SECURITY AND PRIVACY ON SMARTPHONES:
A SENSING APPROACH

by
Ming Li
A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Doctor of Philosophy (Computer Science).

Golden, Colorado

Date __________________________

Signed: _________________________

Ming Li

Signed: _________________________

Dr. Dejun Yang
Thesis Advisor

Golden, Colorado

Date __________________________

Signed: _________________________

Dr. Tracy Camp
Professor and Head
Department of Computer Science
ABSTRACT

The technological advancements have made smartphones an indispensable component of our daily lives. Security and privacy (S&P) on smartphones has thus become an important research topic. Since there are no universally applicable solutions to solve S&P issues on smartphones, we conduct our research in a case study manner, with a focus on smartphone sensors. We propose Spy-Phone to show how smartphones get eavesdropped by motion sensors, Ultra-Unlock to authenticate users with gestures in the air, and MoVo to protect voice authentication systems against spoofing attacks.

In detail, the Spy-Phone system turns smartphones into spy bugs by performing Man-in-the-Phone attack. Such an attack is based on the fact that motion sensors (accelerometers and gyroscopes) can measure audio signals, though at a much lower sampling rate. It is a big threat to smartphone users since the phone’s operating system grants applications permissions to motion sensors automatically.

Ultra-Unlock uses the microphones and speakers in smartphones to send ultrasound signals and catch users’ finger movements, then utilizes these user-specific movements to unlock the phone. It is a great alternative to the password/fingerprint authentication when users’ fingers are dirty or wet and to the face authentication when users wear masks or goggles.

MoVo is a spoof-proof voice authentication system that not only authenticates users by their voices but also differentiates live people and electronic devices. In other words, attackers are unable to unlock the phone by replay attack (attackers record the victim’s voice in person or online, then replay the recording and access the victim’s devices illegally). The idea is to utilize the self demodulation effect and acoustic attenuation effect that occurred when sound signals transmit through human bodies. Motion sensors are used to catch such signals.
# TABLE OF CONTENTS

ABSTRACT ........................................................................................................ iii
LIST OF FIGURES ................................................................. vii
LIST OF TABLES ........................................................................ x
ACKNOWLEDGMENTS .................................................................. xi

CHAPTER 1 INTRODUCTION .............................................. 1
1.1 Security and Privacy on Smartphones ........................................ 1
1.2 Smartphone Sensors: Causes or Solutions .................................... 2
1.3 Objectives and Contributions ......................................................... 4

CHAPTER 2 SPY-PHONE: EAVESDROPPING ON SMARTPHONE SPEAKERS
WITH MOTION SENSORS .................................................. 7
2.1 Introduction ............................................................... 7
2.2 Background .................................................................. 12
   2.2.1 Voice Acoustics .......................................................... 12
   2.2.2 Compressed Sensing .................................................. 13
   2.2.3 Smartphone Hardware .............................................. 16
2.3 Threat Model in the Spy-Phone System ...................................... 17
2.4 Attack Design .................................................................. 19
   2.4.1 Training Dataset .......................................................... 21
   2.4.2 Dictionary Learning ...................................................... 23
   2.4.3 Motion Data Preprocessing ........................................ 26
   2.4.4 Signal Reconstruction .................................................. 28
   2.4.5 Bi-LSTM Learning ...................................................... 29
2.5 Feasibility Experiments .......................................................... 30
   2.5.1 Speech Content Learning (Digits) .................................... 32
   2.5.2 Speech Content Learning (Commands) .............................. 32
   2.5.3 Speaker Gender Classification and Speaker Identification ......... 33
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5.4 User Activity Classification (Sound Type)</td>
<td>34</td>
</tr>
<tr>
<td>2.6 Impact Evaluation</td>
<td>35</td>
</tr>
<tr>
<td>2.6.1 Impact of Training Data Size</td>
<td>36</td>
</tr>
<tr>
<td>2.6.2 Impact of Downsampling Rate</td>
<td>36</td>
</tr>
<tr>
<td>2.6.3 Impact of Training Device</td>
<td>37</td>
</tr>
<tr>
<td>2.6.4 Impact of Sound Volume</td>
<td>37</td>
</tr>
<tr>
<td>2.6.5 Impact of Surrounding Environment</td>
<td>38</td>
</tr>
<tr>
<td>2.6.6 Impact of User Mobility</td>
<td>40</td>
</tr>
<tr>
<td>2.7 Defenses</td>
<td>43</td>
</tr>
<tr>
<td>2.7.1 Hardware-based Defenses</td>
<td>43</td>
</tr>
<tr>
<td>2.7.2 Software-based Defenses</td>
<td>43</td>
</tr>
<tr>
<td>2.8 Related Work</td>
<td>44</td>
</tr>
<tr>
<td>2.8.1 Inferring Information From Smartphone Motion Sensors</td>
<td>44</td>
</tr>
<tr>
<td>2.8.2 Eavesdropping Sound Signals By Non-Acoustic Sensors</td>
<td>45</td>
</tr>
<tr>
<td>2.9 Conclusion</td>
<td>46</td>
</tr>
<tr>
<td>CHAPTER 3 ULTRA-UNLOCK: SMARTPHONE AUTHENTICATION USING GESTURES IN THE AIR</td>
<td></td>
</tr>
<tr>
<td>3.1 Introduction</td>
<td>47</td>
</tr>
<tr>
<td>3.2 Related Works</td>
<td>48</td>
</tr>
<tr>
<td>3.3 Background</td>
<td>50</td>
</tr>
<tr>
<td>3.3.1 I/Q components and I/Q modulation</td>
<td>50</td>
</tr>
<tr>
<td>3.3.2 Hand Movements and I/Q Data</td>
<td>52</td>
</tr>
<tr>
<td>3.4 System Design</td>
<td>54</td>
</tr>
<tr>
<td>3.5 Experiment Results</td>
<td>57</td>
</tr>
<tr>
<td>3.6 Conclusion</td>
<td>63</td>
</tr>
<tr>
<td>CHAPTER 4 MOVO: A SPOOF-PROOF VOICE AUTHENTICATION SYSTEM FOR SMARTPHONES</td>
<td></td>
</tr>
<tr>
<td>4.1 Introduction</td>
<td>64</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>4.1.1 The Popularity of Voice Authentication on Smartphones</td>
<td>64</td>
</tr>
<tr>
<td>4.1.2 Voice Spoofing Attacks</td>
<td>65</td>
</tr>
<tr>
<td>4.1.3 Liveness Detection</td>
<td>68</td>
</tr>
<tr>
<td>4.1.4 Overview of MoVo</td>
<td>73</td>
</tr>
<tr>
<td>4.2 Background</td>
<td>75</td>
</tr>
<tr>
<td>4.2.1 Voice Acoustics</td>
<td>75</td>
</tr>
<tr>
<td>4.2.2 Self Demodulation</td>
<td>76</td>
</tr>
<tr>
<td>4.2.3 Acoustic Attenuation</td>
<td>77</td>
</tr>
<tr>
<td>4.3 Proof-of-Concept</td>
<td>79</td>
</tr>
<tr>
<td>4.4 Attack Model</td>
<td>85</td>
</tr>
<tr>
<td>4.5 System Design</td>
<td>86</td>
</tr>
<tr>
<td>4.5.1 System Overview</td>
<td>86</td>
</tr>
<tr>
<td>4.5.2 Syllable Separation</td>
<td>88</td>
</tr>
<tr>
<td>4.5.3 Preprocessing and Segmentation</td>
<td>89</td>
</tr>
<tr>
<td>4.5.4 The LSTM Network</td>
<td>91</td>
</tr>
<tr>
<td>4.6 Majority Voting</td>
<td>92</td>
</tr>
<tr>
<td>4.7 Implementation and Evaluation</td>
<td>93</td>
</tr>
<tr>
<td>4.8 Conclusion</td>
<td>99</td>
</tr>
<tr>
<td>CHAPTER 5 CONCLUSION</td>
<td>101</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>102</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>1.1</td>
<td>Real Readings of 18 Sensors on a Google Nexus 6P Device.</td>
</tr>
<tr>
<td>2.1</td>
<td>Example of an Attacking Scenario</td>
</tr>
<tr>
<td>2.2</td>
<td>Man-in-the-Phone Attack Should be Speaker-Independent.</td>
</tr>
<tr>
<td>2.3</td>
<td>Mathematical Model of a Typical Compressed Sensing System.</td>
</tr>
<tr>
<td>2.4</td>
<td>Structure of a Desktop Loudspeaker Versus a MEMS Speaker.</td>
</tr>
<tr>
<td>2.5</td>
<td>The Man-in-the-Phone Attack Workflow in the Spy-Phone System.</td>
</tr>
<tr>
<td>2.6</td>
<td>Spy-Phone Data Collection Application</td>
</tr>
<tr>
<td>2.7</td>
<td>Example of Learned Dictionary Atoms. For each digit class, two different</td>
</tr>
<tr>
<td></td>
<td>atoms are shown.</td>
</tr>
<tr>
<td>2.8</td>
<td>The Magnitude and Phase Response of the FIR Highpass filter.</td>
</tr>
<tr>
<td>2.9</td>
<td>Preprocessing of Motion Data.</td>
</tr>
<tr>
<td>2.10</td>
<td>The Bidirectional Long Short-Term Memory Network (Bi-LSTM Network).</td>
</tr>
<tr>
<td>2.11</td>
<td>The Confusion Matrix of the Digit Classification Result.</td>
</tr>
<tr>
<td>2.12</td>
<td>The Confusion Matrix of the Speaker Gender Classification Result.</td>
</tr>
<tr>
<td>2.13</td>
<td>The Confusion Matrix of the Sound Type Classification Result.</td>
</tr>
<tr>
<td>2.14</td>
<td>Impact of Training Data Size.</td>
</tr>
<tr>
<td>2.15</td>
<td>Impact of Downsampling Rate.</td>
</tr>
<tr>
<td>2.16</td>
<td>Impact of Sound Volume.</td>
</tr>
<tr>
<td>2.17</td>
<td>Impact of Surrounding Environment.</td>
</tr>
<tr>
<td>2.18</td>
<td>Testing Setup.</td>
</tr>
<tr>
<td>2.19</td>
<td>Impact of User Mobility.</td>
</tr>
<tr>
<td>2.20</td>
<td>Impact of Surrounding Environments.</td>
</tr>
</tbody>
</table>
Figure 2.21 Accuracy in the walking (left) and driving (right) scenario................. 42
Figure 3.1 In-phase and Quadrature Components from Different View Angles ....... 51
Figure 3.2 Sound Propagation Paths on a Smartphone.................................... 53
Figure 3.3 Android App design for Ultra-Unlock............................................. 55
Figure 3.4 Frequency Response of the Cascaded Integrator-Comb Filter............. 56
Figure 3.5 Screenshots of the UltraUnlockApp............................................... 58
Figure 3.6 I/Q Components Collected by UltraUnlockApp............................... 59
Figure 3.7 Gestures tested in Ultra-Unelock.................................................... 60
Figure 3.8 The Confusion Matrix of the Gesture Classification Result................ 60
Figure 3.9 The Confusion Matrix of the User Classification Result.................... 61
Figure 4.1 Screenshots about Voice Match on Google Nexus 6P......................... 67
Figure 4.2 Authentication Scenario of MoVo................................................... 73
Figure 4.3 Self Demodulation of Sound Signals When Transmitting Through the Human Body............................................................. 76
Figure 4.4 One user speaks “Ok Google” twice.................................................. 80
Figure 4.5 Different users both speak “Ok Google”............................................ 81
Figure 4.6 One user speaks “Ok Google” and “Hi Siri”..................................... 82
Figure 4.7 Live user speaks “Ok Google” once, then replay the recording by an electronic device......................................................... 83
Figure 4.8 MoVo Attack Model................................................................. 85
Figure 4.9 The flow of MoVo................................................................. 87
Figure 4.10 Syllable Separation .............................................................. 88
Figure 4.11 Preprocessing and Segmentation................................................. 90
Figure 4.12 Classification result without syllable separation and majority voting:
  Falsely classifying an ‘Ok Google’ sample to ‘Hi Siri’ .............................. 92

Figure 4.13 Classification Result ...................................................................... 92

Figure 4.14 Screenshots of the MoVo Data Collection App and MoVo Speaker
  Detection App ......................................................................................... 94

Figure 4.15 Attacker Settings ............................................................................ 95

Figure 4.16 Success rate of MoVo on defending against various attacks. ............ 96

Figure 4.17 Confusion Matrix of Matching Motion Data to Different Hot-Words. ...... 97

Figure 4.18 Robustness of MoVo over time ......................................................... 98

Figure 4.19 Confusion Matrix of Matching Motion Data to Different Users ............ 100
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1.1</td>
<td>General Information</td>
<td>3</td>
</tr>
<tr>
<td>Table 2.1</td>
<td>Maximum Sampling Rate of Smartphone Sensors</td>
<td>9</td>
</tr>
<tr>
<td>Table 2.2</td>
<td>Dataset Information</td>
<td>21</td>
</tr>
<tr>
<td>Table 2.3</td>
<td>Comparison with Prior Works</td>
<td>31</td>
</tr>
<tr>
<td>Table 2.4</td>
<td>Statistical Analysis of the Speaker Identification Result</td>
<td>38</td>
</tr>
<tr>
<td>Table 4.1</td>
<td>Related Work on Liveness Detection</td>
<td>69</td>
</tr>
<tr>
<td>Table 4.3</td>
<td>Statistical Analysis of the User Classification Result</td>
<td>98</td>
</tr>
</tbody>
</table>
ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest appreciation to Dr. Dejun Yang, without whom this thesis would not have been possible. Dr. Yang is not only my advisor, but he is also my mentor who gave his endless support and encouraged me to go through all the difficulties throughout my masters and doctoral programs. I would also like to thank my committee members, Dr. Tracy Camp, Dr. Michael Wakin, and Dr. Hao Zhang, not only for their time and support, but also for their valuable comments and insightful questions. Thanks should also go to all my friends at Mines. Special thanks to my research group members, Dr. Jian Lin, Yuhui Zhang, and Nan Jiang. It is a great pleasure of studying and working with them. Pursuing a Ph.D. is difficult, but they make the journey no longer lonely. Lastly, I would like to thank my family and extended family. Near or far, they have always being there for me. Words are powerless to express my gratitude.
For my love, X.
CHAPTER 1

INTRODUCTION

1.1 Security and Privacy on Smartphones

Compared to traditional feature phones which are capable of voice calls and text messages, smartphones bring many more applications including, but not limited to, email checking, web browsing, online shopping, game playing, music listening, video shooting, and GPS navigation. With such extensive capabilities, smartphones have become ubiquitous and all-pervasive. Indeed, the total number of smartphone users worldwide is over 3 billion this year – nearly 40% of the human population, according to reports issued by several market-research firms [79, 72].

The increasing of smartphone users also increases the importance of protecting the security and privacy on smartphones. Smartphones hold our important personal information such as photos and videos, SMS, email, contact list, social media accounts, etc. But existing powerful OSs and applications are not enough to protect those information [4]. For example, Google confirms an Android camera security threat where ‘hundreds of millions’ of users are affected 1. The attackers can control the Google Camera app to take photos and/or record videos through a rogue application 2 that has no permissions to do so.

Researches have done extensive study on the security and privacy problems about smartphones [4, 134, 20, 53]. They have classified the smartphone problems into four categories [134]:

- Authentication. There are mainly three ways to authenticate humans: something you know (PIN, graphical pattern, password, etc.), something you have (One-time passcode (OTP) via SMS, offline OTP using apps, paired devices, etc.), and something you are (voice, face, fingerprint, etc.). However, every method has pros and cons and no one is perfect. Researchers keep proposing new methods as surveyed in [110, 106, 66, 120, 39].

---

2https://www.checkmarx.com/blog/how-attackers-could-hijack-your-android-camera
In this thesis, we also propose a new authentication system Ultra-UnLock. We also improve the existing voice authentication system against spoofing attacks in MoVo.

- Data Protection and Privacy. Smartphones, like computers, bring the same concerns: whether or how data is shared with third parties. Researchers have studied data protection and privacy problems from different perspective [76, 11, 124]. Existing solutions are mainly designing protocols [135, 116] or utilizing cryptography [85, 40].

- Vulnerabilities. Smartphone vulnerabilities include the following: system faults/defects, insufficient management of applications, insecure wireless networks, and lack of user awareness. The hardware and operating systems on smartphones are evolving and upgrading all the time. They are inevitable to have defects at a certain stage. Developers are publishing new applications every day. Detecting malware, hacking, and other harmful codes embedded in the apps is still an open problem [49]. Even when the smartphone is secure, its connection to an insecure network can still cause S&P problems. Moreover, user awareness is another critical factor for smartphone security. A previous research [117] suggests that 35% of smartphone users do not lock their devices to prevent unauthorized persons from using them. Improving the user awareness level is urgent [55].

- Attacks. Researchers have studied various attacks on smartphones. For example, researchers propose theft detection algorithms to protect smartphones against physical attacks [17], and propose liveness detection algorithms to prevent replay attacks [139]. However, there are many other attacks unrevealed or uninvestigated. In this thesis, we propose the Man-in-the-Phone attack, which turns smartphones into spy bugs.

1.2 **Smartphone Sensors: Causes or Solutions**

There are no universally applicable solutions to solve the aforementioned security and privacy issues, so we conduct our research in a case study manner. We propose Spy-Phone to show how smartphones get eavesdropped by motion sensors, Ultra-UnLock to authenticate users with gestures in the air, and MoVo to protect voice authentication system against spoofing attacks. They have one thing in common, utilizing smartphone
Table 1.1: General Information

<table>
<thead>
<tr>
<th>System</th>
<th>Category</th>
<th>Sensors</th>
<th>Targeted Signals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spy-Phone</td>
<td>Attacks, Vulnerabilities</td>
<td>Motion Sensors</td>
<td>Acoustic Signals</td>
</tr>
<tr>
<td>Ultra-Unock</td>
<td>Authentication</td>
<td>Micophones</td>
<td>Ultrasound Signals</td>
</tr>
<tr>
<td>MoVo</td>
<td>Authentication, Attacks</td>
<td>Micophones, Motion Sensors</td>
<td>Acoustic Signals</td>
</tr>
</tbody>
</table>

sensors to achieve the goal. The categories of the three proposed systems and sensors used in each system are shown in Table 1.1. Our main focuses are attacks and authentication methods on smartphones, using motion sensors and/or microphones.

Commercial off-the-shelf (COTS) smartphones nowadays have built-in sensors that measure motions, orientations, and various environmental conditions. Figure 1.1 shows the real sensor readings on a Google Nexus 6P smartphone when running the Sensors Toolbox by ExaMobile. This smartphone has motion sensors (accelerometers, gravity sensors, gyroscopes, and rotational vector sensors), environmental sensors (barometers, photometers, and thermometers), position sensors (orientation sensors and magnetometers), microphones and cameras. Note that the motion sensors are designed to monitor the motion of a device, but in Spy-Phone and MoVo we use them to measure acoustic signals. Microphones are designed to record audible sounds, but we use them in Ultra-Unock to measure ultrasound signals.

In fact, there have been many research papers related to ‘misusing’ smartphone sensors. On the one hand, ‘misusing’ sensors introduces S&P problems. For example, Zhou et al [141] designed PatternListener, which cracks Android pattern lock using acoustic signals. It leverages speakers and microphones of the victim’s device to play and record imperceptible audio in order to achieve motion data. Zhang et al. [136] proposed DolphinAttack, which uses normal speakers to send inaudible voice commands. Since they modulated voice commands on ultrasonic carriers, popular speech recognition systems, including Siri, Google Now, and Alexa, will listen to the commands but human ears cannot detect them. Tripple et al. [113] studied how to use audio signals to control motion sensors
and proposed WALNUT. They also demonstrated how to inject fake steps into a Fitbit with a $5 speaker and how to use a malicious music file from a smartphone’s speaker to control the on-board MEMS accelerometer trusted by a local app to pilot a toy RC car.

On the other hand, ‘misusing’ sensors solves S&P problems. For example, Lee et al. [60] proposed a multi-sensor authentication system to improve smartphone security. They chose accelerometers, orientation sensors, and magnetometers to continuously learn the owner’s behavior patterns and environment characteristics so as to a better authenticate mechanism. Roy et al. [94] developed the Ripple II system, which uses microphones as a receiver of vibrations to achieve secure short-range communication. Compared to the ‘normal’ case where accelerometers are used as the receiver, ‘misusing’ microphone increases the data rate from 200 bits/s to 30,000 bits/s. Chen et al. [18] utilized magnetometers to differ an electronic speaker from a real person. Their work will detect the magnetic field emitted from loudspeakers as the essential characteristic to defend the smartphone against machine-based voice impersonation attacks.

In this thesis, Spy-Phone is an instance where ‘misusing’ motion sensors causes information leakage on smartphones, while Ultra-Unlock and MoVo use sensors to solve security problems.

1.3 Objectives and Contributions

The research objective of this thesis is to come up with new security and privacy (S&P) issues related to smartphone sensors, and to solve existing S&P problems by utilizing those embedded sensors.

In the thesis, one new attack is proposed and two novel solutions are introduced to improve smartphone authentication schemes. A summary of the main contributions are:

• We proposed a new attack on smartphones that uses the data from zero-permission motion sensors to infer permission-required acoustic information. With Spy-Phone, a seemingly harmless app will keep eavesdropping on smartphone speakers covertly. Compared to existing works, this Spy-Phone system is based on a speaker-independent
machine learning model and therefore smartphones become more vulnerable to such attacks.

- We proposed MoVo, a new authentication method that is able to authenticate smartphone users and defend against various voice-spoofing attacks at the same time. Utilizing motion sensors and microphones, MoVo can successfully detect replay attackers with an average accuracy of 90.43%.

- We proposed a silent and quick authentication method Ultra-Unlock that allows users to unlock and control smartphones without touching the phone. The idea is to treat hand movements as I/Q modulation on ultrasound signals. This system is a good alternative to other authentication methods when users have wet/dirty hands or wear gloves/goggles/masks.

In the remainder of this thesis, we will discuss each system in each chapter, and conclude the thesis in Chapter 5.
Figure 1.1: Real Readings of 18 Sensors on a Google Nexus 6P Device.
CHAPTER 2
SPY-PHONE: EAVESDROPPING ON SMARTPHONE SPEAKERS WITH MOTION SENSORS

We introduce the *Man-in-the-Phone* attack, which can turn smartphones into spy bugs. This attack is based on the fact that motion sensors (accelerometers and gyroscopes) can measure audio signals, though at a much lower sampling rate. This attack imposes a big threat to smartphone users since the phone’s operating system grants applications permissions to motion sensors automatically. Compared to prior works, the Man-in-the-Phone attack focuses on the *intra-device* scenario, where motion sensors eavesdrop on the same phone’s built-in speakers. With compressed sensing theories and machine learning techniques, we implement the attack in an eavesdropping system called *Spy-Phone*, which is able to filch various critical information from smartphone users. Experiment results show that Spy-Phone can learn user activity, speaker gender, speaker identity, and speech content with an average accuracy of 81%, 93%, 98%, and up to 90%, respectively. Apart from the good accuracy, the most significant contribution of this work is that the Spy-Phone system is *speaker-independent*. Unlike previous related works that need specific training data from the victim, Spy-Phone is trained just once on public speech datasets and can filch critical information from brand new victims.

2.1 Introduction

Smartphones have become one of the most popular devices in the last few years. According to Statista \(^1\), the current number of smartphone users in the world today is over 3 billion, and this means nearly 40% of the world’s population owns a smartphone.

In this thesis, however, we demonstrate how to turn smartphones to spy bugs which eavesdrop on everything played by smartphones’ built-in speakers. Three billion smartphones? No, they are 3 billion spy-phones! This dreadful attack, referred to as the *Man-in-the-Phone* attack, is based on the fact that motion sensors (accelerometers and

\(^1\)https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/
Figure 2.1: Example of an Attacking Scenario by Spy-Phone. A seemingly harmless application (a weather app for example) is installed on the phone, and it keeps accessing the motion sensors in the background. Attackers can infer acoustic signals from the under-sampled motion data.

gyroscopes) can catch acoustic signals like a crude microphone. Thanks to smartphones’ operating systems, accessing these sensors is effortless. For example, Android ² automatically grants app permissions to motion sensors at installation time. In other words, any app installed in a smartphone can be a tool for attackers to eavesdrop covertly.

An example of attacking scenarios is illustrated in Figure 2.1. A boy has a video call with his mom. He wants to buy a book online and he needs her mom’s credit information to place an order. Her mom’s voice is played by the loudspeaker on the smartphone and affects the readings by motion sensors. The attacker has access to the motion data and therefore can infer the credit card information.

In fact, there have been some recent studies about the side-channel leakage from acoustic signals to smartphones’ motion sensor readings. Michalevsky et al. [74] proposed Gyrophone in 2014. To the best of our knowledge, they are the first to use smartphone gyroscopes as low-frequency microphones to listen to loudspeakers. Gyrophone can differentiate 11 digits with 65% accuracy based on a 10 people dataset. One year later, Zhang et al. [137] proposed AccelWord, which utilizes accelerometers to classify hotwords such as “Okay Google” or “Hi Galaxy” over other short phrases with 85% accuracy. AccelWord is also tested over 10 people.

²iOS, Windows, and Blackberry OS have similar permission-based sensor management systems [102]. In this work, we focus on Android.
However, techniques proposed in neither Gyrophone nor AccelWord can be used to perform a Man-in-the-Phone attack. Because these systems are built upon a **speaker-dependent** model, i.e. training dataset are labeled data from the target speakers. In the Man-in-the-Phone attack, attackers will not get *labeled* motion data from the victim — the attack system should be **speaker-independent**.

In 2018, Anald and Saxena [5] reproduced the aforementioned works and overturned their conclusions. They argued that smartphone motion sensors can not be affected by the speech signals transmitted through the air, no matter the sound source is a loudspeaker or a live person. They reported that only when the speakers and the motion sensors sharing a surface, the *conductive vibrations* will affect motion sensors’ readings. Except for this “Loudspeaker-Same-Surface” scenario, they studied 5 other scenarios [3] and concluded that smartphone motion sensors only pose a limited threat to speech privacy. However, they missed one important scenario, the *intra-device* scenario, where the speakers and motion sensors are inside the same smartphone. In 2019, they investigated this remaining scenario in an arXiv paper [6] and their SpearPhone system recognizes 11 digits with an accuracy of 71%. However, their technique is still speaker-dependent, which means the original speech data of the target speaker must be collected ahead of time. Such a requirement is very hard to be fulfilled in practice.

In this thesis, we studied the side-channel attack in the intra-device scenario. This *Man-in-the-Phone* attack, as we refer to it, is speaker-independent.

### Table 2.1: Maximum Sampling Rate of Smartphone Sensors

<table>
<thead>
<tr>
<th>Device</th>
<th>Release Year</th>
<th>Speakers’ Sampling Rate</th>
<th>Motion Sensors’ Sampling Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Samsung Galaxy S8</td>
<td>2017</td>
<td>192,000 Hz</td>
<td>500 Hz</td>
</tr>
<tr>
<td>Samsung Galaxy S7</td>
<td>2016</td>
<td>192,000 Hz</td>
<td>500 Hz</td>
</tr>
<tr>
<td>Google Nexus 6P</td>
<td>2015</td>
<td>48,000 Hz</td>
<td>400 Hz</td>
</tr>
<tr>
<td>LG Nexus 4</td>
<td>2012</td>
<td>48,000 Hz</td>
<td>200 Hz</td>
</tr>
</tbody>
</table>

[3]“Loudspeaker-Different-Surface”, “Laptop-Same-Surface”, “Phone-Different-Surface”, “Human-Normal”, and “Human-Loud”.
The main challenges in designing such a system are:

- The motion sensor readings are affected by at least four types of signal sources: sensor intrinsic errors, movement of the smartphone, acoustic vibrations from built-in speakers, acoustic vibrations from the air or other sound sources. An efficient filter is needed since only the acoustic vibrations from built-in speakers is the signal of interest.

- As shown in Table 2.1, compared to the sampling rate of smartphones’ built-in speakers which can reach 192 kHz, the sampling rate of motion sensors is 200-500 Hz. With such low frequency, human ears are no longer able to retrieve the original information, neither do state-of-the-art speech recognition systems [74].

- As illustrated in Figure 2.2, the system should be speaker-independent. Prior works such as Gyrophone [74] and AccelWord [137] can only achieve the claimed accuracy (65% for 11 classes and 85% for 3 classes) in the speaker-dependent setting. When Gyrophone uses a speaker-independent setting to identify digits, the accuracy is only 26%. This indicates that building a speaker-independent system is much more challenging than a speaker-dependent one.

Despite these challenges, the Spy-Phone system is able to learn a variety of critical information from smartphone users, such as user activity, speaker gender/identity, and speech content (as elaborated in Section 2.3). These achievements are largely credited to the compressed sensing theories which allow recovering certain signals from fewer samples than required in Nyquist paradigm (as elaborated in Section 2.2); and the machine learning techniques named Bi-directional Long Short-Term Memory (Bi-LSTM) network, which is a special variant of recurrent neural networks.

\[\text{Data is partially from [71] and partially by calling the \texttt{getMinDelay()} function of } \texttt{android.hardware.Sensor} \text{ class.}\]
(a) Speaker-Dependent: the target speaker’s training data is required. After machine learning procedures, the trained models for different speakers are different.

(b) Speaker-Independent: the model is trained on a group of speakers, and can be used to predict brand new speakers.

Figure 2.2: Man-in-the-Phone Attack Should be Speaker-Independent.
In summary, our main contributions are as follows:

- We uncover a new stealth attack named the Man-in-the-Phone attack that eavesdrops on smartphones’ built-in speakers by the intra-device motion sensors. Existing techniques in Gyrophone [74] and AccelWord [137] cannot be used for the Man-in-the-Phone attack because their systems are speaker-dependent, which require the training speech data from the victim. The Man-in-the-Phone attack, however, removes this requirement and thus is more dangerous and harmful.
- To the best of our knowledge, we are the first to apply compressed sensing theories in the audio-to-motion side-channel data so as to bridge the gap between sampling rates. This is the core technique we used to achieve the speaker-independence of the attacking system.
- We design the Spy-Phone system and validate its feasibility on learning user activity, speaker gender/identity, and speech content. Spy-Phone system can achieve higher accuracy than existing works. Besides, we have studied how different internal parameters (used in algorithms) and external parameters (properties of input data) affect the performance of this system.

2.2 Background

2.2.1 Voice Acoustics

The generation of human voice follows a source-filter model [33]. A speech signal can be seen as a source signal (the glottal source at the larynx, or noise generated at a constriction in the vocal tract), filtered with the resonances in the cavities of the vocal tract (tongue, teeth, lips, velum, etc. modifying the sound spectrum over time). This theory has been verified using 3-D printed models of two configurations of a vocal tract to generate sounds to generate the vowels in the words “had” and “heard” [129].

A typical adult male will have a fundamental frequency \(f_0\) of from 85 to 155 Hz, and that of a typical adult female from 165 to 255 Hz [7, 111]. The frequencies of the first, second, and the-i-th resonances are labeled as \(R_1, R_2, \ldots R_i\), and those of the spectral peaks produced by these resonances are called formants, \(F_1, F_2, \ldots F_i\) [112]. According to [58],
English vowels are perceived largely according to the values of the formants $F_1$ and $F_2$. The range of $F_1$ is roughly from 270 to 860 Hz, and that of $F_2$ from 840 to 2790 Hz [86]. As for English consonants, there are six categories: plosive/stop (e.g. /p/), fricative (e.g. /f/), affricate (e.g. /dZ/), nasal (e.g. /m/), lateral (e.g. /l/), and approximant (e.g. /r/). The frequencies of consonants vary a lot. The turbulence of /s/ and /z/ occurs above 3500Hz, and reaches as high as 10,000 Hz, whereas /w/ has $F_1$ from 250 to 450 Hz and $F_2$ from 600 to 850 Hz [57].

By Nyquist–Shannon sampling theorem, to properly sample a signal contains no frequency components higher than $f$ Hz, the sampling rate must be at least $2f$ Hz (Nyquist rate). In other words, a sampling rate of 400 Hz (motion sensors’ rate as shown in Table 2.1) can only handle signals whose component frequencies are below 200 Hz. Except for the part of the fundamentals, all $F_1$ and $F_2$ frequencies can not be sensed. Therefore, it is impossible to perceive the signals with such a low sampling rate.

However, borrowing theories from compressed sensing, the Spy-Phone system can partially reconstruct the signal and obtain critical information such as the numbers that appeared in a conversation, genders, or even identities of the speakers, etc., from motion sensor readings, as discussed in Section 2.3.

### 2.2.2 Compressed Sensing

Compressed sensing [29] (also known as compressive sensing [101], compressive sampling [15]) is a novel sensing/sampling paradigm that acquires and reconstructs signals in a much more efficient way than the established Nyquist–Shannon sampling theorem.

First introduced by Candes et al. in 2004 [13], compressed sensing takes advantage of prior knowledge about inherent characteristics (sparsity) of signals. In this way, even with far fewer samples, the signal of interest can still be perfectly (or nearly perfectly) recovered. The constraint of the Nyquist rate (sampling rate to be 2 times of signal bandwidth) is no longer a requirement. Therefore, we adopt this technique in the design of the Spy-Phone system.
As shown in Figure 2.3, the theory of compressed sensing has solid mathematical backgrounds. The signals of interest should have a low information rate, i.e., the signal is sparse in its original domain (e.g. time domain) or some transform domain (e.g. frequency domain) [16]. More precisely, when expressed in a proper representation dictionary $D$, a
large number of coefficients in vector $w$ are zeros or small enough to be ignored. In fact, natural signals such as sounds, images, or seismic data have this sparsity property and can be stored in compressed form, in terms of their projection on a suitable dictionary [90].

In the signal acquisition stage, compressed sensing aims to *undersample* the signal of interest such that the dimension size $M$ of measurements $x$ is much smaller than the dimension size $N$ of the signal $s$. This goal is achieved by using a sensing matrix $S$ of size $M \times N$, where $M \ll N$. The reconstruction of the original signal is essentially an optimization problem: finding the optimal sparse coefficient vector $w_{opt}'$ such that

$$w_{opt}' = \arg\min_{w'} (\gamma \|w\|_p + \|x - Rw\|_2),$$

(2.1)

where $\gamma$ is the parameter to balance the evaluation weight of sparsit y versus data error, $\|w\|_p$ is the $\ell_p$-norm of $w'$, and $R$ is the reconstruction matrix which is equal to $S \times D$. As $\gamma$ increases the solution is getting more sparse. When $p = 0$, this optimization problem has been proved to be NP-hard [78].

The overall performance of compressed sensing is determined by 3 aspects: how representative is the dictionary, how efficient is the sensing matrix, and how well-performed is optimization solver. There are mainly two types of dictionaries, predefined dictionaries and learned dictionaries. Predefined dictionaries are built from basic functions like Fourier transform. Common dictionaries of this type include the discrete cosine transform basis, wavelet packages, and Gabor bases [105]. Learned dictionaries are learned from a training dataset of signals. Methods to build such type of dictionaries include K-SVD [2], MOD ILS-DLA [31], ODL [67], RLS-DLA [104], and so on. As for the design of the sensing matrix, it is important to check whether the matrix will allow the recovery of a sparse solution. The most famous one is the restricted isometry property [14], though it has been shown too strict [28]. The signal reconstruction solvers span a wide series of techniques that include greedy pursuit, Bayesian framework, iterative thresholding, convex relaxation, nonconvex optimization, and brute force [115]. Some well-known solvers include OMP [114], GPSR [35], BCS [51], and so on. More information can be found in survey
papers like [140, 105, 92].

In our attack design, the audio signals played by smartphone speakers are the signals of interest. When the sound is collected by motion sensors, it is essentially the signal acquisition stage that applies the sensing matrix to get the measurements. The motion data has a very low sampling rate, much lower than the Nyquist rate. However, using a carefully designed reconstruction matrix, the original signals of interest can be (partially) reconstructed from the recovered sparse coefficient vector and the representation dictionary ($s' = D \times w'$).

### 2.2.3 Smartphone Hardware

![Smartphone Hardware Diagram](image)

**Figure 2.4:** Structure of a Desktop Loudspeaker Versus a MEMS Speaker.

Figure 2.4 shows the different structures of a typical desktop loudspeaker and a Micro Electro-Mechanical Systems (MEMS) speaker in smartphones. In a desktop loudspeaker, sounds are created by alternating currents to the voice coil, while a smartphone speaker uses a small MEMS tweeter. As a result, the MEMS speaker consumes much less power than the desktop loudspeaker. But for the same reason, the sound pressure level (SPL) generated by MEMS speakers is much lower than that of desktop loudspeakers.

The Google Nexus 6P and Samsung Galaxy S8 used in this work can only generate sound
with a maximum output of 78.4 dB\textsuperscript{5}. The commercial off-the-shelf desktop loudspeaker, the $22.99 Logitech Multimedia Speakers Z200 as an example, has max SPL greater than 88 dB. Note that a normal speech between two people typically has a range of 50 to 60 decibels and when they are shouting, the range goes to about 75 dB while 15% of men can shout over 96 dB\textsuperscript{12}. The higher the decibels, the easier for the motion sensors to catch the sound signals. With this respect, performing the Man-in-the-Phone attack is harder than Gyrophone\textsuperscript{74} or AccelWord\textsuperscript{137}.

As for the motion sensors, Google Nexus 6P uses Bosch BMI160, whose sampling rate can be 1600 Hz. However, the Android operating system only supports up to 400 Hz in order to save power.

### 2.3 Threat Model in the Spy-Phone System

The attacker’s goal is to eavesdrop on everything played by the smartphone speakers without the user’s awareness.

The attack begins when the user installs a seemingly innocent application, e.g. a car racing game with motion-control steering wheel (tilt the smartphone to steer). We assume such a disguised app has the access to motion sensors (accelerometers and gyroscopes) as well as the network. This assumption is easy to fulfill since the permissions to motion sensors and the internet are all considered as normal permissions by the Android operating system\textsuperscript{6}. In other words, Android automatically grants the app these permissions at installation time. The operating system doesn’t prompt the user to grant permissions, and users cannot revoke these permissions. Moreover, almost every smartphone has motion sensors and is able to connect to the Internet. The Man-in-the-Phone attack is therefore a threat to every smartphone user.

There is no other requirement for the attacker to conduct a Man-in-the-Phone attack. The disguised app just runs in the background and keeps monitoring the motion sensors. Since the power consumption of the motion sensors is very low, the user will not know the

\textsuperscript{5}https://www.phonearena.com/phones/benchmarks
\textsuperscript{6}https://developer.android.com/guide/topics/permissions/overview
motion sensors are in use. In addition, some smartphones are set to be “Rotation On” or “Lift to Unlock”, which means the operating system automatically collects motion data, and the Spy-Phone system does not introduce extra consumption by using the sensors.

The sensor data are sent back to the attacker over wireless networks. The attacker can choose to transmit data only when the smartphone is connected to Wi-Fi; otherwise, the user may notice the attack through suspicious cellular data usage. The data will be processed at the attacker’s end. By utilizing compressed sensing, machine learning and other signal processing techniques, Spy-Phone recovers critical information from undersampled motion data. The critical information could be, but not limited to,

- User activity. Using motion sensor to recognize user activity is not new. However, those activities (e.g., sitting, walking, running or exercising) are recognized based on different macro motions. Spy-Phone, on the other hand, is utilizing the micro motions caused by speakers. Therefore, Spy-Phone can tell whether a user is listening to music or watching an online talk even when the phone seems stationary. Smartphones’ built-in speakers are often used for alarms, phone call ringing, music listening, background sound for game playing, and so on. Different activity plays different sound and creates different motion sensor readings. The Spy-Phone system can be trained to classify the motion data to these different user activities. Put the matter another way, an attacker can know when the user wakes up, when she receives phone calls, how long she listens to music, or how long she plays games, and so on.

- Speaker gender/identity. When the smartphone user has a audio call, video call, or an online meeting with others, the Man-in-the-Phone attackers can learn whether the person she talks to is female or male. Spy-Phone can also learn how often the user contacts each different person. Moreover, if the attacker could get those people’s voice samples (either by recording in public area, or acquiring from social media online, etc.), the attacker can know exactly who they are.

- Speech content. Another goal for the attacker is to recover the whole speech content by motion sensors. However, considering the tremendous gap between the sampling rate
of smartphone speakers and motion sensors, there is still a long way before success. In Section 2.4, as a proof-of-concept, we use digit recognition to demonstrate the design of the Spy-Phone system, since the most critical information such as banking account numbers, credit card information, and certain passwords, are essentially combinations of digits. In Section 2.5, command recognition is also tested.

In addition, it is worth mentioning that the Man-in-the-Phone attack is built upon a speaker-independent machine learning model and does not require any specific training data from the user. Though with such data, the accuracy might be further improved. Prior works such as Gyrophone [74] and AccelWord [137] can only be used in the speaker-dependent case. Using their techniques, the user must obtain the victim’s speech first, which is harder to be carried out in reality.

2.4 Attack Design

The workflow of the Man-in-the-Phone attack in Spy-Phone is shown in Figure 2.5. The system can be separated into two parts: the user side and the attacker side. On the user side, a smartphone user downloads a seemingly harmless application and installs it on her device. Then the disguised app runs in the background and keeps monitoring the motion sensors. The sensor readings are then uploaded and sent to the attacker. On the attacker side, there are training steps and prediction steps. The training steps start from a training dataset that contains both sound data and motion data, while the input for prediction steps only contains motion data.

The training steps first apply compressed sensing techniques on the sound data and build a learned dictionary from audio files, then process the corresponding motion data to remove unwanted signal components. With the preprocessed motion data and a learned dictionary, the signal is reconstructed to a signal with more samples. Afterward, the reconstructed signal is fed to a Bidirectional Long Short-Term Memory (Bi-LSTM) network to train a classification model. The classification labels/classes (e.g. digits, genders, activity types, and so on) is determined by the sound data only.
Seemingly Harmless Application
Target Smartphone
Accelerometers
Gyroscopes
Target Smartphone
Sound Data
Motion Data
Training Dataset
Dictionary Learning
Signal Reconstruction
Bidirectional Long Short-Term Memory Network
Classification Model
Evaluation
Predictions
Output Critical Information
Training Steps
Prediction Steps

Figure 2.5: The Man-in-the-Phone Attack Workflow in the Spy-Phone System.
The prediction steps start with motion data, go by preprocessing, adopt the learned dictionary to reconstruct signals, and finally use the trained Bi-LSTM model to output critical information, without the presence of sound data.

In this section, we explain in detail how the Spy-Phone system implements the Man-in-the-Phone attack to eavesdrop on digits played by smartphone speakers. A similar approach can be applied to obtain user activity type or speaker gender/identity as evaluated in Section 2.5.

### 2.4.1 Training Dataset

**Table 2.2: Dataset Information**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>TIDIGITS [62]</th>
<th>Speech Commands [127]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sampling Rate</td>
<td>20,000 Hz</td>
<td>16,000 Hz</td>
</tr>
<tr>
<td>No. Speakers</td>
<td>326</td>
<td>2,618</td>
</tr>
<tr>
<td>Labels</td>
<td>11 Digits</td>
<td>35 Command Words</td>
</tr>
<tr>
<td>No. Utterances</td>
<td>7,172</td>
<td>105,829</td>
</tr>
<tr>
<td>Training Size</td>
<td>3,586</td>
<td>84,843</td>
</tr>
<tr>
<td>Validation Size</td>
<td>-</td>
<td>9,981</td>
</tr>
<tr>
<td>Testing Size</td>
<td>3,586</td>
<td>11,005</td>
</tr>
</tbody>
</table>

As shown in Table 2.2, we use two datasets, TIDIGITS and Speech Commands. TIDIGITS [62] are professional recordings of isolated digits, which has been used in [74] and [6]. Therefore, in the main sessions of this chapter, we illustrated our system using this dataset for the comparison’s purpose. However, TIDIGITS only contains digits, and the utterances are all recorded in laboratory conditions. It is natural that the accuracy would be higher on such a dataset. Therefore, we consider another dataset Speech Commands [127], which is the TensorFlow Speech Commands Dataset(Version 2). This dataset can be used for limited-vocabulary speech recognition. It consists of 105,829 utterances of 35 command words such as up, down, forward, stop, house, happy, etc. This dataset is recorded by a web-based application in a crowd-sourcing manner. Speakers are from all over the world and they speak the commands in uncontrolled environments. Basically, if a model trained on Speech Commands works well, it indicates that any piece of speech recordings online can be
used as the training data for the Spy-Phone system. The result using this dataset is shown in Section 2.5.2, the correct rate is 83.2%.

From now on, we focus on TIDIGITS.

![Spy-Phone Data Collection Application](image)

**Figure 2.6:** Spy-Phone Data Collection Application

In the beginning, the TIDIGITS dataset is used to build both the sound data and the motion data for the training dataset. This corpus contains speech of 11 isolated digits: “one”, “two”, . . . , “nine”, “zero” and “oh”, which are collected using an Electro-Voice RE-16 Dynamic Cardiod microphone, digitized at 20,000 Hz. These audio files are directly used as training sound data. The motion data, however, are collected by playing these audio files using the built-in speakers of a Google Nexus 6P device. They are the simultaneous recordings from the same phone’s accelerometers and gyroscopes with a sampling rate of 400 Hz. When playing the sound, the volume is set to be the highest level since these data will be used for training and the higher volume, the higher accuracy (according to experiments in Section 2.6.4). Note that when the Man-in-the-Phone attack is conducted in practice, the input motion data may come from a lower volume setting defined by the user. The
attacker cannot control the volume setting of the target smartphone, but she can control the volume when building the training dataset. For the training dataset, only part of the sound and motion data are used. The impact of the training data size on the prediction accuracy is elaborated in Section 2.6.1. All data are collected by the SpyPhoneApp as shown in Figure 2.6.

2.4.2 Dictionary Learning

We now demonstrate how to construct efficient representations of audio files by building a dictionary learned from the data itself. Recall Section 2.2.2, a learned dictionary is used to reconstruct a signal when it is undersampled. In the Spy-Phone system, the sound data are the signals of interest while the motion data are the measurements. The training sound data are grouped into 11 classes, (one to nine, zero, and oh). For each class $i$, the dataset is denoted by $\mathcal{T}_i$ and the representation dictionary $D_i$ is computed by Algorithm 1.

As shown in Figure 2.3, each representation dictionary $D_i$ is a collection of $K$ atoms, where each atom is a column vector of length $N$. An atom is basically some typical patterns of the signal of interest. For sound signal $s_i$ of class $i$, it should be represented or approximated

---

**Algorithm 1: BuildDictionary**

**Input:** Training Sound Dataset $\mathcal{T}_i$ for class $i$, Downsample Rate $r$, Dictionary Size ($N, K$)

**Output:** Dictionary $D_i$ for class $i$

```plaintext
1 $S_i \leftarrow []$ // Initialize training vectors
2 foreach audio file $\tau$ in $\mathcal{T}_i$ do
   // get signal $s$ and sampling frequency $f_s$
   $[s, f_s] = \text{audioRead}(\tau)$
   // remove unvoiced part
   $s \leftarrow \text{removeSilence}(s)$
   // downsample signal to $r$ subsignals and buffer each subsignal of length $N$
   foreach $j$ from 1 to $r$ do
      $s \leftarrow \text{shiftLeft}(s, 1)$ // shift left by 1 sample
      $ss \leftarrow \text{downSample}(s, r)$
      $bs \leftarrow \text{buffer}(ss, N)$
      $S_i \leftarrow \text{concatenate}(S_i, bs)$
   // run K-SVD dictionary training algorithm
10 $D_i \leftarrow \text{kvsd}(S_i, N, K)$
```
Figure 2.7: Example of Learned Dictionary Atoms. For each digit class, two different atoms are shown.

as a linear combination of some few of the dictionary atoms. Mathematically, $s_i = D_i \ast w_i$, for each column $s_i$ in $S_i$. Here $S_i$ is the training vectors calculated in Algorithm 1. Compared to the original sound signal from an audio file, $S_i$ is the result from many functions including removeSilence(), downSample(), and buffer().

Note that the length of a dictionary atom must be the same as the length of the training vector $s_i$. Since the original sound signal can have at most $2 \times 20,000$ Hz = 40,000 samples (The duration of audio files in the dataset is at most 2 seconds.). If directly using original sound to train the dictionary, the atom size must be 40,000 as well. However, not all signals in an audio file are informative. By applying removeSilence(), the unvoiced part of the audio signals is removed, which significantly improves the space and time efficiency of the
algorithm. The `removeSilence()` is based on [91], which calculates the short-time energy of signals and conducts zero-crossing analysis to differentiate sounding and unvoiced parts.

Moreover, from voice acoustics elaborated in Section 2.2.1, the most informative frequency range is roughly 100-4000 Hz, the 20,000 Hz sampling rate oversamples human speech. Therefore, to build a better representation dictionary, we shift the signals (Line 6 in Algorithm 1) and downsample it (Line 7 in Algorithm 1) by keeping the first sample and then every \( r \)-th sample after the first. The impact of the downsample rate \( r \) is evaluated in Section 2.6.2.

Last but not least, different people say different digits with intrinsically different time duration, which results in different signal lengths. To build a general representation dictionary, we buffer every signal with the fixed buffer size as \( N \) using `buffer()`. In other words, no matter how long is the original sound signal, by Algorithm 1, it is transformed into a matrix \( S_i \) with \( N \) rows. The number of columns, however, is not fixed. For convenience’s sake, we denote this number as \( L_i \).

Dictionary Learning is the process of finding a dictionary, \( D_i \) of size \( N \times K \), and a corresponding coefficient matrix \( W_i \) of size \( K \times L_i \) such that the approximations of the training vectors, \( S_i \) of size \( N \times L_i \), are as good as possible, given a sparseness criterion on the coefficients. Mathematically, the dictionary learning problem can be formulated as an optimization problem with respect to \( D_i \) and \( W_i \):

\[
\{ D_{i,\text{opt}}, W_{i,\text{opt}} \} = \arg\min_{D_i, W_i} \sum_{l=1}^{L_i} (\gamma \|W_{i,l}\|_p + \|S_{i,l} - D_i w_{i,l}\|_2),
\]

where \( \gamma \) and \( p \) are as in Eq. (2.1), \( s_{i,l} \) is the \( l \)-th column of \( S_i \), and \( w_{i,l} \) is the \( l \)-th column of \( W_i \).

The solver we use for this optimization problem is K-SVD [2], as it is one of the most well-known shared dictionary learning algorithms [131]. We did not test all existing dictionary learning algorithms, but among those we tested, PCA [44], K-SVD, and GAD [50], K-SVD provides the best results.
K-SVD is an iterative method with two main steps: First, keep \( D_i \) fixed then solve \( W_i \), which is essentially \( L_i \) separate problems as in Eq. (2.1); Second, keep only non-zero positions in \( W_i \) fixed and find \( D_i \) and \( W \) using singular-value decompositions (SVD). Figure 2.7 shows the first two atoms in the learned dictionary of each digit class using K-SVD. Generally, the atoms are different inter-class and similar intra-class.

By concatenating every \( D_i \) together, the overall dictionary is

\[
D = [D_1, D_2, \ldots, D_i, \ldots, D_{11}], \tag{2.2}
\]

which will be used for signal reconstruction in later steps.

### 2.4.3 Motion Data Preprocessing

The input motion data are collected similarly to how motion data are collected for the training dataset, i.e., playing TIDIGITS audio files by the smartphone’s built-in speakers. However, since these data are to simulate the attacking scenarios in reality, the data are collected multiple times using 15 different volume settings (Section 2.6.4). Moreover, we collected the motion data in both quiet and noisy environment (Section 2.6.5), and in both stable and moving states (Section 2.6.6).

![Magnitude Response (dB) and Phase Response](image)

**Figure 2.8:** The Magnitude and Phase Response of the FIR Highpass filter.
Figure 2.9: Preprocessing of Motion Data. The top two figures show the raw data from 3-axis accelerometers and 3-axis gyroscopes. There are 450 samples shown in the figure. These samples span about one second (450 / 400Hz = 1.125s) and are collected when the user is dropping the head of the phone while playing a “Zero” utterance from a male speaker. The frequency of such movement is far less than the frequency of human voices. Therefore, by applying the high pass filter, the noise caused by hand movements will be removed.
As mentioned in Section 2.1, the motion data is impeded by low sampling rates, weak target signals, and large interference noises. To overcome such problems, we must preprocess the motion data. We apply a high pass filter to mitigate the noise and increase the signal-to-noise ratio. The cutoff frequency is set to be 50 Hz, since noises such as walking or other human body movements are unlikely to generate signals as high as 50 Hz. We also use the speech-background separation algorithms from [91] to differentiate the data affected by sound signals from those do not. The red vertical lines in Figure 2.9a are the borderlines between the speech part and the background part. Figure 2.9b and Figure 2.9c show that various noises can be removed by applying a high pass filter. The result after the preprocessing stage is shown in Figure 2.9d. These data will be used in the next signal reconstruction stage.

2.4.4 Signal Reconstruction

In this stage, the preprocessed motion data will be used to reconstruct the original audio data.

Mathematically, the goal of signal reconstruction is to solve \( w' \) in \( x = R \times w' \) as in Figure 2.3. Here the measurement \( x \) is the motion data after preprocessing, and the reconstruction matrix \( R \) is calculated by

\[
R = S \times D, \tag{2.3}
\]

where \( S \) is the sensing matrix and \( D \) is the dictionary obtained in Section 2.4.2.

In the Spy-Phone system, the sensing matrix describes the sensing procedure from audio data to motion data, which can be regarded as a downsampling operation. The downsampling rate \( r \) is determined by the sampling frequencies of smartphone speakers \( (f_1) \) and motion sensors \( (f_2) \): \( r = f_1/f_2 \). For example, a sensing matrix with downsample
rate \( r = 3 \) is as follows:

\[
S = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & 0 & 0 & 0 & \cdots & 1 & 0 & 0
\end{bmatrix}
\]

In the Spy-Phone system, the true value of this rate can be as high as \( 48,000 / 400 = 120 \), which indicates a huge information loss and a big challenge to recover the signals.

Note that Algorithm 1 also has a downsampling operation (Line 7), therefore the \( R = S \times D \) operation can be replaced by changing the parameter \( r \) in Algorithm 1. When reconstructing the signals, \( D \) is used as \( R \). The impact of the downsample rate \( r \) is evaluated in Section 2.6.2. To solve Eq. (2.3), GPSR [35] is used, which is able to find 

\[
w_{opt}' = \arg\min_{w'} (\gamma \| w' \|_p + \| x - Rw' \|_2)
\]

The last step is to reconstruct the signal of interests by \( s' = D \times w_{opt}' \). These signals will be used as training data for the Bi-LSTM network discussed in the next section.

### 2.4.5 Bi-LSTM Learning

The last stage is to use the reconstructed data to establish a Bi-directional Long Short-Term Memory (Bi-LSTM) network model, which will be used for classifying the input data later on. LSTM was first proposed by Sepp Hochreiter and Jürgen Schmidhuber in 1997 [45]. It is a special variant of Recurrent Neural Networks (RNN), and is widely used in learning, processing, and classifying sequential data because of its great property of selectively remembering patterns for long durations of time. Over the years, there have also been many variants of LSTM networks. However, based on a study in 2017, none of the variants can improve upon the standard LSTM architecture significantly [38]. Therefore, we still choose to implement the standard LSTM network in this work except for the bi-directional calculation. The original unidirectional LSTM network only preserves information from the inputs seen in the past. Bi-LSTM network, on the contrary, preserves information both from the past and the future. As shown in Figure 2.10, our Bi-LSTM
network has five layers in total. In the sequence input layer, the input data have 6 feature dimensions, which consists of 3 accelerometer dimensions and 3 gyroscope dimensions. Then we establish an LSTM layer formed by LSTM blocks, where each block publishes its cell state to the next LSTM block. The output of the LSTM layer is sent to the fully connected hidden status layer. We set the total number of hidden units to be 100, and each hidden unit has two hidden states, one from the past and the other from the future. Then we feed the combined hidden status to a softmax function and output the classification results.

2.5 Feasibility Experiments

In this section, we validate the Spy-Phone system and show it can eavesdrop on smartphone’s built-in speakers and obtain critical information discussed in Section 2.3.
<table>
<thead>
<tr>
<th>Work</th>
<th>Sensors</th>
<th>Setting</th>
<th>Main Techniques</th>
<th>Classification Classes</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gyrophone</td>
<td>Gyroscopes</td>
<td>10 people from TIDIGITS Desktop Loudspeakers</td>
<td>SVM GMM DTW</td>
<td>1 to 9, Oh, Zero (11) Speaker-independent: 26% Speaker-dependent: 65% Speaker Identification 65% Speaker Gender 84%</td>
<td></td>
</tr>
<tr>
<td>[74]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accelword</td>
<td>Accelerometers</td>
<td>10 people</td>
<td>Decision Tree</td>
<td>Ok Google, Hi Galaxy, Others (3) Speaker-dependent: 85% Speaker Identification 86%</td>
<td></td>
</tr>
<tr>
<td>[137]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SpearPhone</td>
<td>Accelerometers</td>
<td>326 people from TIDIGITS Built-in Speakers in Smartphones</td>
<td>SVM with SMO Logistic RF RT</td>
<td>1 to 9, Oh, Zero (11) Speaker-dependent: 71% Speaker Identification 80% Speaker Gender 90%</td>
<td></td>
</tr>
<tr>
<td>[6]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spy-Phone</td>
<td>Both</td>
<td>Same as above</td>
<td>Compressed Sensing Bi-LSTM</td>
<td>1 to 9, Oh, Zero (11) Speaker-independent: 90% User Activity 81% Speaker Gender 93%</td>
<td></td>
</tr>
</tbody>
</table>
The main result and the comparison to prior works are summarized in Table 2.3.

### 2.5.1 Speech Content Learning (Digits)

The TIDIGITS dataset [62] has 7172 audio files of isolated digits. We use 3586 of them (3586 / 7172 = 50%) to train the dictionary. With the downsample rate $r$ set to 40, dictionary size set to $(N=400, K=10)$, and the $p$-norm set to be the $\ell_1$-norm (absolute value), we learn an overall dictionary of size $400 \times 110$ by concatenating each dictionary of each individual digit class. For each audio file, we play it by smartphones’ built-in speakers for 30 times. Therefore, the size of the motion data is actually 30 times of the size of the sound data in the training dataset. The Bi-LSTM network is actually trained with $3586 \times 30 = 107,580$ data, which are the resulted signals after preprocessing and signal reconstruction. Note that although this number seems big, each data is just 1-second data at a sampling rate of 400 Hz, and each sample is 16 bits. So the total size is $107,580 \times 400 \times 16 = 688,512,000$ bits $= 86.1$ MB, which is indeed not big at all.

The test data is motion data from the remaining 3586 audio files. This data has never been seen by the Bi-LSTM network before. The classification result is shown as a confusion matrix in Figure 2.11. In the confusion matrix, each row of the matrix represents the instances in a predicted/output class while each column represents the instances in an actual/target class. The digit class “zero” has the highest classification accuracy of 98.6%. The digit class “eight” has the lowest accuracy: only 70.5% instances are classified to the correct class, while 16.2% are classified as “six”. The overall accuracy of all 11 classes is 90.13%, with a sensitivity (true positive rate) of 90.13% and a specificity (true negative rate) of 99.02%.

### 2.5.2 Speech Content Learning (Commands)

The TensorFlow Speech Commands Dataset Version 2 [127] was used for limited-vocabulary speech recognition. This dataset consists of 105,829 utterances of 35 words such as forward, house, happy, etc. A random guessing provides accuracy of $1/35 = 2.9\%$, but Spy-Phone could achieve 73.4%. The accuracy is relatively lower than
**Figure 2.11:** The Confusion Matrix of the Digit Classification Result.

TIDIGITS, as TIDIGITS are professionally recorded while the commands data are crowdsourced online. Using the original 16,000 Hz audio data can only achieve accuracy of 88.2% [127]. Spy-Phone (using 400 Hz motion data) has the correct rate of 73.4/88.2=83.2%.

### 2.5.3 Speaker Gender Classification and Speaker Identification

The same training and testing data are also used for gender classification and speaker identification, since the TIDIGITS audio files are labeled with gender and speaker ID. The
result for gender classification is shown in Figure 2.12.

### 2.5.4 User Activity Classification (Sound Type)

We also test whether the Spy-Phone can classify the type of sound signals played by smartphone speakers. The dataset consists of two parts: the first is the built-in (default) alarm sounds, notification sounds and ringtones provided by the Android operating system, the other is the speech sounds from the TIMIT [52] dataset where 10 sentences are spoken by each speaker. In detail, we obtain 18 alarm sounds, 11 notification sounds, and 12 ringtones from `/system/media/sound/` on a Google Nexus 6P smartphone with an Android 8.1 “Oreo” system. The speech data consist of 10 sentences from a male speaker and 10 sentences from a female speaker. We increase the atom size $N$ to 800 when building the dictionary, so that the measurement size is 800 as well, which means a piece of motion data of 800 / 400 Hz = 2 seconds. In other words, when classifying the sound type, we evaluate the motion data using a 2-second threshold, which is longer than that of classifying the digit (1 second).

The result is shown in Figure 2.13. The Spy-Phone system can successfully differentiate between different sound types with an overall accuracy of 80.56%, which means attackers can know when you get up in the morning (“Alarm”), when you receive a notification (“Notification”), when you are called by others (“Ringtone”), and when the person from the other end of call starts speaking (“Ringtone” followed by “Speech”). Moreover, attackers can also infer activities such as watching videos or listening to audio-books (“Speech” without
Figure 2.13: The Confusion Matrix of the Sound Type Classification Result.

“Ringtone” as a precursor). In other words, various sound-related user activities are not secrets to Man-in-the-Phone attackers, not to mention that motion-related activities can be monitored by motion sensors as a default. Note that the average accuracy of identifying “speech” is 94.7%, so the Spy-Phone system can first determine whether the input belongs to “speech”, then classify the speaker gender/identity or the digit class as mentioned above.

The overall accuracy is much lower because of the other three classes. These classes are misclassified since alarm sounds, notification sounds and ringtones are not strictly defined. Android groups them in a generally conventional way, not by scientific methods. In fact, smartphone users may choose to use ringtones as alarms, or use alarm sounds as ringtones. Such inherent ambiguity is the main reason for the low accuracy. To further improve the accuracy, more features such as the total duration (“notification” tends to be shorter than “ringtones”) or the repetitive (“alarm” tends to ring once a day) should be considered. Integrating algorithms to learn these features is a potential future work for a better design of Spy-Phone.

2.6 Impact Evaluation

In this section, we evaluate the impact of two internal parameters and three external parameters on the performance of the Spy-Phone system for digit classification. The internal parameters control how the dictionary is learned and the external parameters control the
quality of input motion data. Note that a random guess results in an accuracy of $1/11 = 9.09\%$.

2.6.1 Impact of Training Data Size

The size of training data for dictionary learning is an internal parameter. In the previous section, we use 3586 audio files (50% of full dataset) to train the dictionary. In this section, we vary the size from 660 to 1760 and show the experiment results in Figure 2.14, which is a box and whisker plot\textsuperscript{7}. We can see that the classification accuracy increases from $\sim56\%$ to $\sim78\%$ as the data size increases. This result is reasonable since the more data is used in training, the learned dictionary is more representative and the machine learning model is more accurate. In fact, recent researches have shown that to build a representation dictionary, the typically sufficient number of training samples grows up quasilinearly with the signal dimension, i.e., $\mathcal{O}(N \log N)$ [93]. In our experiment, the atom size $N$ is set to be 400, therefore, several hundred or a few thousand of training data should be enough.

2.6.2 Impact of Downsampling Rate

The downsampling rate $r$ is the other internal parameter to study. This parameter is influenced by four frequencies: the sampling rate of smartphones’ built-in speakers

---

\textsuperscript{7}The ends of the box are the upper and lower quartiles (25th and 75th percentiles), the central line inside the box indicates the median, the whiskers extend to the highest and lowest accuracy values not considered outliers, and the outliers are plotted individually using the ‘+’ symbol.
(48,000 Hz in Nexus 6P), the sampling rate to record human speech (20,000 Hz in TIDIGIT), the frequency range of human speech (100-4,000 Hz), and the sampling rate of motion sensors (400 Hz in Nexus 6P). We vary this rate from 5 to 50 and find that the performance of Spy-Phone improves with the increase of the downsampling rate at the beginning, then the accuracy enters a relatively stable stage when $r = 30$ and $r = 40$. The accuracy declines if a higher downsampling rate is used ($r = 50$). From Figure 2.15, when $r \in [20,50]$, the average accuracy is above 90%. We do not test the cases when $r > 50$, because $50 = 20,000Hz/400Hz$ is the gap between the sampling rate of training sound data and that of motion data. If we use a larger downsampling rate, the learned atoms would contain less information than the motion data, which invalidates the dictionary and contradicts with the goal of using compressed sensing to reconstruct more signal samples.

### 2.6.3 Impact of Training Device

We tested whether the trained LSTM network is robust across different devices. Due to time limit, only two devices are tested with the result shown in Table 2.4. Different devices with same model, and different devices with different model will be tested as a future work.

### 2.6.4 Impact of Sound Volume

In this section, we study how the sound volume affects the performance of Spy-Phone. By calling the `getStreamVolume(AudioManager.STREAM_MUSIC)` for the AudioManager
Table 2.4: Statistical Analysis of the Speaker Identification Result.

<table>
<thead>
<tr>
<th></th>
<th>Nexus 6P Training</th>
<th>Galaxy S8 Training</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nexus 6P Testing</td>
<td>90.52%</td>
<td>83.26%</td>
</tr>
<tr>
<td>Galaxy S8 Testing</td>
<td>84.39%</td>
<td>90.48%</td>
</tr>
</tbody>
</table>

![Figure 2.16: Impact of Sound Volume.](image)

class, the Google Nexus 6P device supports 15 volume levels. All these volume levels are tested and the results are shown in Figure 2.16. The accuracy goes up when the volume goes up. The classification accuracy is above 60% when the volume is set to be level 10 or larger, and the accuracy goes to above 90% when the volume is set to be the highest two levels. Even when the volume is low, the average accuracy is about 30-40%, over 3 times higher than the random guess accuracy of 9.09%.

2.6.5 Impact of Surrounding Environment

In this section, we test how background noises like real human talking influence the classification accuracy. When the smartphone speakers are playing sounds, a real person talks 20 cm away from the smartphone, with a similar volume. The result is shown in Figure 2.17, the overall accuracy for all classes is about 60%. Since Spy-Phone still works in this experiment, it is an evidence that motion sensors are more sensitive to the built-
in speakers than other sound sources that transmit signals through the air. It is worth mentioning that the performance of Spy-Phone can be further improved if more training data is added. At this time, all training data are collected in a quiet environment, so directly feeding an input data from a noisy environment decreases the accuracy from 91% to 60%. Adding data from various environments, however, is expected to increase the robustness of the system and therefore a future research direction.
To further test the impact of noises, we use a babble noise file and play the noise with different volumes. We control the sound pressure level received by the target phone to 55dB (quiet urban street), 60dB (normal conversation), and 65 dB (busy street). The results are shown in Figure 2.20.

2.6.6 Impact of User Mobility

![Figure 2.19: Impact of User Mobility.](image)

All previous experiments are conducted when the smartphone is placed on a desk, without being touched or moved during the data collection stage. As illustrated in Figure 2.9, user movements are regarded as noises. Fortunately, since user movement is very slow and is unlikely to generate signals as high as 50 Hz. Thus, we can apply a high pass filter to mitigate the noise. In Figure 2.9c, we see noises are removed after the filter. In this section, we validate the performance of Spy-Phone with the presence of user mobility.

We conduct two experiments: the walking scenario and the driving scenario. The users put the phone in their pockets while playing the sounds using the speakers. The results are shown in Figure 2.21. The accuracy for the walking scenario is 83.94% and that of driving scenario is 73.34%. The accuracy of the driving scenario is relatively lower because of frequencies of the noises are relatively higher and thus close to the aliasing of human speech.
![Table 2.20](image)

**Figure 2.20:** The Impact of Surrounding Environments: Quiet Street, Normal Conversation, and Noisy Street. Note that the decibel values are the average sound pressure levels received by the smartphone.
Figure 2.21: Accuracy in the walking (left) and driving (right) scenario. The machine learning model trained in the stable environment is directly used to predict the testing data collected when user is walking or driving.
Moreover, we conduct an experiment in the extreme case that the user holds the phone in the hand and keeps moving it. The results are plotted in Figure 2.19, the average accuracy can still achieve 45%.

2.7 Defenses

To defend against the Man-in-the-Phone attack, the smartphone user can adopt the hardware-based defenses or the software-based defenses if supported by the smartphone’s operating systems.

2.7.1 Hardware-based Defenses

The easiest way to defend the Man-in-the-Phone attack is to not use smartphone’s built-in speakers. Instead, a user can use headphones or other wireless connected loudspeakers. As long as there is no direct contact between the speakers and the motion sensors, the attack is defended [5]. In fact, as shown in Section 2.6.4, with lower volume setting, the Man-in-the-Phone attack can also be largely impeded. Though such hardware-based defenses may cause inconvenience to the user, their security and privacy can be protected.

2.7.2 Software-based Defenses

The software-based defenses are better sensor management by smartphones’ operating systems (OS). For example, the OS should treat the permissions to motion sensors as dangerous permissions and require users to grant permissions at installation time. Moreover, the OS should keep monitoring the sensor usage such as when the sensors are used, whether they are used in background, and what sampling rate is required by the application.

Though there has been some research in designing better sensor management systems [102], the Man-in-the-Phone attack may still be a big threat for at least three years. This is because the smartphone users may not or can not upgrade to the newly-released OS in a timely manner. Indeed, by the end of May 7, 2019, more than half the Android devices are still using OS released three years ago. Therefore, three years

---

8More than half the Android devices are still using OS as old as Android Nougat, which was released in 2016 according to Android Dashboards (https://developer.android.com/about/dashboards)
later, there is a large chance that more than half the devices are still using the OS released so far. Since current Android OS cannot defend the SpyPhone system, those devices are vulnerable to the Man-in-the-Phone attack. However, if smartphone users know the importance of the update and cares about their security and privacy, the adoption speed may increase. Note that there is a huge lag between Google and the manufacturers such as Samsung and Huawei. Even Google release the security updates quickly enough, the manufacturer may release them too late.

2.8 Related Work

2.8.1 Inferring Information From Smartphone Motion Sensors

In recent years, much attention has been paid to inferring private information through motion sensors in the literature. A typical side-channel attack is keystrokes inference on smartphones through motion sensors [83, 75]. The general idea of these attacks is that when typing on different locations on a screen, the keystrokes cause distinct vibrations or rotations. In addition, Wang et al. [122] proposed to track the movement of the wrist to infer what the user has typed. Similarly, a practical attack has been shown in [121], which infers a user’s personal PIN sequence by exploiting wearable devices. The feasibility of inferring user’s location information using motion sensors instead of GPS data has been shown in [63, 42]. Bojinov et al. [10] demonstrated that motion sensors can be used as device fingerprint to uniquely identify a device. This motion sensor-based device fingerprint was further utilized by [24] to track a user across multiple visits to websites. In [61], Lee et al. proposed to use motion sensors to infer users’ handwritten patterns. Huang et al. [48] implemented a reliable liveness detection system called Breathlive, which is based on the inherent correlation between sounds and chest motion caused by deep breathing. Roy et al. [95] demonstrated the possibility of communication through motion sensors by modulating the vibration motor and decoding through accelerometers. Recent works on activity recognition using motion sensors are presented in [100, 123]. In addition, a detailed survey of works on sensor-based threats for smart devices can be found in [103, 23]. Note that we only list related works
using smartphone motion sensors, but not works using standalone sensors. This is because smartphone motion sensors can only report readings at a low frequency, but standalone accelerometers or gyroscopes can record signals with frequencies as high as 10 KHz. Some special models of piezoelectric accelerometers can even measure 1 MHz signals.

2.8.2 Eavesdropping Sound Signals By Non-Acoustic Sensors

Recent studies have shown that sound signals can be eavesdropped through non-acoustic sensors instead of microphones. Among these side-channel attacks, MEMS motion sensors are widely used since motion sensors are prone to acoustic signals. Michalevsky et al. [74] found that the MEMS gyro sensors are able to pick up air vibrations from sound. They proposed GyroPhone, a new threat which uses gyroscope on smartphone to intercept human speech. Zhang et al. [137] proposed to utilize accelerometer for hotword detection to reduce power consumption. In addition, Anand et al. [5] also demonstrated that it is possible to eavesdrop speech signals in certain scenarios by using inertial sensors in a smartphone. Han et al. [41] proposed to combine multiple signals from non-acoustic sensors to create a higher sample rate signal for speech reconstruction. Hawley et al. [43] proposed to use sensors on smartphone to visualize the properties of sound directivity, interference and other acoustical phenomena. Recently, other techniques have been proposed to eavesdrop sound signals besides motion sensors. Roy et al. [96] have shown that the vibration motor can be used as microphone since the vibrating mass inside the motor responds to air vibrations from nearby sounds. Davis et al. [25] used a high speed camera to retrieve digital audio by capturing the vibration of objects near the sound source. Similarly, Fuse et al. [36] found that a better sound can be obtained by trying to recover the sound based on the vibration direction of the object. Kwong et al. [56] demonstrated that the mechanical components in magnetic hard disk drives can be used to extract and parse human speech with sufficient precision.
2.9 Conclusion

In this work, we introduced the Man-in-the-Phone attack and implemented it in the Spy-Phone system. This Spy-Phone system achieves user-independent attacking by compressed sensing and long short-time memory networks. The average accuracies for classifying user activity, speaker gender, speaker identity, and speech content are 81%, 93%, 98%, and up to 90%, respectively. Future work is to improve the robustness of Spy-Phone by adding more training data collected in noisy environment and in moving state, and by extending the vocabulary.
In this chapter, we propose a new authentication method for smartphones. During the authentication period, the smartphone’s built-in speaker transmits ultrasound signals continuously and the user draws a user-defined gesture in the air. Then the signals reflected from the user’s hand are fed into a pre-trained machine learning model to get the authentication results. The key idea is to use in-phase (I) and quadrature(Q) components to represent hand movements, which not only provides high accuracy but also decrease the data size. Moreover, to calculate I/Q components, we choose to use cascaded integrator–comb (CIC) filters, which utilizes only delay and addition and subtraction. Since CIC filters require no multiplication operations, it does not introduce high computational overhead and therefore can be implemented on most smartphones. Experiments of the proposed Ultra-Unlock system on Google Nexus 6P devices show that the system can identify 3 different gestures with an average accuracy of 90.89% and identify 6 people with an average accuracy of 78.47%. We also propose six directions to further increase the accuracy, as a guide for future work.

3.1 Introduction

Smartphones have been an indispensable part of our daily lives, and most of them still require a password or pin code if you want to unlock them. However, in today’s busy world, typing those passwords is a waste of time. As a result, biometric security systems have been on the rise and many smartphones adopt the fingerprints to do authentication. The problem is, if your hands are dirty or wet, the smartphone just won’t let you in! Researchers have developed a lot more other authentication methods, such as the face ID or smart card, but these methods all require special hardware. Moreover, if users wear face masks, air-purifying respirators, goggles, or face shields, as during the COVID-19 pandemic, authentication methods such as face ID do not work anymore. Some may argue
that users can use voice authentication systems instead. However, the security of voice authentication systems is relatively low due to the replay attack and many users don’t use the method in public because it’s uncomfortable talking to their phones with others around.

Therefore, we propose a new authentication method called Ultra-Unlock, which provides a silent, quick, and smooth user experience for unlocking without your hands touch the phone. It utilizes the speakers and microphones and conducts the authentication through processing ultrasound signals. The smartphone has the ability to reconstruct the user’s hand movement and determine whether it belongs to a legitimate user or not. Following the classification method proposed in [120], our method is a combination of knowledge-based method and identity-based method, since the user must have the knowledge of what the shape he should draw and meet the biometric features of how to draw the shape.

3.2 Related Works

Our Ultra-Unlock system has three important building blocks: 1) smartphones can generate and process ultrasound signals 2) the ultrasound signals are sensitive to hand movements 3) hand movements are feasible features for user authentication. There have been many researches on each building block.

First, commercial on-the-shelf smartphones can transmit and receive ultrasound signals, though the band is very narrow. Smartphones nowadays are mostly equipped with speakers and microphones whose highest sample rate is either 44100 Hz or 48000 Hz. According to the Nyquist-Shannon sampling theorem, in order to sample a signal of frequency \( f \), a sufficient sample rate would be \( 2f \). Then by the reverse, we know that commercial smartphones can generate and record sounds whose frequencies are lower than 22050 Hz or 24000 Hz. Since the range of human hearing is generally considered to be 20 Hz - 20000 Hz, we conclude that smartphones are able to create and process ultrasounds in the range of 20000 Hz - 22050 Hz or 20000 Hz - 24000 Hz.

\[ ^1 \] Some models support higher resolutions, for example, Samsung Galaxy S10 features 32-bit/384kHz Hi-Fi playback
Next, various algorithms have been developed to extract information on movements from ultrasound signals. Graham et al. [37] proposed a smartphone sonar system that calculates distances by measuring the elapsed time between the initial pulse of the ultrasound signal and its reflection. They were able to measure the distances of objects accurately with an error bound of 12 cm. Liu et al. [64] designed the Guoguo algorithm and ecosystem to realize the smartphone-based fine-grained indoor localization with average localization accuracy being about 6 cm - 15 cm in typical indoor environments. Nandakumar et al. [77] presented a sub-centimeter level smartphone tracking system which adopts a modulation technique commonly used in wireless communication called orthogonal frequency division multiplexing. Their FingerIO system can achieve 2D finger tracking with an average accuracy of 8 mm. Mao et al. [68] developed a high-precision acoustic tracker system by sending a distributed frequency modulated continuous waveform (FMCW). They also designed an optimization framework that combines FMCW estimation with Doppler shifts and inertial measurement unit measurements to enhance the accuracy to be 8 mm - 9 mm in 3D space. Wang et al. [125] proposed the LLAP system which measures the phase changes of the sound signals caused by hand/finger movements and then converts the phase changes into the distance of the movement. They increase the tracking accuracy to 7 mm in 2D space. Later on, Yun et al. [133] took into account multipath propagation and designed a novel acoustic-based device-free tracking system, called Strata, which outperforms FingerIO and LLAP. In 2018, Sun et al. [108] improved LLAP and the new system can capture finger movements with an accuracy of 3.59 mm.

Lastly, hand movements have intra-person similarity and inter-person difference. Therefore, it can be used for user authentication. works of literature on finger/hand movement based authentication methods. Niu et al. [80] proposed using finger gestures with taps to the screen to conduct authentication. They tested the recall and forgery of gesture authentication and show, using dynamic time warping, that even simple gestures are repeatable by their creators yet hard to forge by attackers when taps are added. Hong et al. [46] proposed Waving Authentication (WA) which is a motion gesture authentication
system based on accelerometers. WA utilizes eight distinguishing features hiding in the acceleration traces of motion gestures and exploits a one-class Support Vector Machine (SVM) for classification. Yang et al. [132] studied how free-form gestures perform in the wild. Their 91 participants generated 347 text passwords and 345 gesture passwords with 2002 completed log-in tasks. They found that, with gesture passwords, participants generated new passwords and authenticated faster with comparable memorability while being more willing to retry.

However, all the above gestures are collected from the smartphone touch screens. There is no existing paper on how to authenticate users based on their hand movements in the air.

3.3 Background

The general idea of Ultra-Unlock is to treat hand movements as a special I/Q modulation on a signal with fixed frequency. Different users have different hand movements, and modulate the same carrier signal in different ways. After collecting the modulated signals, machine learning techniques can be used for modulation classification. If the modulation is classified to be the hand movement of a legitimate user, the smartphone will accept that user.

In this section, we first provide the necessary background on I/Q data, and then discuss the relationship between hand movements and I/Q modulation.

3.3.1 I/Q components and I/Q modulation

The most common way to represent a wave is to use a series of samples of the momentary amplitude of the signal. However, this method cannot differentiate between a positive or negative frequency since they both generate the same curve, for example, \( \cos(x) = \cos(-x) \). This becomes a problem working with the signal. Mixing (multiplying) two signals and it’ll cause multiple solutions due to the uncertainty of the sign. A better way is using I/Q data and representing a wave using in-phase and quadrature components. As shown in Figure 3.1, the signal is plotted in three dimensions. The in-phase component and the quadrature component are the 2D projections of the signal. I/Q data together show the
changes in amplitude and phase of a sine wave. The in-phase component indicates the “real” signal. For two signals of a positive and a negative frequency, they will have the same in-phase component but reversed quadrature components. Moreover, if the signal is viewed along the time axis, the spiral winds counter-clockwise, which means the frequency is positive. The radius of each circle indicates the peak amplitude of the signal.

Mathematically, for a complex signal \( Ae^{i\varphi} \) where \( i \) is the imaginary unit, the in-phase component is \( I = A \cos(\varphi) \) and the quadrature component is \( Q = A \sin(\varphi) \). That is,

\[
Ae^{i\varphi} = A \cdot (\cos(\varphi) + i \cdot \sin(\varphi)) = I + Qi
\]
\[ A = \sqrt{I^2 + Q^2} \quad \text{and} \quad \varphi = \tan^{-1} \frac{Q}{I} \]

In electrical engineering, there are three basic ways to modulate a waveform: Amplitude Modulation (AM), Frequency Modulation (FM) and Phase Modulation (PM). All of them can be achieved by I/Q modulation. Suppose the carrier’s frequency is \( f \), I/Q modulation is to solve the following equation:

\[
\text{ModulatedSignal} = I \cdot \cos(2\pi ft) + Q \cdot \sin(2\pi ft).
\]

To decode the baseband signal, signal multiplication and low pass filters are needed:

\[
I = \text{lowpass}(\text{ModulatedSignal} \cdot \cos(2\pi ft));
\]

\[
Q = \text{lowpass}(\text{ModulatedSignal} \cdot \sin(2\pi ft)).
\]

### 3.3.2 Hand Movements and I/Q Data

Now we show how hand movements can be regarded as I/Q modulation. As shown in Figure 3.2, the smartphone speaker plays an ultrasound signal \( A \cos \left( 2\pi f \frac{n}{f_s} \right) \) with the peak amplitude of \( A = 1 \), the frequency \( f = 20,000 \) Hz, and the sampling frequency of \( f_s = 44,100 \) Hz. At the time \( t_p = \frac{n}{f_s} \), the sound propagation path \( p \) decreases by \( \Delta p \) due to the hand movement. The received signal therefore becomes \( 2A'_p \cos \left( 2\pi f \frac{n}{f_s} - 2\pi \frac{\Delta p}{\lambda} - \theta_p \right) \), where \( 2A'_p \) is the new amplitude, the term \( 2\pi \frac{\Delta p}{\lambda} \) comes from the phase lag caused by the propagation delay of \( \frac{\Delta p}{\lambda} \) and \( \lambda \) is the wavelength of the ultrasound. In our system, \( \lambda = \frac{c}{f} = \frac{343 \text{ m/s}}{20,000 \text{ Hz}} = 1.72 \text{ cm} \), where \( c \) is the speed of sound. The last term \( \theta_p \) is the initial phase, which is caused by the hardware delay and phase inversion due to reflection [125]. In conclusion, the existence of hand movement changes the amplitude and phase of the original signal. Since I/Q data shows the changes in amplitude and phase of a waveform, we use I/Q data to represent hand movements.
To calculate the in-phase value, we first multiply the received signal by \( \cos \left( 2\pi f \frac{n}{f_s} \right) \):

\[
2A_p' \cos \left( 2\pi f \frac{n}{f_s} - 2\pi \frac{\Delta p}{\lambda} - \theta p \right) \times \cos \left( 2\pi f \frac{n}{f_s} \right)
= A_p' \cos \left( 2\pi f \frac{n}{f_s} - 2\pi \frac{\Delta p}{\lambda} - \theta p - 2\pi f \frac{n}{f_s} \right) + A_p' \cos \left( 2\pi f \frac{n}{f_s} - 2\pi \frac{\Delta p}{\lambda} - \theta p + 2\pi f \frac{n}{f_s} \right)
= A_p' \cos \left( -2\pi \frac{\Delta p}{\lambda} - \theta p \right) + A_p' \cos \left( 4\pi f \frac{n}{f_s} - 2\pi \frac{\Delta p}{\lambda} - \theta p \right).
\]

Note that the second term \( A_p' \cos \left( 4\pi f \frac{n}{f_s} - 2\pi \frac{\Delta p}{\lambda} - \theta p \right) \) is a signal of frequency \( 2f \) and will be removed after applying the low pass filter. Therefore, we get

\[
I_p = A_p' \cos \left( -2\pi \frac{\Delta p}{\lambda} - \theta p \right).
\]

Similarly, we can get the quadrature value by multiplying \(-\sin \left( 2\pi f \frac{n}{f_s} \right)\):

\[
2A_p' \cos \left( 2\pi f \frac{n}{f_s} - 2\pi \frac{\Delta p}{\lambda} - \theta p \right) \times \left(-\sin \left( 2\pi f \frac{n}{f_s} \right) \right)
= A_p' \sin \left( 2\pi f \frac{n}{f_s} - 2\pi \frac{\Delta p}{\lambda} - \theta p - 2\pi f \frac{n}{f_s} \right) - A_p' \sin \left( 2\pi f \frac{n}{f_s} - 2\pi \frac{\Delta p}{\lambda} - \theta p + 2\pi f \frac{n}{f_s} \right)
= A_p' \sin \left( -2\pi \frac{\Delta p}{\lambda} - \theta p \right) - A_p' \sin \left( 4\pi f \frac{n}{f_s} - 2\pi \frac{\Delta p}{\lambda} - \theta p \right).
\]

After low pass filter, we get:

\[
Q_p = A_p' \sin \left( -2\pi \frac{\Delta p}{\lambda} - \theta p \right).
\]
Combining these two components as the real and imaginary part of a complex signal, we have the complex baseband as follows:

$$\text{BasebandSignal} = A_p e^{-i(2\pi \frac{\Delta p}{\lambda} + \theta_p)}.$$  

Note that the phase for path $p$ is $\varphi_p = (2\pi \frac{\Delta p}{\lambda} + \theta_p)$, which changes by $2\pi$ when $\Delta p$ changes by the amount of sound wavelength $\lambda = 1.72$ cm. In other words, a small movement of a few millimeters will significantly change the phase of the received sound wave.

In conclusion, if there is no hand movement at the time $t_p$, i.e., $\Delta p = 0$, then $I_p$ and $Q_p$ will be stable. Otherwise, the I/Q components vary like sinusoids.

### 3.4 System Design

We now provide the system design of Ultra-Unlock. The system is built based on Android operating system and tested by Google Nexus 6P smartphones.

In Ultra-Unlock, the smartphone speaker would send continuous ultrasound waves with frequency $f = 20,000$ Hz. The signals are encoded with 16-bit pulse-code modulation (PCM). Then we use the smartphone microphones to catch the reflective signals of the ultrasound simultaneously. Though the smartphone has more than one microphone to support stereo recording, our system only needs one channel to calculate the I/Q data. As shown in Figure 3.3, we first normalize the signal, then multiply the received signal with $\cos\left(2\pi f \frac{n}{f_s}\right)$ and $-\sin\left(2\pi f \frac{n}{f_s}\right)$. After converting the data to the fixed-point data type with a word length of 16 and a fraction length of 15, we feed the data through Cascaded Integrator-Comb (CIC) filters to remove high-frequency components and decimate the signal. To achieve better computational efficiency, we do not use a frequency compensate FIR filter after the CIC but directly output the in-phase and quadrature values.
Figure 3.3: Android App design for Ultra-Unlock
CIC filter is an optimized class of finite impulse response (FIR) filter combined with a decimator. It provides a linear phase response and utilizing only delay and addition and subtraction. In other words, it requires no multiplication operations and therefore has less computational costs, which is more suitable to be implemented on smartphones. Our CIC filter is a two-section filter (two integrator sand two comb filters) with a decimate ratio of 15 and a differential delay of 16. Figure 3.4 shows the frequency response of the CIC filter. We select the parameters so that the first and second zeros of the filter appear at 183 Hz and 366 Hz. The pass-band of the CIC filter is 0 - 100 Hz, which corresponds to the movements with a speed lower than 0.86 m/s when the wavelength is 1.72 cm.

![Magnitude Response (dB) and Phase Response](image)

**Figure 3.4:** Frequency Response of the Cascaded Integrator-Comb Filter.

As discussed in Section 3.3, the in-phase and quadrature components are a good indicator of the path changes. In detail, the I/Q waveforms remain static when the hand is not moving and vary like sinusoids when the hand moves. Different hand movements generate different sinusoids, which can be trained by a Long Short-Term Memory LSTM network for classification. We use the same LSTM network as discussed in Section 2.4.5, but with different parameter settings.
3.5 Experiment Results

We implemented the Ultra-Unlock on a Google Nexus 6P smartphone and first validate the correlation between hand movements and I/Q data. Three screenshots are shown in Figure 3.5. The yellow lines are the quadrature components and the blue lines are the in-phase components. The x-axis is the indexes of samples. In each figure, 256 samples are shown, which spans in the time period of $\frac{256 \times 44100}{4410} = 0.087$ s. (256 is the buffer size of the array plot, 294 is the sampling rate after CIC filter, 4410 is the frame size of audio recorder, and 44100 is the sampling frequency of the carrier signal.) As shown in Figure 3.5, when there is no hand movement, the I/Q data are stable. When the user puts one hand parallel to the phone and moves towards the screen, then I/Q value become regular sinusoids. If the user conducts an open-close gesture (make a loose fist and then spread all fingers wide), the I/Q data are still sinusoidal but they become less regular. Note that these screenshots cannot represent the whole changes caused by hand movement, since a whole gesture usually costs about 0.3-1 second, which spans 3 to 12 frames.

Figure 3.6 shows the I/Q data collected when the user moves the hand down and up. The phase of the I/Q data reverses at 0.24 seconds, which indicates the direction change of the hand movement. Moreover, we noticed that there are 4 full peaks and valleys before the hand changes the direction. Combining with the fact that the wavelength of the signal is 1.72 cm, we know that the path changes $1.72 \times 4 = 6.88$ cm. In other words, in 0.2 seconds, the user moves one hand towards the smartphone by 3.44 cm. Note that this distance is calculated based on the assumption that the user's hand moves in one dimension. However, in reality, the user's hands move in 3D. Fortunately, smartphones nowadays are usually equipped with more than two speakers and more than two microphones, which enable Ultra-Unlock to locate user hands in 3D. Due to the time limit, the current Ultra-UnlockApp does not have this feature yet. In future work, we will implement this feature, as more dimensions of features will also increase the accuracy of user authentication.

Our current machine learning modal only has two dimensions of features: the amplitude
and the phase calculated from I/Q data. We tested our Ultra-Unlock system on 6 people and 3 gestures. Each user is asked to perform the same gestures for 20 times where we use half of them to train the LSTM network and the other half to test the model. The three gestures are: up-down, open-close, and drawing “8” in the air as shown in Figure 3.7. The
Figure 3.6: I/Q Components from Different View Angles. The hand first move towards the smartphone, then move upwards.

6 people are 3 females and 3 males aging 20-30.

The classification results of 3 gestures are shown in Figure 3.8. The average accuracy is 90.89% and the up-down gesture has the least false negative rate. The classification results of user identification are shown in Figure 3.9, where the average accuracy is 78.47%. Note
Figure 3.7: Gestures tested in Ultra-Unlock.

<table>
<thead>
<tr>
<th>Predicted Class</th>
<th>Up-Down</th>
<th>Open-Close</th>
<th>Eight-In-The-Air</th>
</tr>
</thead>
<tbody>
<tr>
<td>True Class</td>
<td>99.8%</td>
<td>0.2%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Error Rate: 9.11%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.8: The Confusion Matrix of the Gesture Classification Result.

that the true negative rate (specificity) is 95.68% and the false positive rate is 4.32%, which means there is a higher chance that a legitimate user would be rejected than the chance that an attacker gets accepted by the phone. Though such a setting will provide more security to the smartphone, the overall accuracy is not satisfactory. There are at least three ways to further increase it, and we leave it for future work:

- Utilizing more speakers and microphones. In this chapter, we only utilizing one microphone and one speaker on the smartphone. However, most smartphones nowadays support stereo audio, which means the device is equipped with at least a
Figure 3.9: The Confusion Matrix of the User Classification Result.

pair of microphones and a pair of speakers. If all sensors and transmitters are adopted, we will increase the signal channel from $1 \times 1$ to $2 \times 2$. If doing so, Ultra-Unlock will be able to localize the hand in three dimensions and the feature vectors will increase from 2 to 8. Intuitively, the performance of the classification model will be improved with more features. However, there would also be many challenges to achieve upgrades. For example, we need to propose effective and efficient algorithms to deal with the interference caused by simultaneous signals.

- Using multiple-frequency signals as the carrier instead of single-frequency signals. The current Ultra-Unlock system suffers from the multipath effect. Ideally, when there is no hand movement, the I/Q data remain the same. However, signals received from different paths can be regarded as signals reflected by a moving hand, which causes the I/Q data to be changed too. To mitigate the multipath effect, we could use signals with multiple frequencies as the carrier. Since those signals have different frequencies
and thus have different wavelengths, they can differentiate between multiple stable paths and one varying path. The challenge is how to select those frequencies and how to set corresponding parameters for the CIC filters.

• Extracting more information from I/Q data instead of directly feeding them to LSTM networks. In this chapter, we feed the raw I/Q data to the machine learning model for classification. However, we could extract more information from them. With the aforementioned two improvement methods implemented, I/Q data can be used to reconstruct the hand trajectories in three dimensions. We can use the trajectories for user classification. Moreover, we could also extract time and frequency features from the I/Q data. Some commonly used temporal features include mean, min, max, RMS, ZCR, skewness, kurtosis, and peak count. Some commonly used frequency features include spectral centroid, spectral spread, spectral skewness, spectral kurtosis, spectral flatness, spectral irregularity, spectral entropy, spectral roll-off, spectral brightness, spectral RMS, and spectral roughness. All these features can be implemented and tested to achieve higher accuracy. In addition, Ultra-Unlock now treats the hand and the fingers as an integrated part, reconstructing the gestures from hand level to finger level may improve the gesture recognition and user authentication performance.

• Implementing the system fully on smartphones. The current Ultra-UnlockApp only serves as a data collection app, the training and predicting procedures are done in a MacBook Pro with 2.6 GHz 6-Core Intel Core i7 and 16 GB 2400 MHz DDR4. In future work, we will move this classification part to smartphones and evaluate the latency.

• Evaluating the performance of Ultra-Unlock under different settings. Many factors may have an impact on the classification result. In the current experiments, we just showed the volunteers how to perform hand movements in the air. But we didn’t restrict the users to perform the gestures in certain spaces. The volunteers were free to choose their own speeds to draw the gestures and the sizes of the gestures are flexible too. In future work, we plan to test the influence of such factors. Moreover, current
experiments are conducted in a relatively stable environment. There are no other moving objects within 1 meter of the testing smartphone. How to remove the noises from background movements is another challenge to overcome.

- Recruiting more volunteers and testing more gestures. In this chapter, Ultra-Unlock is tested over a very limited dataset. This is because of the limited time and the difficulties of recruiting volunteers during the pandemic. To popularize this new authentication method, a lot more volunteers and gestures are required.

3.6 Conclusion

In this chapter, we demonstrated how to use I/Q data to represent hand movements and show the potential of using the I/Q data for smartphone authentication. Experiments showed that the proposed Ultra-Unlock system can achieve gesture recognition with accuracy of 90.8% and user identification with accuracy of 78.47%. This Ultra-Unlock system is still at its early stage and we pointed out six directions to improve it in future work.
MOVO: A SPOOF-PROOF VOICE AUTHENTICATION SYSTEM FOR SMARTPHONES

Voice authentication is drawing increasing attention and becomes an attractive alternative to passwords for mobile authentication. However, existing voice authentication systems are vulnerable to various spoofing attacks. For example, attackers can record the victim’s voice in person or online, then replay the recording and access the victim’s devices illegally. We propose MoVo, a spoof-proof voice authentication system which not only differentiates different people but also differentiates live people and electronic devices. The idea is to utilize the self demodulation effect and acoustic attenuation effect when sound signals are transmitted through human bodies. Our system can defend the smartphone’s voice authentication system against 3 attack scenarios, each contains 3 attack types. Experiments show MoVo can identify different users with 92.98% accuracy and defend voice spoofing attacks with 90.43% accuracy.

4.1 Introduction

4.1.1 The Popularity of Voice Authentication on Smartphones

According to reports issued by several market research firms, the total number of smartphone users worldwide is over 3 billion this year and is expected to reach 3.9 billion by 2023 [79, 72]. The rapidly increasing use of smartphones is actuating the need for better protection. User authentication on smartphones has thus been an important area of research.

Survey papers [119, 66, 99] have compared the strengths and limitations of existing authentication methods, from knowledge-based methods such as PIN or password to identity-based methods such as fingerprint and face. PIN and password are the most widely used authentication methods. However, when the users have wet or dirty fingers or wear gloves on their hands, such touchscreen-related authentication methods will not work. The fingerprint authentication system suffers from the same problem. As for face recognition, it stops working when users are wearing moisturizing mask sheets or other head wearables such as ski.
goggles. In the aforementioned scenarios, voice authentication provides better convenience to users and thus a great alternative.

In fact, voice authentication has been adopted in a wide variety of smartphone applications. For example, Android users can say “Ok Google” to access Google assistant directly [30]; the Tencent company adopts Voiceprint to provide securely, faster and easier log-ins to WeChat accounts, available on both Android and iOS platforms [128]; the BioTrust uses voice biometrics to allow elderly and sick people to order prescription medication without the need to leave the house [9]; the Citi bank uses voice biometrics authentication system for phone banking, which reduces the number of tedious security questions [21]; the LMH Blockchain adopts Say-Tec [97] to authenticate, validate, process, and protect users’ blockchain assets and cryptocurrency by voice [65]. Based on a market research report published in 2019 [70], the speech and voice recognition market is expected to grow from $7.5 billion as in 2018 to $21.5 billion by 2024.

4.1.2 Voice Spoofing Attacks

In spite of the increasing trend for the adoption of voice authentication, this method is not unassailable, just like other methods. Researchers have found that voice authentication systems are vulnerable to the following four attacks [130]: impersonation attack, replay attack, speech synthesis, and voice conversion.

- **Impersonation attack** refers to the scenario where an attacker tries to mimic the legitimate user’s voice without any computer-aided technology.
- **Replay attack** refers to the scenario where an attacker replays a pre-recorded speech sample collected from the legitimate user.
- **Speech synthesis** refers to the scenario where an attacker generates intelligible, natural-sounding artificial speech from text.
- **Voice conversion** refers to the scenario where an attacker converts his speech signals to an artificial speech signal which has similar timbre and prosody to that of the legitimate user.

Among the four voice-spoofing attacks, the impersonation attack is the hardest to
perform, as Lau et al. [59] have found that successful impersonation attacks require professional impersonators or attackers whose natural voices already similar to the legitimate user’s. Even with professional mimicry artists or linguists, the existing voice authentication system is hard to fool [69].

The other three types of attacks, however, are much easier to be conducted. Because attackers could get the victim’s voice recordings. It is common to see people use voice assistants (Alexa, Bixby, Cortana, Google, Siri, etc.) in public. Attackers can also record the victim’s voice on the site or remotely. Moreover, people nowadays not only post text or image to social media sites (Facebook, Twitter, LinkedIn, YouTube, etc.), but also upload videos containing their voices. Attackers can extract the victim’s voice from those online videos and build the speech profile of each victim. With a properly built speech profile, attackers are able to use the victim’s voice to say just about anything, using algorithms such as vector quantization [1], probabilistic transform [107], or neural networks [27]. Such voice generation techniques are well-established. As evidence, celebrity voice changer websites or apps could generate natural-sounding speeches the same as those directly from Obama,Trump, Stephen Hawking, Bruce Wayne, and many more.

Some may argue that large training data is required to build the victim’s speech profile. However, for the purpose of attacking, the attackers may only need to use small training data to generate the hot-word or the activation phrase. There is no need to generate every possible sentence. This is because most voice authentication systems adopt the one-time authentication scheme to grant access, instead of a continuous authentication scheme [34].
(a) Enabling Voice setting.

(b) Voice Match warning.

(c) Information leakage when the phone is locked.

**Figure 4.1:** Screenshots about Voice Match on Google Nexus 6P.
For example, on a Google Nexus 6P smartphone running Android 8.1.0, it provides the Voice Match feature which allows users to access their personal data by voices\textsuperscript{1}. If an attacker replays the victim’s “Ok Google” utterance, followed by “show me my emails” in his own voice (not the victim’s), the Android system will regard him as the legitimate user (false acceptance) and show the emails. Because the authentication only checks the hot-word. Once the access is granted, any command coming after will be executed, no matter in whose voice.

Fig. 4.1 are the screenshots when the aforementioned replay attack is tested in reality. Fig. 4.1a shows how to enable the Voice Match function. Fig. 4.1b indicates the system is aware of its vulnerability to impersonation attack and replay attack. Fig. 4.1c demonstrates the attacker could steal sensitive information without unlocking the phone (the closed padlock at the top). In this test, the leaked information is the emails from the Gmail Inbox, which include the credit card information sent from the victim’s mom and a secret from his friend. Recall that to conduct this attack successfully, all the thing the attacker need is a short recording of “Ok Google” in the victim’s voice, but the harm can be huge.

4.1.3 Liveness Detection

Voice spoofing attacks drastically degrade the performance of standard voice authentication systems by increasing false acceptance rates [126, 32], leading to severe security and privacy issues. Fortunately, researchers have done extensive research on defending these attacks and building spoof-proof voice authentication systems.

\textsuperscript{1}When the Voice Match feature first came out, Android users could fully unlock the phone with this function. However, starts from January 2019, Google removes the ability for Voice Match to act as a password due to security concerns. The previous “Unlock with Voice Match” is replaced to only provide “personal results”. Such results come from emails, calendar entries, contacts, etc., and are still sensitive information.
<table>
<thead>
<tr>
<th>Authors</th>
<th>Method</th>
<th>Shortcoming</th>
<th>Accuracy</th>
<th>No Extra Devices</th>
<th>No Cumbersome User Interaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Girija Chetty and Michael Wagner [19]</td>
<td>Detecting lip movements using cameras.</td>
<td>Inherits shortcomings of face authentication and introduces high computational overhead.</td>
<td>99%</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Poss et al. [88]</td>
<td>Using neural tree networks to determine unique aspects of utterances and Hidden Markov Models to classify them.</td>
<td>The accuracy is unkown.</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Wei Shang and Maryhelen Stevenson [98]</td>
<td>Testing whether an incoming recording shares the same originating utterance as any of N stored recordings.</td>
<td>Performance is largely based on the pre-stored recordings.</td>
<td>88.1%/93.2%</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Jesús Villalba and Eduardo Lleida [118]</td>
<td>Detecting noises and spectrum changes caused by far-field microphone and loudspeakers.</td>
<td>Limits the replay attackers to use far-field microphones.</td>
<td>91%-100%</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Wang et al. [126]</td>
<td>Detecting channel pattern noise caused by microphone and loudspeakers.</td>
<td>Limits the replay attackers to use low-quality microphones.</td>
<td>97%</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Aley-Raz et al. [3]</td>
<td>Integrating intra-session voice variation to Nuance VocalPassword [22].</td>
<td>Requires the user to cumbersomely repeat prompted sentences.</td>
<td>-</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Zhang et al. [139]</td>
<td><strong>VoiceLive</strong>: Measuring the time-difference-of-arrival changes of a sequence of phoneme sounds to the two microphones of the phone.</td>
<td>Requires at least two high-quality microphones in one smartphone.</td>
<td>99%</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Authors</td>
<td>Method</td>
<td>Shortcoming</td>
<td>Accuracy</td>
<td>No Extra Devices</td>
<td>No Cumbersome User Interaction</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>----------</td>
<td>-----------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>Chen et al.</td>
<td>Detecting the magnetic field emitted from loudspeakers.</td>
<td>Requires the user to move the smartphone with the predefined trajectory around the sound source.</td>
<td>100%</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Zhang et al.</td>
<td><strong>VoiceGesture</strong>: Leverages Doppler shifts in signals caused by users' articulatory gestures when speaking.</td>
<td>Requires high quality microphones and needs a longer computation time.</td>
<td>99%</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Feng et al.</td>
<td><strong>VAuth</strong>: Utilizing the instantaneous consistency of the entire signal from the accelerometer and the microphone.</td>
<td>Requires the user to wear high-sampling-rate accelerometers on the facial, throat, or sternum areas.</td>
<td>97%</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Huang et al.</td>
<td><strong>BreathLive</strong>: Utilizing chest movement when making deep breaths.</td>
<td>The sound is deep breath sound instead of human speech; Stethoscope is needed.</td>
<td>91%/94%/96%</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Ment et al.</td>
<td><strong>WiVo</strong>: Using channel state information (CSI) from WiFi signals to detect mouth movement.</td>
<td>Requires WiFi antennas to collect the CSI info; the distance between antennas and human is short (20cm).</td>
<td>99%</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>
As mentioned in Section 4.1.2, the threat of impersonation attack is much less than that of the other three attacks, so impersonation attack is not a research focus. For replay attack, speech synthesis, and voice conversion, researchers have noticed that they have one thing in common – the attacking sound is played by an electronic speaker\(^2\) rather than spoken by a real person. Therefore, if the authentication system can distinguish whether the sound comes from a live person or an electronic device, it would be immune from those voice spoofing attacks.

Existing liveness detection methods can be classified into two groups: detecting human-related characteristics, or detecting device-related characteristics.

As listed in Table 4.1, there have been many research works on detecting human-related characteristics. Girija Chetty and Michael Wagner [19] use cameras to detect lip movements to detect liveness. However, their work requires camera access and inherits the shortcomings of face authentication systems. Similarly, Meng et al. [73] tries to detect mouth movement, but from channel state information from WiFi signals. Their approach requires antenna pairs and the antennas are placed very close to humans (20cm), which is not practical in reality. Poss et al. [88] use neural tree networks to determine unique aspects of utterances and hidden Markov models to classify those features. However, their work requires high computing power and long processing time. Wei Shang and Maryhelen Stevenson [98] detect liveness by testing whether an incoming recording shares the same originating utterance as any of previously-stored recordings. However, the performance of their work is largely based on the previously-stored recordings. Aley-Raz et al. [3] integrate intra-session voice variation to Nuance VocalPassword [22] for liveness detection, but they require the user to cumbersomely repeat prompted sentences. Zhang et al. [139] detect liveness by measuring the time-difference-of-arrival changes of a sequence of phoneme sounds using the two microphones of the phone, which requires at least two

\(^2\)In this and many other papers, attackers must play the attacking sound near the targeted smartphone. We do not consider the scenario where sound files are directly injected. Because if attackers can inject sounds to the system, the phone is already hacked. So there is no need to go through the voice authentication procedure anymore.
high-quality microphones in one smartphone. They also propose another work [138] to
detect users’ articulatory gestures when speaking. However, their work requires
high-quality microphones again and needs a longer computation time. Last but not least,
Feng et al. [34] utilizes the instantaneous consistency of the entire signal from the
accelerometer and the microphone for liveness detection. Their work is the most closely
related work to ours. However, their work requires the user to wear extra accelerometers
on the facial, throat, or sternum areas. Moreover, the accelerometer used in their work
requires a very high sampling rate (11,000 Hz), which cannot be supported by current
smartphones. In this work, we are only using a 400 Hz sampling rate for the accelerometer
and the gyroscope.

The aforementioned researchers detect the liveness of a user directly, but we can also
detect the liveness from the reverse side: detecting the presence of electronic devices. For
example, Jesús Villalba and Eduardo Lleida [118] detect noises and spectrum changes caused
by far-field microphones and loudspeakers. Wang et al. [126] detect channel pattern noise
calused by microphone and loudspeakers to identify replay attackers. However, they can only
deal with attackers who use low-quality microphones to record the legitimate user’s voices
or record the voice at a long distance. More recently, Chen et al. [18] detects the magnetic
field emitted from loudspeakers to identify attacks, but their work requires the user to move
the smartphone with the predefined trajectory around the sound source.

In conclusion, liveness detection methods either detect the presence of human beings
or the presence of electronic devices. Existing work has at least one of the following
shortcomings: 1) requiring special or extra devices; 2) requiring cumbersome user
interaction; 3) requiring high computing power or long processing time; 4) limited ability
in defending against spoofing attacks. Therefore, building a spoof-proof voice
authentication system is still an open problem.
4.1.4 Overview of MoVo

In this thesis, we propose MoVo, a spoof-proof voice authentication system that uses motion sensors (accelerometers and gyroscopes) to measure voice.

As shown in Fig. 4.2, the user places the smartphone horizontally and makes sure the phone is in close contact with his throat. Then the embedded motion sensors inside the phone capture the conductive vibrations from vocal organs to the throat, and to the smartphone. Afterward, the collected motion sensor data will be used for user authentication.

The intuition behind MoVo is the fact that human voice is essentially vibrations, so it can be recorded by motion sensors [47, 84, 74]. Such motion-sensor data can be regarded as downsampled microphone data, so it has the potentials to be used for voice authentication too. Moreover, since the human body is a nonlinear medium similar to water [54], sounds go through the body will be affected by acoustic attenuation [109] and self demodulation [8]. Such effects are human-only effects in that electronic devices are not water-like medium and have totally different acoustic properties. Therefore, using motion data for authentication can effectively differentiate live people from electronic devices, so that the system is protected against various voice spoofing attacks.

In fact, there have been some recent studies that show the possibility of acquiring acoustic signals by smartphones’ motion sensors. Michalevsky et al. [74] proposed Gyrophone.
in 2014. To the best of our knowledge, they are the first to use smartphone gyroscopes as low-frequency microphones to listen to loudspeakers. Gyrophone can differentiate 11 digits\(^3\) with 65% accuracy based on a 10 people dataset. One year later, Zhang et al. [137] proposed AccelWord, which utilizes accelerometers to classify hot-words such as “Okay Google” or “Hi Galaxy” over other short phrases with 85% accuracy. AccelWord is also tested over 10 people. In 2018, however, Anald and Saxena [5] reproduced the aforementioned works and overturned their conclusions. They argued that smartphone motion sensors can not be affected by the speech signals transmitted through the air, no matter the sound source is a loudspeaker or a live person. They reported that only when the speakers and the motion sensors sharing a surface, the conductive vibrations will affect motion sensors’ readings. Consistent with this newest research, MoVo asks the user to press the phone on his throat so that the body-borne vibrations are recorded, not the air-borne sounds.

In summary, compared to previous works, the MoVo system have the following features:

- **All-in-One**: MoVo is an integral method which handles user authentication and liveness detection at the same time.

- **Applicable**: MoVo works with current-off-the-shelf commercial smartphones. It does not require any extra electronic device nor any special phone model, since the sensors being used (motion sensors) are embedded on almost every smartphone.

- **Easy**: Except for pressing the smartphone on the user’s throat, MoVo does not ask users to do extra movements other than an ordinary speaking behavior.

- **Improved Robust**: General voice authentication systems are sensitive to the surrounding noises and their performance will degrade a lot in noisy environments. MoVo, however, will not be affected. This is because smartphone’s motion sensors measure the conductive vibrations and the affection from air-borne sounds is very limited [5].

- **Expandable**: MoVo currently is a text-dependent voice authentication system that detects certain hot-words. However, it can be expanded to a text-independent system.

\(^3\)One, two, \ldots, nine and zero.
since it is syllable-based (will be elaborated in Section 4.5).

4.2 Background

4.2.1 Voice Acoustics

<table>
<thead>
<tr>
<th>Device</th>
<th>Release Year</th>
<th>Microphones’ Sampling Rate</th>
<th>Motion Sensors’ Sampling Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Samsung Galaxy S8</td>
<td>2017</td>
<td>192,000 Hz</td>
<td>500 Hz</td>
</tr>
<tr>
<td>Samsung Galaxy S7</td>
<td>2016</td>
<td>192,000 Hz</td>
<td>500 Hz</td>
</tr>
<tr>
<td>Google Nexus 6P</td>
<td>2015</td>
<td>48,000 Hz</td>
<td>400 Hz</td>
</tr>
<tr>
<td>LG Nexus 4</td>
<td>2012</td>
<td>48,000 Hz</td>
<td>200 Hz</td>
</tr>
</tbody>
</table>

The generation of human voice follows a source-filter model [33]. A speech signal can be seen as a source signal (the glottal source at the larynx, or noise generated at a constriction in the vocal tract), filtered with the resonances in the cavities of the vocal tract (tongue, teeth, lips, velum, etc. modifying the sound spectrum over time). This theory has been verified using 3-D printed models of two configurations of a vocal tract to generate sounds to generate the vowels in the words “had” and “heard” [129].

A typical adult male will have a fundamental frequency \( f_0 \) of from 85 to 155 Hz, and that of a typical adult female from 165 to 255 Hz [7, 111]. The frequencies of the first, the second, and the \( i \)-th resonances are labeled as \( R_1, R_2, \ldots R_i \), and those of the spectral peaks produced by these resonances are called formants, \( F_1, F_2, \ldots F_i \) [112].

According to [58], English vowels are perceived largely according to the values of the formants \( F_1 \) and \( F_2 \). The range of \( F_1 \) is roughly from 270 to 860 Hz, and that of \( F_2 \) from 840 to 2790 Hz [86]. As for English consonants, there are six categories: plosive/stop (e.g. /p/), fricative (e.g. /f/), affricate (e.g. /dZ/), nasal (e.g. /m/), lateral (e.g. /l/), and approximant (e.g. /r/). The frequencies of consonants vary a lot. The turbulence of /s/

\(^4\)Data is partially from [71] and partially by calling the `getMinDelay()` function of `android.hardware.Sensor` class. In fact, the sensors can sample at a higher rate, but the operating systems restrict this rate in order to save power or for security concerns. For example, Google Nexus 6P uses Bosch BMI160, whose sampling rate can be 1600 Hz., but the Android operating system only supports up to 400 Hz on the phone.
and /z/ occurs above 3500Hz, and reaches as high as 10,000 Hz, whereas /w/ has $F_1$ from 250 to 450 Hz and $F_2$ from 600 to 850 Hz [57].

By Nyquist–Shannon sampling theorem, to properly sample a signal contains no frequency components higher than $f$ Hz, the sampling rate must be at least $2f$ Hz (Nyquist rate). In other words, a sampling rate of 400 Hz (motion sensors’ rate of Google Nexus 6P as shown in Table 4.2) can only handle signals whose component frequencies are below 200 Hz. Except for the part of the fundamentals, all $F_1$ and $F_2$ frequencies can not be sensed. Therefore, it is impossible to perceive the signals with such a low sampling rate.

Fortunately, the objective of using motion data in MoVo is liveness detection and user identification, not signal recovery. With some proper machine learning technology, the undersampled data is informative enough to fulfill the purpose. The reason is, in signal processing, there exists the aliasing phenomenon that high-frequency data will have aliases at the low-frequency range, which indicates that the information is kept, though distorted.

4.2.2 Self Demodulation

Motion sensors not only capture the original sound data but also capture the modulated signals. In detail, with self demodulation [8], the original sounds self interacts inside the human body, resulting in sounds with lower frequency.

\[
\begin{align*}
fa & \rightarrow f_a + f_b \\
fb & \rightarrow f_b \\
fa & \rightarrow f_a \\
fb & \rightarrow f_b \\
|f_a - f_b| & \rightarrow |f_a - f_b|
\end{align*}
\]

**Figure 4.3:** Self Demodulation of Sound Signals When Transmitting Through the Human Body.

Researchers have found that sounds with different frequencies that transmitted through a nonlinear medium would interact with each other [87]. This interaction produces new frequencies upon the combination of the sums and differences of the individual frequency components by Khokhlov-Zabolotskaya-Kuznetsov(KZK) parabolic nonlinear
wave equation [81].

Since the acoustic impedance of the human body is similar to that of water [54], the self-demodulation would occur in the human body as shown in Fig. 4.3. The original sound signals with frequency $f_a$ and $f_b$ would introduce two more signals with frequency $f_a + f_b$ and $|f_a - f_b|$. For different person, the original signals generated have different frequency, so the low frequency signal $|f_a - f_b|$ are different, which can be utilized for user authentication. Moreover, note that electronic devices have different acoustic properties from that of the human body. Therefore, those low-frequency signals can be used for liveness detection.

### 4.2.3 Acoustic Attenuation

Another effect that helps MoVo to do spoof-proof authentication is the acoustic attenuation by the human body. It is known that the human voice is emitted by the vocal organ and is a combination of mechanical vibrations with multiple amplitudes and different frequencies. When a person speaks, the airflow from the lungs through the trachea compresses the vocal cords causing vibrations to make sounds. The lung, trachea, and vocal cord form a resonance chamber.

Suppose the length of vocal cords is $d$, the lung volume is $V_0$, and the cross-sectional area at the vocal cords is $S$. According to the polytropic process equation, when the airflow moves $d$, the air pressure at the vocal cords can be expressed as follows,

$$P_1 = \frac{P_0 \cdot V_0^\gamma}{(V_0 - d \cdot S)^\gamma},$$

where $P_0$ is the normal atmospheric pressure, and $\gamma$ is a coefficient about the air specific heat. According to the definition of pressure, if the area at the vocal cords is $S_v$, the force at the vocal cord is,

$$F_0 = P_1 \cdot S_v = \frac{S_v \cdot P_0 \cdot V_0^\gamma}{(V_0 - d \cdot S)^\gamma}.$$

When the force is applied to the vocal cords, vertical displacement occurs. According to the Newton’s second law of motion, we have,

$$F(t) = ma(t) + kx(t) + cv(t),$$
where $F(t)$ is the external force, $v(t)$ is the speed, $x(t)$ is the vertical displacement, $c$ is the damping coefficient, and $k$ is the spring constant and $m$ is the mass. The relation can further be explained as,

\[ \dot{F}(t) = m \frac{d^2 x(t)}{dt^2} + kx(t) + c \frac{x(t)}{dt}. \]  

(4.1)

The vibration during an airflow pass the vocal cords can be separated into two phases. In the first phase, the airflow is passing the vocal cords which is considered to be a forced vibration with constant force $F_0$. After the airflow passed, in the second phase, the pressure of airflow disappears which leaves the system to vibrate on its own and this is called free vibration. In the forced vibration phase, after applying the Fourier transform to both side of e.q. (4.1), we have,

\[ \frac{F_0}{j \omega} (1 - e^{-j \omega \delta t}) = -\omega^2 m X(\omega) + kX(\omega) + j \omega c X(\omega). \]

That is,

\[ X(\omega) = \frac{1 - e^{-j \omega \delta t}}{-\frac{2m}{F_0} \omega^3 - \frac{c}{F_0} \omega^2 + \frac{jk}{F_0} \omega}, \]

where $X(\omega)$ is the spectrum of the vertical vibration signal and $\omega$ is the frequency. During the horizontal propagation of the vibration signal from the vocal cords to the throat, the vibration suffers from attenuation, and the corresponding model can be stated as follows,

\[ x_s(t) = x(t) e^{-\alpha d}, \]

where $x_s(t)$ is the vertical displacement at the throat where the vibration has propagated, $x(t)$ is the vertical displacement at the vocal cords, $d$ is the propagation distance, and $\alpha$ is the attenuation coefficient. After applying the Fourier transform to both side of e.q. (4.1), we have,

\[ X_s(\omega) = X(\omega) e^{-\alpha d}. \]

Note that $\alpha$ is related to the propagation medium. Wave propagation in the body is dispersive by nature, which implies that different frequencies propagate with different attenuation coefficients at different velocities. Roughly speaking, the attenuation is small
when the vibration signal propagates through the hard bone, whereas the attenuation is large through the soft tissue. Therefore, vibration waves generated at different positions at throat result in different values of $\alpha$ and $d$, which make the vibration signals unique at different positions. After putting all equations together, we obtain,

$$X_s(\omega) = \frac{(1 - e^{-j\omega t})e^{-\alpha d}}{(-jm\omega^3 - c\omega^2 + jk\omega)((\frac{V_0 - dS}{S_\infty P_0 V_0})^\gamma)}.$$ 

For the same location of the human body, $m$, $c$ and $k$ are stable and belong to the same biometric feature. Each person’s lung volume and vocal cords are also different. Therefore, the vibration at the throat of different people can uniquely be identified, which can be leveraged for authentication. The propagation from the electronic device to the target smartphone is different from that from the vocal organ through the human body. Thus, this effect is also valuable for liveness detection.

### 4.3 Proof-of-Concept

We test the feasibility of MoVo and the results are shown from Fig. 4.4 to Fig. 4.7. In each figure, we show both the raw signal and the spectrogram for the microphone data and the motion sensor data. All data are collected by Google Nexus 6P. The audio data is sampled at 8,000 Hz (telephone quality) while the motion data is sampled at 400 Hz as it is the highest sampling rate on Nexus 6P.

Fig. 4.4, Fig. 4.5 and Fig. 4.6, show the example data when the same user speaks the same command “Ok Google”, different users speak the same command “Ok Google”, and the same user speaks different commands “Ok Google” and “Hi Siri”, respectively. The data are collected as in Fig 4.2c. In each figure, the top subfigure (a) is the raw microphone data; the subfigure (b) contains the 3-axis accelerometers data and 3-axis gyroscopes data; the subfigure (c) shows the frequency-domain information of raw audio data while the subfigure (d) show that of raw motion data. In subfigure (d), we only choose acc-z data to draw since it is the most representative one. The vertical red lines demonstrate the start and end points of the sounding period.
Figure 4.4: One user speaks “Ok Google” twice.
Figure 4.5: Different users both speak “Ok Google”.

(a) Raw Audio Data.

(b) Raw Motion Data.

(c) Spectrogram of Raw Audio Signals.

(d) Spectrogram of Raw Motion Signals.
Figure 4.6: One user speaks “Ok Google” and “Hi Siri”.

(a) Raw Audio Data.
(b) Raw Motion Data.
(c) Spectrogram of Raw Audio Signals.
(d) Spectrogram of Raw Motion Signals.
Figure 4.7: Live user speaks “Ok Google” once, then replay the recording by an electronic device.
From these three figures, we can observe that the motion data are nosier and contain much fewer data and less representative than audio data, which indicates the challenge of designing MoVo. Fortunately, the results meet our expectations. Fig. 4.4 shows the consistency when the same user speaks the same command and Fig. 4.5 shows the difference when different users speak the same command. Such intra-class similarities and inter-class differences indicate the feasibility of using motion data for user authentication. Moreover, different users have similar raw audio spectrogram (Fig. 4.5c) but different raw motion spectrogram (Fig. 4.5d), which is an evidence of different acoustic attenuation effect of different people. Note that Fig. 4.5d shows the spectrograms are similar when one user speaks different commands. Such observations indicate that frequency-domain data are not of much use to match between motion data and the same commands. Therefore, we adopt a Long Short-Term Memory (LSTM) network, a variant of the Recurrent Neural Network (RNN), to learn the patterns of motion data in time-domain.

Fig. 4.7 shows how MoVo can be spoof-proof. During the test, the user speaks “Ok Google” once, and two smartphones (one is Google Nexus 6P and the other is iPhone XS Max) record his voice. After the user finishes speaking, the iPhone XS Max replays the recordings to Nexus 6P. The replay volume is set to be the maximum possible and the two smartphones are physically contacted. The data in Fig. 4.7 are the readings from Nexus 6P, which contains the live user’s voice followed by the iPhone replayed voice. We observe that motion data for live person shows noticeable signals from 50 Hz to 200 Hz while motion data for the electronic device shows only noises as in Fig. 4.7d, which is an evidence of that the self demodulation effect of the human body generates more low-frequency signals (compared to original sound signals, the frequency of 50-200 Hz signals are low). Note that there exists a clicking noise at the time around 6.8 s, which is the time of clicking the button on the iPhone to replay the voice. The iPhone is in close contact with the Nexus 6P, therefore the power of this clicking noise is very high.
Figure 4.8: Attack Model: There are three types of attack scenarios. To conduct a simple playback attack, the target phone is placed in contact with the electronic speaker. To conduct a mimicry attack, the target phone is placed on the attacker’s throat, but the attacker will not speak during the authentication period. As for a sophisticated mimicry attack, the attacker would try to mimic the victim’s voice while playing the victim’s sounds through electronic speakers. In the two mimicry attacking scenarios, the target phone is also in contact with the electronic speaker. In all three scenarios, the sound played by the electronic speaker could be the pre-recorded sound from the legitimate user, synthesized sound, or converted sound.

4.4 Attack Model

As mentioned in Section 4.1.2, traditional attacks to voice authentication are impersonation attacks, replay attacks, speech synthesis attacks, and voice conversion attacks. Real person impersonation attacks can be effectively defended by existing speaker recognition algorithms in general. For the other three, attackers acquire the legitimate user’s voice samples either in place or online. The attacker then processes the samples in three ways to perform attacks: replay attack - concatenate voice segments to match the legitimate user’s passphrase followed by harmful action or commands; speech synthesis - build speaker model and synthesize passphrase from texts; voice conversion - the attacker says the passphrase, then by spectral mapping and prosody conversion, the signals are manipulated to sound like the legitimate user’s.

If the attacker directly plays the processed sound signals through the speaker, he is conducting the simple playback attack, no matter the processed signals are pre-recorded (replay attack), synthesized (speech synthesis), or converted (voice conversion).

A stronger attacker would perform the mimicry attack, where the processed signals...
are still played by electronic speakers, but the attacker also needs to place the smartphone on his throat. In this case, the attacker can observe how the legitimate user passes the authentication\textsuperscript{5} and try to mimic the throat movement of the legitimate user. Note that we do not consider that the attacker uses the built-in speaker of the smartphone to play the processed signal, because if the attacker can control the built-in speaker, he has already hacked the targeted smartphone, which means there is no use to do voice authentication anymore. Therefore, the speaker, shown as yellow icons in Fig. 4.8 must be a different device from the smartphone.

The strongest attack is the \textit{sophisticated mimicry attack}, where the attacker would not only mimic the sound by the electronic device, but also by himself. Compared to the previous case, the attacker would speak the same word along while the electronic speaker is playing. Recall that impersonate the victim’s voice using vocal organs is very hard, the attacker should either be professional impersonators or have natural voices similar to the victim’s. Such a condition is hard to be met. Therefore, in this thesis, the sophisticated mimicry attacker only controls the timing (starting or pausing when speaking), but not timbre.

Note that we consider two mimicry attacks because these two have different emphasis: the mimicry attack makes sure the motion sensor data are affected by the victim’s sound, while the sophisticated mimicry attack tries to add “liveness” to the motion sensor data.

4.5 System Design

4.5.1 System Overview

MoVo currently is a text-dependent voice authentication system. In other words, the speaker recognition algorithm will work on hot-words such as “Ok Google”, “Hi Siri”, or “Alexa”. When a user triggers the voice authentication system, the microphone works normally and the motion sensors measure the modulated and attenuated sound signals at the same time.

\textsuperscript{5}MoVo is a spoof-proof voice authentication system and require users to place the phone on the throat. For best user experience, it can downgrade to a normal voice authentication system and only process microphone data for normal use. Only when the user accesses sensitive information or makes dangerous operation, the spoof-proof mechanism will be invoked and the motion sensor would be in use.
Since the hot-words are usually short, less than 2 seconds in our experiment, it is acceptable to process the data together after the whole hot-word is spoken.

We first conduct the syllable separation on speech signals. For example, “Ok Google” has 4 syllables in total: O-K-Goo-Gle, “Hi Siri” has 3, and “Alexa” has 3 too. We will, later on, use the detected hot-word beginning and ending time, as well as the syllable nuclei time to segment the accelerometer and gyroscope data. Since the motion data suffer from noises from body movements, heartbeats, and breathings, we must preprocess the motion data. We apply a high pass filter to mitigate the noises and increase the target signal.

We then segment the motion data based on different syllables. We focus on data collected among the syllable nucleus. Because when the sound signal is the maximum, the chance is high that the motion is also the maximum. The maximum motion indicates a higher accuracy of the collected data, which is beneficial for training a more effective and efficient classification model. Another benefit of the segmentation is that segmentation provides the opportunity to do majority voting. For the motion data corresponding to “Ok, Google”, as long as more than half of the samples are classified into the correct category, we regard the speech and throat movement as matching each other. This greatly increases the true positive rate of our liveness detection algorithm.

Due to the low sampling rate of motion sensors, the on-body vibration of speech cannot be fully recorded in motion sensors. Finding good representative features is hard. Therefore, we adopt the long short-term memory (LSTM) network, a variant of the recurrent neural
Figure 4.10: Syllable Separation. The original signal is a female user saying “Ok Google”; then the signal is filtered with a bandpass filter with passing frequency range [50 1000]. The black dots in subfigure a) are the calculated pitches. The vertical red dotted lines indicate the start and end of a sounding period. The blue vertical lines show the calculated time for each syllable nucleus.

After the majority voting on the classification results on all data samples, MoVo outputs whether the motion data match the live legitimate user’s training model. If yes, a live user is asking the permission; otherwise, an attacker is using a speaker to attack the system.

4.5.2 Syllable Separation

A syllable is a unit of pronunciation having one vowel sound, with or without surrounding consonants, forming the whole or a part of a word [89]. The vowel in the middle of a syllable is referred to as a nucleus in phonetics and phonology. We modify the syllable nuclei detection algorithm proposed by De and Wempe [26] and the detailed steps are listed as the following:

**Step 1.** Before conducting the syllable separation, we first apply a [50 – 1000] Hz bandpass filter to remove noises so that the frequency range is speech-band limited. We
then calculate the intensity and pitch to detect the syllable nucleus since the vowel within a syllable has higher energy surrounding sounds.

**Step 2.** The intensity of a sound in air is the sound pressure level relative to $2 \times 10^{-5}$ Pascal, which is the normative auditory threshold for a 1000-Hz sine wave. We calculate the intensity contour by squaring all values in the sound, then convolved with a Kaiser window. To guarantee that a periodic signal is analyzed as having a pitch-synchronous intensity ripple not greater than 0.00001 dB, we set the length of the Kaiser window to be 64 ms and the sidelobe height to be -190 dB. In this way, we are able to find peaks in the energy contour.

**Step 3.** We consider all peaks above a certain threshold in intensity to be potential syllables. We set the threshold to 20 dB below the maximum intensity measured over the total sound file.

**Step 4.** We then use the intensity contour to make sure that the intensity between the current peak and the preceding peak is sufficiently low. We consider only a peak with a preceding dip of at least 2 dB with respect to the current peak as a potential syllable. In this way, we also delete multiple peaks within one syllable.

**Step 5.** We use the algorithm proposed by Boersma {boersma1993accurate} to calculate the pitch (fundamental frequency) contour of audio data. The window size is set to be 100 ms with 20 ms time steps. We then exclude all peaks that are unvoiced. The remaining peaks are considered syllable nuclei and will be used to segment motion data.

Fig. 4.10 shows the pitch contour in subfigure a) and the intensity contour in subfigure b). The resulting appearance times of syllable nuclei are marked using blue vertical lines.

### 4.5.3 Preprocessing and Segmentation

The motion data is impeded by the low sampling rate, low target movement, and large interference noises. To overcome such a problem, we must preprocess the motion data. We apply a high pass filter to mitigate the noise and increase the signal-to-noise ratio. The cutoff frequency is set to be 100 Hz, since noises such as breathing or walking or other human movements can not create signals as high as 100 Hz. As shown in Fig. 4.11, after
applying a high pass filter, the motion data in Fig. 4.11b is much cleaner than the original data in Fig. 4.11a.

We then use the syllable nuclei calculated in the previous section to segment the motion data. As shown in Fig. 4.11c, we first calculate the half time points (green lines) from sounding start time (red line), syllable times (blue lines), and sound end time (red line). Then we extract each segmentation from two adjacent half time points (green lines). Note that if the time duration between two adjacent half time points is large than 100 samples, we will only keep the middle 100-samples data and discard the data at the beginning area and the end area. This is because the data far from syllable nuclei are not as reliable as data around syllable nuclei. Keeping those unreliable data does no good for the classification model. Lastly, concatenating reliable data around syllable nuclei together gives the final processed data. One example of the resulting segmentation is illustrated in Fig. 4.11d. Note that Fig. 4.11c shows motion data of all six dimensions, while Fig. 4.11d only shows two of
them: the data of the solid line is from the gyroscope and the data of the dashed line is from the accelerometer.

Note that we do not consider the synchronization problem between the microphone and the motion sensors. This is because the sampling rate of motion sensors is set to be 400Hz, which means that an error with just one sample represents $\frac{1}{400} = 0.0025$ s. Within such a period of time, the sound travels about $343 \text{ m/s} \times 0.0025 \text{ s} \approx 0.85 \text{ m}$ in air, which is much longer than the distance between the voice source and the microphone. In other words, due to the low sampling rate of motion sensors, the true lag between microphone reading and motion reading always falls in one sample period, which makes the synchronization procedure unnecessary.

4.5.4 The LSTM Network

After the preprocessing and segmentation of the motion data, we use the data to establish a sequence-to-sequence Long Short-Term Memory (LSTM) network model. LSTM was first proposed by Sepp Hochreiter and Jürgen Schmidhuber in 1997 [45]. It is a special variant of Recurrent Neural Networks (RNN), and is widely used in learning, processing, and classifying sequential data because of its great property of selectively remembering patterns for long durations of time. Over the years, there have also been many variants of LSTM networks. However, based on a study in 2017, none of the variants can improve upon the standard LSTM architecture significantly [38]. Therefore, we still choose to implement the standard LSTM network in this work.

Our sequence-to-sequence LSTM has five layers in total and it is able to make different predictions for each individual time step of the input data. In the sequence input layer, the input data have 6 feature dimensions, which consists of 3 accelerometer dimensions and 3 gyroscope dimensions. Then we establish an LSTM layer formed by LSTM blocks, where each block publishes its cell state to the next LSTM block. The output of the LSTM layer is the fully connected hidden status layer. We set the total number of hidden units to be 100, then feed the hidden status to a softmax function and output the classification of each
**Figure 4.12:** Classification result without syllable separation and majority voting: Falsely classifying an ‘Ok Google’ sample to ‘Hi Siri’.

**Figure 4.13:** Classification result from the sequence-to-sequence LSTM network. Although many parts of the classification is incorrect, with majority voting, the final classification is the correct ‘Ok Google’.

4.6 Majority Voting

Since we are using the sequence-to-sequence LSTM network, an example classification result is shown in Fig. 4.13. Though the ground truth of the test data is O-K-Goo-Gle, the predicted result is O-Hi-O-K-O-Gle-O. However, with a majority voting algorithm, as long as half of the sample falls in category ‘O’, ‘K’, ‘Goo’, and ‘Gle’, we will regard the whole input data as in category ‘Ok Google’. The principle behind this majority voting is the consistency...
of the throat movement when speaking different syllables of one single command. In addition, since the syllable segmentation algorithm is heuristic, its uncertainty in separating syllables also increases the demand for adopting majority voting to compensate for the uncertainty. In all, adopting syllable separation and majority voting can greatly increase the true positive rate of MoVo.

**Remark.** In our experiment, we only train our model with 10 syllables since our system is designed as a text-dependent voice authentication system. However, with a larger training database, we can build a model with more syllables, and extend our system to work for text-independent systems. According to [82], 322 syllables can form 5000 most frequent English words. With such an extension, MoVo can also become a continuous voice authentication system.

### 4.7 Implementation and Evaluation

**Phones and Placements.** We use Huawei Nexus 6P Android smartphone to collect user data. Since we mainly use the microphone data to detect the syllable nuclei time, we do not require a high sampling frequency of audio data. Indeed, we only record the data at 8000 Hz (telephone quality). For the motion data, however, the sampling frequency is the higher the better. The Nexus 6P is manufactured in 2015, but we have updated its operating system to Android Oreo (API level 26), which is released in 2017. By calling the `getMinDelay()` function, we found the minimum delay allowed between two motion sensor events is 2500 microsecond, which is a sampling frequency of 400 Hz. Therefore, we use 400 Hz for both gyroscope and accelerometer. As shown in Fig. 4.2, a smartphone user places his device to his throat tightly so that conductive vibrations can be measured. The data collection app is shown in 4.14a.
Data Collection. Our experiment involves 20 participants aged from 20 to 35. Among them, 13 are males and 7 are females; 15 are native English speakers and 5 uses English as a second language. For each user, we ask them to speak the following three hot-words:
“Ok Google”, “Hi Siri”, and “Alexa”. For each command, each user repeats it for 5 times. Therefore, we have 300 command samples in total. When we train our LSTM network, we are using the segmented motion data to train 10 different categories (‘O’, ‘K’, ‘Goo’, ‘Gle’, ‘Hi’, ‘Si’, ‘Ri’, ‘A’, ‘Le’, ‘Xa’). In this respect, we have 3000 sample sequences, where each sequence is about 100 samples long.

Figure 4.15: Attacker Settings
Attacks. As elaborated in Section 4.4, we evaluate our system against three types of attack scenarios, simple playback attack, mimicry attack, and sophisticated mimicry attack, where each scenario contains three kinds of attacks. Since speech synthesis and voice conversion will generate speech signals similarly if use the same user’s speech profile. Therefore, we only test the replay attack.

Accuracy: 90.43%

<table>
<thead>
<tr>
<th>True Class</th>
<th>Accuracy</th>
<th>Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>91.8%</td>
<td>8.2%</td>
</tr>
<tr>
<td>Device</td>
<td>86.1%</td>
<td>13.9%</td>
</tr>
<tr>
<td>N/A</td>
<td>94.3%</td>
<td>5.7%</td>
</tr>
</tbody>
</table>

Figure 4.16: Success rate of MoVo on defending against various attacks.

In simple playback attacks, the recordings of the legitimate user are replayed by either Logitech S120 2.0 Stereo Speakers, the built-in speakers of Apple Macbook Pro, and the built-in speaks of an Apple iPhone XS Max. Each of the 20 participants is considered as a legitimate user separately. For each participant, 5 attacks are conducted by the loudspeaker, 5 attacks are conducted by the laptop speaker, and 5 attacks are conducted by the iPhone speaker. All three hot-words are tested. In total, there are 900 attacks. The attacking target is always the Nexus 6P. The replay sound level is about 80dB, which is consistent with the decibel level of normal human speech. The attacking device and the attacking target are contacted, so conductive vibrations are measured. In mimicry attacks, two smartphones are
placed together and attached to the human throat. In sophisticated mimicry attacks, we consider 2 attackers: one male attacker to mimic the 13 male participants, and the other female attacker to mimic the 7 female participants. The attacker holds the phone tightly to his/her throat while replaying the target user’s voice commands as in the simple playback attack cases. The sophisticated mimicry attack is repeated 5 times for each hot-word for each participant. Again, there are 300 such attacks. The final results are shown in Fig. 4.16, MoVo can defend replay attack with at least 90.43% accuracy.

We have also implemented an application on Android to run Ultra-Unlock in real-time. As shown in Fig. 4.14, the MoVoApp will decide among the following three cases: 1) a real human is speaking, 2) an electronic device is replaying human speeches, 3) no human sounds are made. Currently, this app only serves as an liveness detection app. We plan to add the user authentication features to this app in future work.

![Confusion Matrix](image)

**Figure 4.17:** Confusion Matrix of Matching Motion Data to Different Hot-Words.

**Results.** Besides defending against various attacks, MoVo should accept legitimate users as in normal voice authentication systems. In other words, MoVo should correctly
classify a legitimate user’s motion data to the hot-words he says. As shown in Fig. 4.17, the overall accuracy of correct classification is 93.67%. Note that we have two confusion matrices. Figure 4.17a is the original results provided by the machine learning network, while Figure 4.17b is the result with the presence of the majority voting procedure. There is a significant accuracy improvement with the presence of majority voting. Therefore, we only show the statistic evaluations of Figure 4.17b.

The results for user authentication are shown in Fig. 4.19 and Table. 4.3. Without majority voting, the accuracy of user authentication is only 54.48%. With majority voting, the accuracy increases to 92.98%. Note that this accuracy cannot beat existing speaker recognition systems based on audio files. Therefore, for better security, this result can serve as an extra channel of information.

We also test the robustness of MoVo in Fig. 4.18. We test both the one-time trained model and the learning model which will use the accepted data as trained data for future

<table>
<thead>
<tr>
<th>Table 4.3: Statistical Analysis of the User Classification Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy: 92.98%</td>
</tr>
<tr>
<td>Precision: 93.33%</td>
</tr>
<tr>
<td>$F_1$ Score: 0.924</td>
</tr>
<tr>
<td>False Negative Rate: 5.42%</td>
</tr>
</tbody>
</table>

![Figure 4.18: Robustness of MoVo over time.](image)

The results for user authentication are shown in Fig. 4.19 and Table. 4.3. Without majority voting, the accuracy of user authentication is only 54.48%. With majority voting, the accuracy increases to 92.98%. Note that this accuracy cannot beat existing speaker recognition systems based on audio files. Therefore, for better security, this result can serve as an extra channel of information.

We also test the robustness of MoVo in Fig. 4.18. We test both the one-time trained model and the learning model which will use the accepted data as trained data for future
authentication. Over 8 weeks, the learning model has a more stable performance.

4.8 Conclusion

Self demodulation and acoustic attenuation can be used to build MoVo, a spoof-proof voice authentication system. When a user speaks with the smartphone placed on his throat, his voice not only influences the microphone readings, but also affects the accelerometer and gyroscopes. By adopting a sequence-to-sequence long short-term memory network, syllable separation, and majority voting, MoVo can defend against 3 different types of attacks with 90.43% defend rate and a 92.98% acceptance rate for legitimate users.
(a) Without Majority Voting

(b) With Majority Voting

**Figure 4.19:** Confusion Matrix of Matching Motion Data to Different Users.
CHAPTER 5

CONCLUSION

In this thesis, we studied three privacy and security problems on smartphones. We first uncovered a new stealth attack named the Man-in-the-Phone attack that eavesdrops on smartphones’ built-in speakers by the intra-device motion sensors. The attack is implemented in Spy-Phone system utilizing speaker-independent machine learning, which makes the attack more dangerous and harmful. We also provided hardware-based defenses and software-based defenses for this attack, but this attack is still a threat for smartphone users and requires user awareness. We then introduced two different authentication methods. The Ultra-Unlock system enables users to unlock the smartphone with gestures in the air. It is a good alternative to existing authentication methods. Moreover, the same technique can be used for smartphone control, which allows users to unlock and control the phone without touching the screen. The MoVo system is a patch for the current voice authentication mechanism. It defends smartphones against various voice-spoofing attacks, especially the replay attack. The three systems either propose new S&P problems on smartphones, or solve existing S&P problems with novel approaches. However, we have also noticed the limitations of the current work and will continue to improve it in future work.
REFERENCES


[40] Jiawei Han, Yanheng Liu, Xin Sun, and Lijun Song. Enhancing data and privacy security in mobile cloud computing through quantum cryptography. In *2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pages 398–401. IEEE, 2016.


[56] Andrew Kwong, Wenyuan Xu, and Kevin Fu. Hard drive of hearing: Disks that eavesdrop with a synthesized microphone. In Hard Drive of Hearing: Disks that Eavesdrop with a Synthesized Microphone, page 0. IEEE.


[85] Payal Patel, Rajan Patel, and Nimisha Patel. Integrated ecc and blowfish for


[114] Joel A Tropp and Anna C Gilbert. Signal recovery from random measurements via


