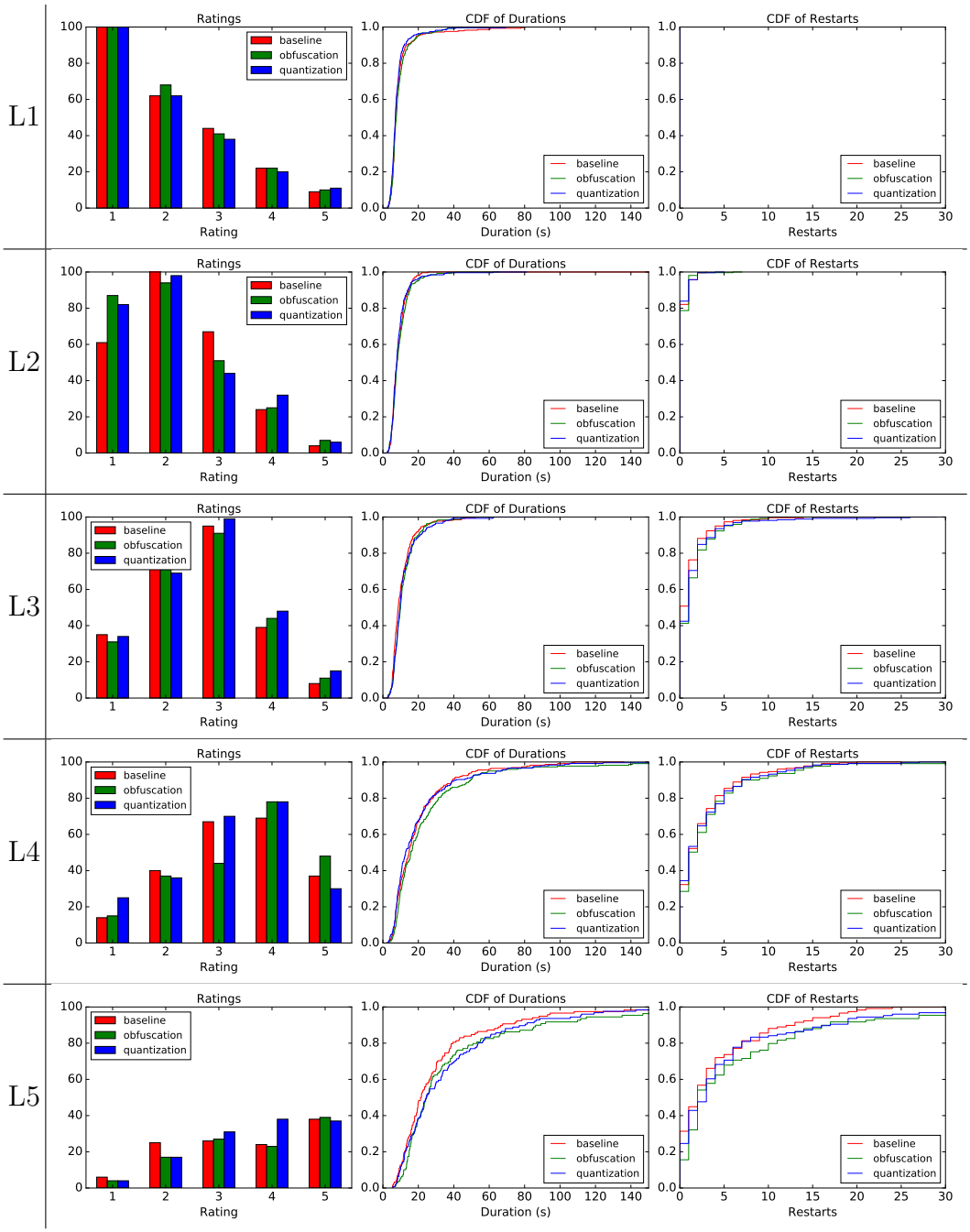


Figure 7.20: Impact of privacy method on subjective and objective ratings, when considering second and third attempts only. Shown are the histogram of subjective ratings and CDFs of game durations and number of restarts for all 5 levels. No significant difference found in any of the metrics.

7.7 Summary

We evaluate the tradeoff between privacy and utility as realized by two different fingerprinting mitigation strategies. While many applications of sensor

data are unlikely to be affected, our user study shows that even for sensitive applications that use motion sensors as control input, there is no significant impact of privacy mitigation techniques on the usability of motion sensors in this context, according to both subjective and objective metrics.

CHAPTER 8

CONCLUSION

The widespread adoption of smartphone and their increased computing capabilities are creating new threats for user privacy. As a result, we are in the middle of a war over user privacy on the web. While users are slowly becoming more concern about their privacy on smartphones, there are still may issues which have not been well studied and hence not well known to general users. In this dissertation we show that onboard sensors such as microphone, accelerometer and gyroscope can be exploited as side-channels to uniquely fingerprint smartphones. Fingerprinting smartphones enable publishers to track users across multiple applications or websites. Publishers can then sell this behavioral information to the ad networks. To make things worse some of these sensors such as accelerometer and gyroscope can be accessed surreptitiously without any explicit user permission, so a user might not even be aware of such fingerprinting mechanism.

In this dissertation we first look at how onboard acoustic hardware such as microphones and speakers can be used to track smartphones. Microphone and speaker are fundamental components present in any smartphone, thus fingerprinting smartphones through microphones and speakers creates a serious privacy concern for users. We found that the manufacturing imperfections of microphones and speakers are substantial and prevalent enough that we can reliably track phones by looking at the spectral properties of the transmitted and/or recorded audio signals. Surprisingly, you only need to extract a few spectral features from the audio signal to track the device responsible to generating and/or recording the audio signal. We found $MFCCs$ as the dominant features (at times the only set of features) required to uniquely fingerprint smartphones. With 50 smartphones we were able to uniquely fingerprint them with an average F-score of 98%. We also show the feasibility of our approach even in the presence of ambient background noise. Our results indicate that fingerprinting smartphones through onboard microphone

and speaker is a privacy concern and as a word caution we would recommend users to be careful about providing applications or websites access to microphone.

We next investigate if motion sensors can also be exploited to generate a unique fingerprint for the device. Interestingly, motion sensor data is considered less sensitive and as a result no explicit permission is required to access the motion sensors such as accelerometer and gyroscope. HTML5 enables websites to directly access motion sensors without the user even knowing about it. We develop our own web page to collect sensor data from both our lab phones and participants through institutional mass email and social media. We then show that it is indeed possible to fingerprint smartphones by exploiting the manufacturing imperfections of onboard motion sensors. Both accelerometer and gyroscope can be used individually to fingerprint smartphones but we obtain the best result when we combine features from both the accelerometer and gyroscope. We were able to fingerprint 93 devices with an average F-score of 96% when devices were kept stationary on top of a flat surface. We also show that we can obtain similar fingerprinting accuracy even when the device is kept in the hand of the user while the user is sitting down silently. Finally, we showcase that even though our fingerprinting technique is influenced by various environmental factors such as temperature and temporal wear and tear, we can still retain an F-score of greater than 75%.

Given that we were able to showcase that motion sensor fingerprinting is feasible with around 100 devices, we next extend our data collection process to gather data from a total of 610 devices; bulk of which comes from participants recruited through Amazon's Mechanical Turk. We then rerun our fingerprinting technique over this large dataset and show that we can still identify devices with an F-score of 86% with only 25 seconds (equivalent to 5 training samples) worth of motion sensor data from each device. To estimate accuracy with even a larger device population, we derive intra- and inter-device distance distribution from this large dataset and use these distributions to emulate k-NN classifier. A conservative estimation of classification accuracy was found to be in the range of 10–16% with 100 000 devices. This result suggests that motion sensors alone might not to sufficient to distinguish users in large populations, but when combined with other browser-based fingerprints they are likely to generate unique fingerprints even among large

populations.

To get a better understanding of our websites access motion sensors we conduct a large-scale measurement study where we analyze how many of the top 100 000 Alexa websites access motion sensors. We found that around 1% of these websites access motion sensor data. We also determine 8 broad use cases for accessing motion sensor data; distressingly, most of the websites accessing motion sensor data send such data to a third party website. This raises suspicion as to how third party websites are utilizing sensor data.

Finally, we focus heavily on devising countermeasures against motion sensor fingerprinting as such sensors can be accessed by applications and/or websites without any user permission. We propose three major countermeasure schemes – sensor calibration, data obfuscation and sensor quantization. We also explore different variations of data obfuscation. All of our countermeasures are effective in reducing fingerprinting accuracy, however, some countermeasures adversely impact the utility of the motion sensors. We found that manually calibrating certain sensors like gyroscope was not feasible and certain data obfuscation schemes require prior knowledge of the underlying application to be effective. Through a simple 'step counter' application we were able to showcase that basic data obfuscation and sensor quantization were the most promising countermeasures that could be readily applied to existing browsers. To showcase that these two countermeasures are benign to applications that interactively use motion sensors, we conduct a large-scale user study where users were asked to play a game by tilting their smartphone. After analyzing both subjective and objective feedback from a total of 403 users we were able to show that our proposed privacy mitigation techniques did not have any significant impact on the usability of motion sensors.

As a final statement we want to convey that as smartphones are becoming more intelligent by employing more sensors, we should also become more aware of their potential privacy risks. In this context, our work not only provides foundation towards understanding the privacy risks of fingerprinting smartphones through acoustic and motion sensors, but also provides simple yet effective mitigations against some of these fingerprinting techniques that can be readily adopted by web browsers today.

REFERENCES

[1] "2 Billion Consumers Worldwide to Get Smart(phones) by 2016." [Online]. Available: http://www.emarketer.com/Article/2-Billion-Consumers-Worldwide-Smartphones-by-2016/1011694 (Date last accessed 05-June-2016).

[2] "IPhone and Android Apps Breach Privacy." [Online]. Available: http://online.wsj.com/article/SB10001424052748704694004576020083703574602.html (Date last accessed 05-June-2016).

[3] K. Mahaffey and J. Hering, "App Attack: Surviving the Explosive Growth of Mobile Apps," 2010. [Online]. Available: https://media.blackhat.com/bh-us-10/presentations/Mahaffey_Hering/Blackhat-USA-2010-Mahaffey-Hering-Lookout-App-Genome-slides.pdf

[4] M. Egele, C. Kruegel, E. Kirda, and G. Vigna, "PiOS: Detecting Privacy Leaks in iOS Applications," in *Proceedings of the 17th Annual Network and Distributed System Security Symposium*, ser. NDSS '11, 2011.

[5] C. Gibler, J. Crussell, J. Erickson, and H. Chen, "AndroidLeaks: Automatically Detecting Potential Privacy Leaks in Android Applications on a Large Scale," in *Proceedings of the 5th International Conference on Trust and Trustworthy Computing*, ser. TRUST'12, 2012, pp. 291–307.

[6] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, "TaintDroid: An Information-flow Tracking System for Realtime Privacy Monitoring on Smartphones," in *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'10, 2010, pp. 1–6.

[7] Z. Yang, M. Yang, Y. Zhang, G. Gu, P. Ning, and X. S. Wang, "AppIntent: Analyzing Sensitive Data Transmission in Android for Privacy Leakage Detection," in *Proceedings of the 20th ACM SIGSAC Conference on Computer Communications Security (CCS)*, 2013, pp. 1043–1054.

[8] "Changes to the rules on using cookies and similar technologies for storing information." [Online]. Available: http://www.allaboutcookies.org/privacy-concerns/new-european-laws.html (Date last accessed 05-June-2016).

[9] "Do Not Track. Universal Web Tracking Opt Out." [Online]. Available: http://donottrack.us/ (Date last accessed 05-June-2016).

[10] G. Acar, M. Juarez, N. Nikiforakis, C. Diaz, S. Gürses, F. Piessens, and B. Preneel, "FPDetective: dusting the web for fingerprinters," in *Proceedings of the 2013 ACM SIGSAC conference on Computer and Communications Security (CCS)*, 2013, pp. 1129–1140.

[11] P. Eckersley, "How Unique is Your Web Browser?" in *Proceedings of the 10th International Conference on Privacy Enhancing Technologies (PETS)*, 2010, pp. 1–18.

[12] K. Mowery and H. Shacham, "Pixel perfect: Fingerprinting canvas in HTML5," in *Proceedings of Web 2.0 Security and Privacy Workshop (W2SP)*, 2012.

[13] N. Nikiforakis, L. Invernizzi, A. Kapravelos, S. Van Acker, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna, "You are what you include: large-scale evaluation of remote javascript inclusions," in *Proceedings of the 19th ACM SIGSAC conference on Computer and Communications Security (CCS)*, 2012, pp. 736–747.

[14] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan, and C. Diaz, "The Web never forgets: Persistent tracking mechanisms in the wild," in *Proceedings of the 21st ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2014, pp. 674–689.

[15] "Mobile apps overtake PC Internet usage in U.S." [Online]. Available: http://money.cnn.com/2014/02/28/technology/mobile/mobile-apps-internet/ (Date last accessed 05-June-2016).

[16] "We Spend More Time On Smartphones Than Traditional PCs: Nielsen." [Online]. Available: http://www.ibtimes.com/we-spend-more-time-smartphones-traditional-pcs-nielsen-1557807 (Date last accessed 05-June-2016).

[17] "U.S. Mobil App Report." [Online]. Available: http://www.ella.net/pdfs/comScore-US-Mobile-App-Report-2014.pdf (Date last accessed 05-June-2016).

[18] "Percentage of all global web pages served to mobile phones from 2009 to 2016." [Online]. Available: http://www.statista.com/statistics/241462/global-mobile-phone-website-traffic-share/ (Date last accessed 05-June-2016).

[19] T. Hupperich, D. Maiorca, M. Kührer, T. Holz, and G. Giacinto, "On the Robustness of Mobile Device Fingerprinting: Can Mobile Users Escape Modern Web-Tracking Mechanisms?" in *Proceedings of the 31st Annual Computer Security Applications Conference (ACSAC)*. ACM, 2015, pp. 191–200.

[20] J. Spooren, D. Preuveneers, and W. Joosen, "Mobile Device Fingerprinting Considered Harmful for Risk-based Authentication," in *Proceedings of the 8th European Workshop on System Security (EuroSec)*. ACM, 2015, pp. 6:1–6:6.

[21] "Does Alexa have a list of its top-ranked websites?" [Online]. Available: https://support.alexa.com/hc/en-us/articles/200449834-Does-Alexa-have-a-list-of-its-top-ranked-websites (Date last accessed 05-June-2016).

[22] A. Das, N. Borisov, and M. Caesar, "Fingerprinting Smart Devices Through Embedded Acoustic Components," *CoRR*, vol. abs/1403.3366, 2014. [Online]. Available: http://arxiv.org/abs/1403.3366

[23] A. Das, N. Borisov, and M. Caesar, "Do You Hear What I Hear?: Fingerprinting Smart Devices Through Embedded Acoustic Components," in *Proceedings of the 21st ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2014, pp. 441–452.

[24] A. Das, N. Borisov, and M. Caesar, "Exploring Ways To Mitigate Sensor-Based Smartphone Fingerprinting," *CoRR*, vol. abs/1503.01874, 2015. [Online]. Available: http://arxiv.org/abs/1503.01874

[25] A. Das, N. Borisov, and M. Caesar, "Tracking Mobile Web Users Through Motion Sensors: Attacks and Defenses," in *Proceedings of the 23rd Annual Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2016.

[26] A. Das, N. Borisov, E. Chou, and M. H. Mughees, "Smartphone Fingerprinting Via Motion Sensors: Analyzing Feasibility at Large-Scale and Studying Real Usage Patterns," *CoRR*, vol. abs/1605.08763, 2016. [Online]. Available: http://arxiv.org/abs/1605.08763

[27] "Apple dictates the ranking of top 10 MEMS manufacturers in 2014." [Online]. Available: https://technology.ihs.com/527700/apple-dictates-the-ranking-of-top-10-mems-manufacturers-in-2014 (Date last accessed 05-June-2016).

[28] "MEMS microphone market." [Online]. Available: http://www.digikey.com/supply-chain-hq/us/en/articles/semiconductors/mems-microphone-market-revenues-soar-42-in-2012/1497 (Date last accessed 05-June-2016).

[29] "MEMS microphone shipments to climb 30 percentage in 2013." [Online]. Available: http://electroiq.com/blog/2013/02/mems-microphone-shipments-to-climb-30-percent-this-year/ (Date last accessed 05-June-2016).

[30] "MEMS Microphone Model." [Online]. Available: http://www.comsol.com/blogs/mems-microphone-model-presented-asa-166-san-francisco/ (Date last accessed 05-June-2016).

[31] "How MEMS Microphones Fucntion." [Online]. Available: http://www.eeherald.com/section/design-guide/mems-microphone.html (Date last accessed 05-June-2016).

[32] J. Chang and Y. Peng, "Speaker, yoke thereof and method for manufacturing yoke," Jan 2012, US Patent 8,094,867.http://www.google.com/patents/US8094867.

[33] I. Shahosseini, E. Lefeuvre, M. Woytasik, J. Moulin, X. Leroux, S. Edmond, E. Dufour-Gergam, A. Bosseboeuf, G. Lemarquand, and V. Lemarquand, "Towards high fidelity high efficiency MEMS microspeakers," in *IEEE Sensors*, 2010, pp. 2426–2430.

[34] S.-S. Je, F. Rivas, R. Diaz, J. Kwon, J. Kim, B. Bakkaloglu, S. Kiaei, and J. Chae, "A Compact and Low-Cost MEMS Loudspeaker for Digital Hearing Aids," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 3, no. 5, pp. 348–358, 2009.

[35] M. Cheng, W. Huang, and S. R. Huang, "A silicon microspeaker for hearing instruments," *J. of Micromechanics and Microengineering*, vol. 14, no. 7, pp. 859–866, Jul 2004.

[36] "STMicroelectronics." [Online]. Available: http://www.st.com/web/en/home.html (Date last accessed 05-June-2016).

[37] "Invensense." [Online]. Available: http://www.invensense.com/ (Date last accessed 05-June-2016).

[38] "Research and Markets: Global MEMS Market 2015-2019." [Online]. Available: http://www.businesswire.com/news/home/20150216005540/en/Research-Markets-Global-MEMS-Market-2015-2019 (Date last accessed 05-June-2016).

[39] "iPhone 4 Teardown." [Online]. Available: https://www.ifixit.com/Teardown/iPhone+4+Teardown/3130 (Date last accessed 05-June-2016).

[40] "iPhone 5 Teardown." [Online]. Available: https://www.ifixit.com/Teardown/iPhone+5+Teardown/10525 (Date last accessed 05-June-2016).

[41] "iPhone 6 Teardown." [Online]. Available: https://www.ifixit.com/Teardown/iPhone+6+Teardown/29213 (Date last accessed 05-June-2016).

[42] "Inside the Samsung Galaxy SIII." [Online]. Available: http://www.chipworks.com/en/technical-competitive-analysis/resources/blog/inside-the-samsung-galaxy-siii/ (Date last accessed 05-June-2016).

[43] "Inside the Samsung Galaxy S4." [Online]. Available: http://www.chipworks.com/en/technical-competitive-analysis/resources/blog/inside-the-samsung-galaxy-s4/ (Date last accessed 05-June-2016).

[44] "Nexus 4 Teardown." [Online]. Available: https://www.ifixit.com/Teardown/Nexus+4+Teardown/11781 (Date last accessed 05-June-2016).

[45] "Nexus 5 Teardown." [Online]. Available: https://www.ifixit.com/Teardown/Nexus+5+Teardown/19016 (Date last accessed 05-June-2016).

[46] "MEMS-based accelerometers." [Online]. Available: http://www.wikid.eu/index.php/MEMS-based_accelerometers (Date last accessed 05-June-2016).

[47] STMicroelectronics, "Everything about STMicroelectronics 3-axis digital MEMS gyroscopes." [Online]. Available: http://www.st.com/web/en/resource/technical/document/technical_article/DM00034730.pdf (Date last accessed 05-June-2016).

[48] "MEMS gyroscopes." [Online]. Available: http://www.findmems.com/wikimems-learn/introduction-to-mems-gyroscopes (Date last accessed 05-June-2016).

[49] A. Ross and A. Jain, "Information fusion in biometrics," *Pattern Recognition Letters*, vol. 24, no. 13, pp. 2115 – 2125, 2003.

[50] S. COLE and S. Cole, *Suspect Identities: A History of Fingerprinting and Criminal Identification*. Harvard University Press, 2009.

[51] L. Langley, "Specific emitter identification (SEI) and classical parameter fusion technology," in *Proceedings of the IEEE WESCON*, 1993, pp. 377–381.

[52] M. Riezenman, "Cellular security: better, but foes still lurk," *IEEE Spectrum*, vol. 37, no. 6, pp. 39–42, 2000.

[53] Z. Li, W. Xu, R. Miller, and W. Trappe, "Securing Wireless Systems via Lower Layer Enforcements," in *Proceedings of the 5th ACM Workshop on Wireless Security (WiSe)*, 2006, pp. 33–42.

[54] N. T. Nguyen, G. Zheng, Z. Han, and R. Zheng, "Device fingerprinting to enhance wireless security using nonparametric Bayesian method," in *Proceedings of the 30th Annual IEEE International Conference on Computer Communications (INFOCOM)*, 2011, pp. 1404–1412.

[55] N. Patwari and S. K. Kasera, "Robust Location Distinction Using Temporal Link Signatures," in *Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2007, pp. 111–122.

[56] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, "Wireless Device Identification with Radiometric Signatures," in *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2008, pp. 116–127.

[57] R. M. Gerdes, T. E. Daniels, M. Mina, and S. F. Russell, "Device identification via analog signal fingerprinting: A matched filter approach," in *Proceedings of the 13th Network and Distributed System Security Symposium (NDSS)*, 2006.

[58] S. Moon, P. Skelly, and D. Towsley, "Estimation and removal of clock skew from network delay measurements," in *Proceedings of the 18th Annual IEEE International Conference on Computer Communications (INFOCOM)*, vol. 1, 1999, pp. 227–234.

[59] T. Kohno:2005, A. Broido, and K. C. Claffy, "Remote Physical Device Fingerprinting," *IEEE Trans. Dependable Secur. Comput.*, vol. 2, no. 2, pp. 93–108, 2005.

[60] L. C. C. Desmond, C. C. Yuan, T. C. Pheng, and R. S. Lee, "Identifying Unique Devices Through Wireless Fingerprinting," in *Proceedings of the First ACM Conference on Wireless Network Security (WiSec)*, 2008, pp. 46–55.

[61] J. Franklin, D. McCoy, P. Tabriz, V. Neagoe, J. Van Randwyk, and D. Sicker, "Passive Data Link Layer 802.11 Wireless Device Driver Fingerprinting," in *Proceedings of the 15th Conference on USENIX Security Symposium*, 2006.

[62] F. Guo and T. cker Chiueh, "Sequence Number-Based MAC Address Spoof Detection," in *Proceedings of 8th International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2005.

[63] G. Lyon, "Nmap: a free network mapping and security scanning tool." [Online]. Available: http://nmap.org/ (Date last accessed 05-June-2016).

[64] F. Yarochkin, M. Kydyraliev, and O. Arkin, "Xprobe project." [Online]. Available: http://ofirarkin.wordpress.com/xprobe/ (Date last accessed 05-June-2016).

[65] K. Mowery, D. Bogenreif, S. Yilek, and H. Shacham, "Fingerprinting Information in JavaScript Implementations," in *Proceedings of IEEE Web 2.0 Security & Privacy Workshop (W2SP)*, 2011.

[66] L. Olejnik, C. Castelluccia, and A. Janc, "Why Johnny Can't Browse in Peace: On the Uniqueness of Web Browsing History Patterns," in *5th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs)*, 2012.

[67] N. Nikiforakis, W. Joosen, and B. Livshits, "PriVaricator: Deceiving Fingerprinters with Little White Lies," in *Proceedings of the 24th International Conference on World Wide Web (WWW)*, 2015, pp. 820–830.

[68] "Apple places kill date on apps that use 'UDID' device identifiers." [Online]. Available: http://www.zdnet.com/article/apple-places-kill-date-on-apps-that-use-udid-device-identifiers/ (Date last accessed 05-June-2016).

[69] "Android TelephonyManager." [Online]. Available: http://developer.android.com/reference/android/telephony/TelephonyManager.html#getDeviceId() (Date last accessed 05-June-2016).

[70] P. Laperdrix, W. Rudametkin, and B. Baudry, "Beauty and the Beast: Diverting modern web browsers to build unique browser fingerprints," in *Proceedings of the 37th IEEE Symposium on Security and Privacy (S&P)* , 2016.

[71] A. Kurtz, H. Gascon, T. Becker, K. Rieck, and F. Freiling, "Fingerprinting Mobile Devices Using Personalized Configurations," *Proceedings on Privacy Enhancing Technologies*, no. 1, 2016.

[72] W. B. Clarkson, "Breaking assumptions: Distinguishing between seemingly identical items using cheap sensors," Ph.D. dissertation, Princeton, NJ, USA, 2012.

[73] H. Bojinov, Y. Michalevsky, G. Nakibly, and D. Boneh, "Mobile Device Identification via Sensor Fingerprinting," *CoRR*, vol. abs/1408.1416, 2014, http://arxiv.org/abs/1408.1416.

[74] Z. Zhou, W. Diao, X. Liu, and K. Zhang, "Acoustic Fingerprinting Revisited: Generate Stable Device ID Stealthily with Inaudible Sound," in *Proceedings of the 21st ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2014, pp. 429–440.

[75] P. Cano, E. Batlle, T. Kalker, and J. Haitsma, "A Review of Audio Fingerprinting," *J. VLSI Signal Process. Syst.*, vol. 41, no. 3, pp. 271–284, Nov 2005.

[76] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[77] G. Guo and S. Li, "Content-based audio classification and retrieval by support vector machines," *IEEE Transactions on Neural Networks*, vol. 14, no. 1, pp. 209–215, Jan 2003.

[78] J. Campbell, J.P., "Speaker recognition: a tutorial," *Proceedings of the IEEE*, vol. 85, no. 9, pp. 1437–1462, Sep 1997.

[79] F. Bimbot, J.-F. Bonastre, C. Fredouille, G. Gravier, I. Magrin-Chagnolleau, S. Meignier, T. Merlin, J. Ortega-Garcia, D. Petrovska-Delacretaz, and D. A. Reynolds, "A Tutorial on Text-Independent Speaker Verification," *EURASIP Journal on Advances in Signal Processing*, vol. 4, pp. 430–451, 2004.

[80] M. Mckinney and J. Breebaart, "Features for Audio and Music Classification," in *Proceedings of the 2003 International Symposium on Music Information Retrieval*, 2003, pp. 151–158.

[81] T. Li, M. Ogihara, and Q. Li, "A Comparative Study on Content-based Music Genre Classification," in *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, ser. SIGIR '03, 2003, pp. 282–289.

[82] J. Haitsma and T. Kalker, "A Highly Robust Audio Fingerprinting System," in *Proceedings of the 2002 International Symposium on Music Information Retrieval*, 2002, pp. 107–115.

[83] S. Dey, N. Roy, W. Xu, R. R. Choudhury, and S. Nelakuditi, "AccelPrint: Imperfections of Accelerometers Make Smartphones Trackable," in *Proceedings of the 21st Annual Network and Distributed System Security Symposium (NDSS)*, 2014.

[84] Y. Michalevsky, D. Boneh, and G. Nakibly, "Gyrophone: Recognizing Speech from Gyroscope Signals," in *Proceedings of the 23rd USENIX Conference on Security Symposium*, 2014, pp. 1053–1067.

[85] M. Frank, B. Dong, A. Porter Felt, and D. Song, "Mining Permission Request Patterns from Android and Facebook Applications," in *Proceedings of the 2012 IEEE 12th International Conference on Data Mining (ICDM)*, 2012, pp. 870–875.

[86] P. Kelley, S. Consolvo, L. Cranor, J. Jung, N. Sadeh, and D. Wetherall, "A Conundrum of Permissions: Installing Applications on an Android Smartphone," in *Financial Cryptography and Data Security*, 2012, pp. 68–79.

[87] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, "Android Permissions Demystified," in *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS)*, 2011, pp. 627–638.

[88] A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner, "Android Permissions: User Attention, Comprehension, and Behavior," in *Proceedings of the 8th Symposium on Usable Privacy and Security*, ser. SOUPS '12, 2012, pp. 3:1–3:14.

[89] "My Talking Tom." [Online]. Available: https://play.google.com/store/apps/details?id=com.outfit7.mytalkingtomfree (Date last accessed 05-June-2016).

[90] "WAVE PCM soundfile format." [Online]. Available: http://soundfile.sapp.org/doc/WaveFormat/ (Date last accessed 05-June-2016).

[91] "MIRtoolbox." [Online]. Available: https://www.jyu.fi/hum/laitokset/musiikki/en/research/coe/materials/mirtoolbox (Date last accessed 05-June-2016).

[92] "Netlab: Algorithms for Pattern Recognition." [Online]. Available: http://www1.aston.ac.uk/eas/research/groups/ncrg/resources/netlab/book/ (Date last accessed 05-June-2016).

[93] "Audacity is free, open source, cross-platform software for recording and editing sounds." [Online]. Available: http://audacity.sourceforge.net/ (Date last accessed 05-June-2016).

[94] "Hertz, the WAV recorder." [Online]. Available: https://play.google.com/store/apps/details?id=uk.ac.cam.cl.dtg.android.audionetworking.hertz (Date last accessed 05-June-2016).

[95] M. A. Bartsch and G. H. Wakefield, "Audio Thumbnailing of Popular Music Using Chroma-based Representations," *IEEE Transactions on Multimedia*, vol. 7, no. 1, pp. 96–104, Feb 2005.

[96] C. Chen, *Signal Processing Handbook*, ser. Electrical and Computer Engineering, 1988.

[97] F. Gouyon, F. Pachet, and O. Delerue, "On the Use of Zero-Crossing Rate for an Application of Classification of Percussive Sounds," in *Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-00*, 2000.

[98] H. Misra, S. Ikbal, H. Bourlard, and H. Hermansky, "Spectral entropy based feature for robust ASR," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, 2004, pp. 193–196.

[99] K. Jensen, *Timbre Models of Musical Sounds*, ser. PhD Dissertation. University of Copenhagen, 1999.

[100] G. Peeters, "A large set of audio features for sound description (similarity and classification) in the CUIDADO project," Icram, Tech. Rep., 2004. [Online]. Available: http://recherche.ircam.fr/equipes/analyse-synthese/peeters/ARTICLES/Peeters_2003_cuidadoaudiofeatures.pdf

[101] P. N. Juslin, "Cue utilization in communication of emotion in music performance : Relating performance to perception," *Journal of Experimental Psychology: Human Perception and Performance*, vol. 26, no. 6, pp. 1797–1813, 2000.

[102] J. Johnston, "Transform coding of audio signals using perceptual noise criteria," *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 2, pp. 314–323, Feb 1988.

[103] B. Logan, "Mel Frequency Cepstral Coefficients for Music Modeling," in *In International Symposium on Music Information Retrieval*, 2000.

[104] T. Fujishima, "Realtime chord recognition of musical sound: A system using common Lisp Music," in *International Computer Music Conference (ICMA)*, 1999, pp. 464–467.

[105] C. Harte, M. Sandler, and M. Gasser, "Detecting Harmonic Change in Musical Audio," in *Proceedings of the 1st ACM Workshop on Audio and Music Computing Multimedia*, ser. AMCMM '06, 2006, pp. 21–26.

[106] R. Duda, P. Hart, and D. Stork, *Pattern classification*. Wiley, 2001.

[107] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *J. of the Royal Statistical Society. Series B*, vol. 39, no. 1, pp. 1–38, 1977.

[108] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker verification using adapted gaussian mixture models," *Digital Signal Processing*, vol. 10, no. 1-3, pp. 19–41, 2000.

[109] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing and Management*, vol. 45, no. 4, pp. 427–437, 2009.

[110] I. Guyon and A. Elisseeff, "An Introduction to Variable and Feature Selection," *Journal of Machine Learning Research*, vol. 3, no. 26, pp. 1157–1182, Mar 2003.

[111] Y. Yang and J. O. Pedersen, "A Comparative Study on Feature Selection in Text Categorization," in *Proceedings of the Fourteenth International Conference on Machine Learning*, ser. ICML '97, 1997, pp. 412–420.

[112] "Audio 4 Smartphones – Wolfson Microelectronics." [Online]. Available: http://www.wolfsonmicro.com/documents/uploads/misc/en/Audio4Smartphones.pdf (Date last accessed 05-June-2016).

[113] "5 of the best DACs." [Online]. Available: http://www.stuff.tv/music/5-best-dacs-how-make-your-digital-music-sound-amazing/feature (Date last accessed 05-June-2016).

[114] "Ambient Sound Effects." [Online]. Available: http://www.pacdv.com/sounds/ambience_sounds.html (Date last accessed 05-June-2016).

[115] "SOUNDJAY-Ambient Sound Effects." [Online]. Available: http://www.soundjay.com/ambient-sounds.html (Date last accessed 05-June-2016).

[116] "DeviceOrientation Event Specification." [Online]. Available: https://www.w3.org/TR/orientation-event/ (Date last accessed 05-June-2016).

[117] "Percentage of all global web pages served to mobile phones." [Online]. Available: http://www.statista.com/statistics/241462/global-mobile-phone-website-traffic-share/ (Date last accessed 05-June-2016).

[118] "Top Mobile Browsers from Jan 2014 toJan 2015." [Online]. Available: http://gs.statcounter.com/#mobile_browser-ww-monthly-201401-201501 (Date last accessed 05-June-2016).

[119] "Browser Trends September 2014: Chrome Is the Top Mobile Browser." [Online]. Available: http://www.sitepoint.com/browser-trends-september-2014-chrome-top-mobile-browser/ (Date last accessed 05-June-2016).

[120] "Android Sensors Overview." [Online]. Available: http://developer.android.com/guide/topics/sensors/sensors_overview.html (Date last accessed 05-June-2016).

[121] "Corona SDK API reference." [Online]. Available: http://docs.coronalabs.com/api/library/system/setAccelerometerInterval.html (Date last accessed 05-June-2016).

[122] S. McKinley and M. Levine, "Cubic Spline Interpolation," *College of the Redwoods*, vol. 45, no. 1, pp. 1049–1060, 1998.

[123] W. A. Sethares, "Adaptive Tunings," in *Tuning, Timbre, Spectrum, Scale*. Springer, 1998, pp. 147–164.

[124] O. Lartillot, "MIRtoolbox 1.5 — Users Manual." [Online]. Available: https://www.jyu.fi/hum/laitokset/musiikki/en/research/coe/materials/mirtoolbox/MIRtoolbox1.5Guide (Date last accessed 05-June-2016).

[125] "Supervised Learning (Machine Learning) Workflow and Algorithms." [Online]. Available: http://www.mathworks.com/help/stats/supervised-learning-machine-learning-workflow-and-algorithms.html (Date last accessed 05-June-2016).

[126] "LibXtract Documentation." [Online]. Available: http://libxtract.sourceforge.net/ (Date last accessed 05-June-2016).

[127] A. Pocock and G. Brown, "FEAST," 2014. [Online]. Available: http://mloss.org/software/view/386/

[128] G. Brown, A. Pocock, M.-J. Zhao, and M. Luján, "Conditional Likelihood Maximisation: A Unifying Framework for Information Theoretic Feature Selection," *Machine Learning Research*, vol. 13, pp. 27–66, 2012.

[129] "Amazon Mechanical Turk." [Online]. Available: https://www.mturk.com/mturk/welcome (Date last accessed 05-June-2016).

[130] A. Bellet, A. Habrard, and M. Sebban, "A Survey on Metric Learning for Feature Vectors and Structured Data," *CoRR*, vol. abs/1306.6709, 2013, http://arxiv.org/abs/1306.6709.

[131] M. Köstinger, M. Hirzer, P. Wohlhart, P. M. Roth, and H. Bischof, "Large scale metric learning from equivalence constraints," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 2012, pp. 2288–2295.

[132] K. Weinberger and L. Saul, "Distance Metric Learning for Large Margin Nearest Neighbor Classification," *The Journal of Machine Learning Research*, vol. 10, pp. 207–244, 2009.

[133] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, "Information-theoretic Metric Learning," in *Proceedings of the 24th International Conference on Machine Learning (ICML)*. ACM, 2007, pp. 209–216.

[134] M. Guillaumin, J. Verbeek, and C. Schmid, "Is that you? Metric learning approaches for face identification," in *Proceedings of the 12th International Conference on Computer Vision (ICCV)*. IEEE, 2009, pp. 498–505.

[135] "Fit probability distribution object to data." [Online]. Available: http://www.mathworks.com/help/stats/fitdist.html (Date last accessed 05-June-2016).

[136] H. Akaike, *Information Theory and an Extension of the Maximum Likelihood Principle*. Springer New York, 1998, pp. 199–213.

[137] "How Long Do Users Stay on Web Pages?" [Online]. Available: https://www.nngroup.com/articles/how-long-do-users-stay-on-web-pages/ (Date last accessed 05-June-2016).

[138] "What You Think You Know About the Web Is Wrong." [Online]. Available: http://time.com/12933/what-you-think-you-know-about-the-web-is-wrong/ (Date last accessed 05-June-2016).

[139] "Refining a k-Nearest-Neighbor classification." [Online]. Available: https://www3.nd.edu/~steve/computing_with_data/17_Refining_kNN/refining_knn.html (Date last accessed 05-June-2016).

[140] "Selenium Web Driver." [Online]. Available: http://www.seleniumhq.org/projects/webdriver (Date last accessed 05-June-2016).

[141] "Open source V8 JavaScript engine." [Online]. Available: https://github.com/v8/v8/wiki (Date last accessed 05-June-2016).

[142] C. Curtsinger, B. Livshits, B. Zorn, and C. Seifert, "ZOZZLE: Fast and Precise In-browser JavaScript Malware Detection," in *Proceedings of the 20th USENIX Conference on Security (SEC)*. USENIX Association, 2011.

[143] "ECMAScript parsing infrastructure for multipurpose analysis." [Online]. Available: http://esprima.org (Date last accessed 05-June-2016).

[144] L. van der Maaten, "Matlab Toolbox for Dimensionality Reduction." [Online]. Available: http://lvdmaaten.github.io/drtoolbox/ (Date last accessed 05-June-2016).

[145] D. K. Agrafiotis, "Stochastic Proximity Embedding," *Journal of computational chemistry*, vol. 24, no. 10, pp. 1215–1221, 2003.

[146] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[147] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987.

[148] "Stanford Javascript Crypto Library." [Online]. Available: https://github.com/bitwiseshiftleft/sjcl/blob/master/sjcl.js (Date last accessed 05-June-2016).

[149] "Parallax.js." [Online]. Available: https://github.com/wagerfield/parallax (Date last accessed 05-June-2016).

[150] "jGestures." [Online]. Available: https://jgestures.codeplex.com/ (Date last accessed 05-June-2016).

[151] "MotionCAPTCHA." [Online]. Available: https://github.com/josscrowcroft/MotionCAPTCHA (Date last accessed 05-June-2016).

[152] "Trapezoid Rule." [Online]. Available: http://www.mathwords.com/t/trapezoid_rule.htm (Date last accessed 05-June-2016).

[153] C. Dwork, "Differential Privacy," in *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming (ICALP)*. Springer Verlag, 2006, pp. 1–12.

[154] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating Noise to Sensitivity in Private Data Analysis," in *Proceedings of the 3rd Conference on Theory of Cryptography (TCC)*, 2006, pp. 265–284.

[155] "Wearables vs. Smartphone Apps: Which Are Better to Count Steps?" [Online]. Available: http://www.livescience.com/49756-smartphone-apps-wearables-step-counts.html (Date last accessed 05-June-2016).

APPENDIX A

FEATURE DISTRIBUTIONS FOR MOTION SENSOR DATA

We collected motion sensor data from a total of 610 devices. This large dataset enabled us to derive a distribution for each feature. Figure A.1 shows the distributions for the top 12 features selected based on *JMI* criterion [127].

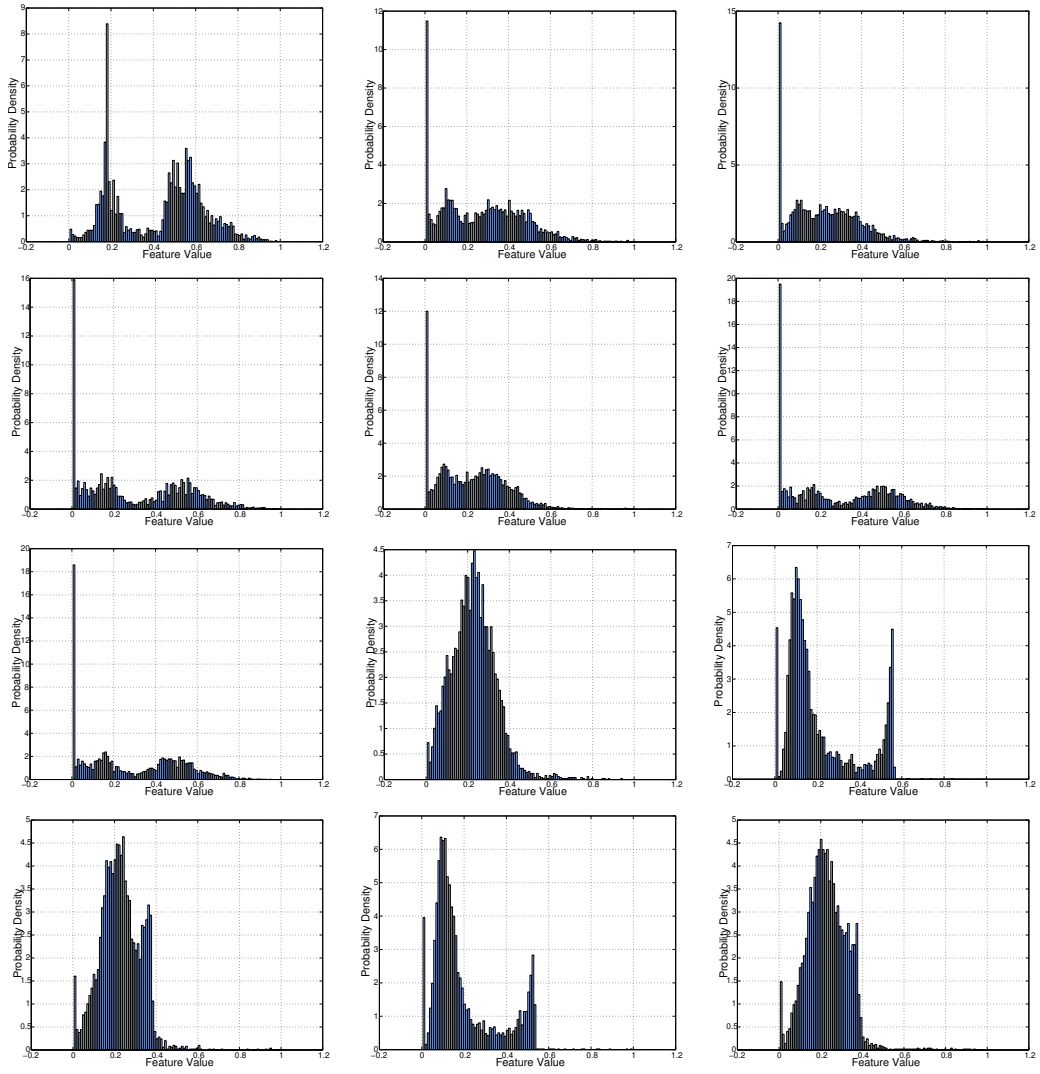


Figure A.1: Distributions for the top 12 original features.

Similarly, Figure A.2 shows the distributions for the top 12 (out of 100) features after LDML transformation is performed on the original dataset. Compared to the original untransformed features we can see that the top features under LDML have a higher degree of variation and hence higher entropy.

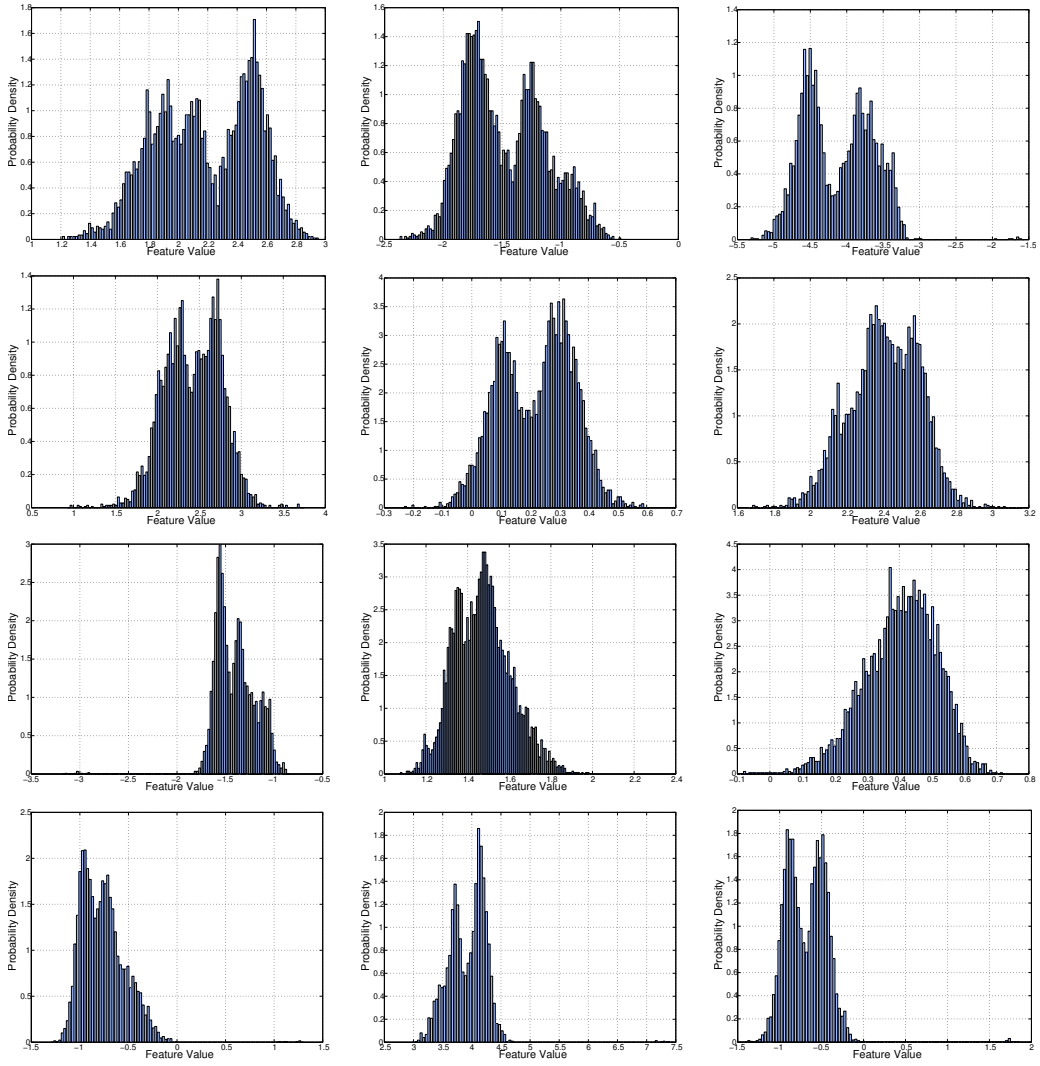


Figure A.2: Distributions for the top 12 features selected based on *JMI* criterion after LDML transformation is performed.

APPENDIX B

DEVICE MODELS IN OUR DATASET

We collected sensor data from a total of 610 devices which covered 108 different make and models of devices. Figure B.1 shows the distribution of the different device models. We can see that different models of iPhones comprise around 47% of all devices.

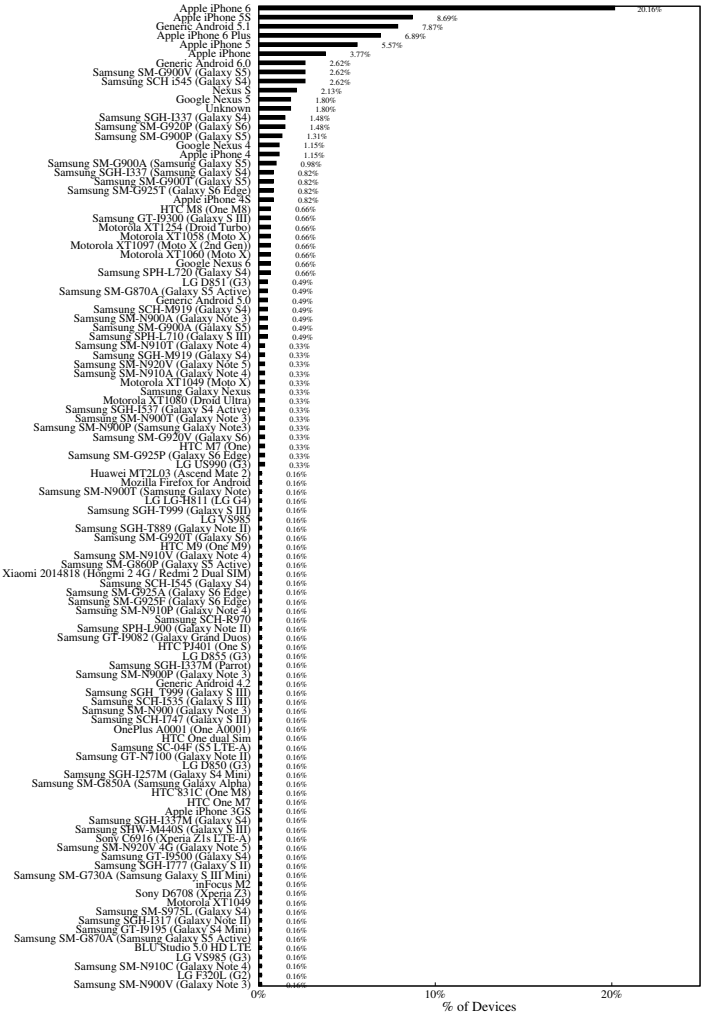


Figure B.1: Distribution of the different make and model of smartphones that provided sensor data for our study.

APPENDIX C

DEVICE MODELS IN OUR USER STUDY

In total 407 users participated in our user study covering 144 different device models. Figure C.1 shows the distribution of the different device models that participated. 41% of the devices were different models of iPhones.

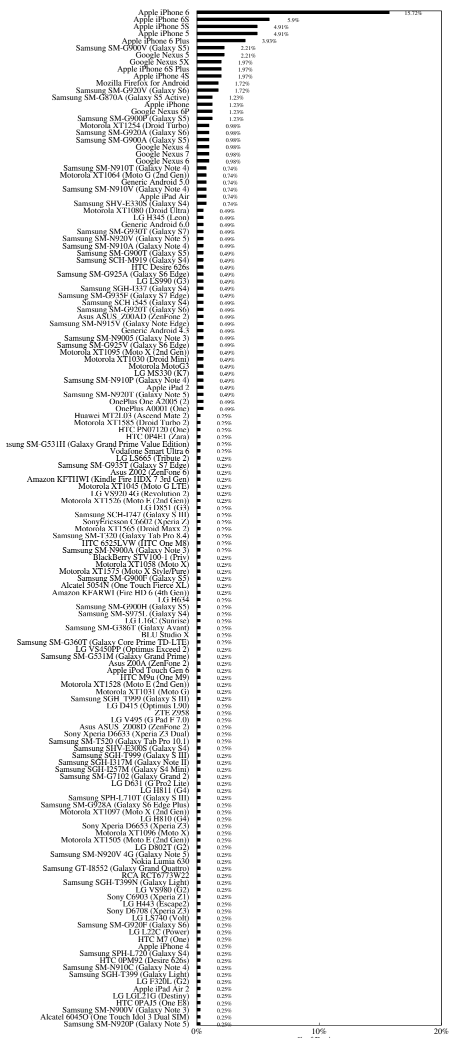


Figure C.1: Distribution of the different make and model of smartphones that participated in our user study.