SECURITY AND PRIVACY ISSUES OF MOBILE CYBER-PHYSICAL SYSTEMS

A Dissertation
Submitted to
the Temple University Graduate Board

in Partial Fulfillment
of the Requirements for the Degree of
DOCTOR OF PHILOSOPHY

by
Shang, Jiacheng
August, 2020

Examining Committee Members:

Dr. Jie Wu, Dept. of Computer and Information Sciences (Advisor)
Dr. Chiu C. Tan, Dept. of Computer and Information Sciences (Committee chair)
Dr. Si Chen, Dept. of Computer Sciences, West Chester University of Pennsylvania
Dr. Yan Wang, Dept. of Computer and Information Sciences
Dr. Fauzia Ahmad, Dept. of Electrical and Computer Engineering (External reader)
Cyber-physical systems (CPS) refer to a group of systems that combine the real physical world with cyber components. Traditionally, the applications of CPS in research and the real world mainly include smart power grid, autonomous automobile systems, and robotics systems. In recent years, due to the fast development of pervasive computing, sensor manufacturing, and artificial intelligence technologies, mobile cyber-physical systems that extend the application domains of traditional cyber-physical systems have become increasingly popular. In mobile cyber-physical systems, devices have rich features, such as significant computational resources, multiple communication radios, various sensor modules, and high-level programming languages. These features enable us to build more powerful and convenient applications and systems for mobile users. At the same time, such information can also be leveraged by attackers to design new types of attacks. The security and privacy issues can exist in any application of mobile CPS. In terms of defense systems, we focus on three important topics: voice liveness detection, face forgery detection, and securing PIN-based authentication. In terms of attack systems, we study the location privacy in augmented reality (AR) applications.

We first investigate the voice replay attacks on smartphones. Voice input is becoming an important interface on smartphones since it can provide better user experience compared with traditional typing-based input methods. However, because the human voice is often exposed to the public, attackers can easily steal victims’ voices and replay it to victims’ devices to issue malicious commands. To defend the smartphone from voice replay attacks, we propose a novel liveness detection
system, which can determine whether the incoming voice is from a live person or a loudspeaker. The key idea is that voices are produced and finalized at multiple positions in human vocal systems, while the audio signals from loudspeakers are from one position. By using two microphones on the smartphone to record the voice at two positions and measure their relationship, the proposed system can defend against voice replay attacks with a high success rate.

Besides smartphones, voice replay attacks are also feasible on AR headsets. However, due to the special hardware positions, the current voice liveness detection system designed for smartphones cannot be deployed on AR headsets. To address this issue, we propose a novel voice liveness detection system for AR headsets. The key insight is that the human voice can propagate through the internal body. By attaching a contact microphone around the user’s temple, we can collect the internal body voice. A voice is determined from a live person as long as the collected internal body voice has a strong relationship with the mouth voice. Since the contact microphone is cheap, tiny, and thin, it can be embedded in current AR headsets with minimal additional cost.

Next, we propose a system to detect the fake face in real-time video chat. Recent developments in deep learning-based forgery techniques largely improved the ability of forgery attackers. With the help of face reenactment techniques, attackers can transfer their facial expressions to another person’s face to create fake facial videos in real-time with very high quality. In our system, we find that the face of a live person can reflect the screen light, and this reflected light can be captured by the web camera. Moreover, current face forgery techniques cannot generate such light change with acceptable quality. Therefore, we can measure the correlation and similarity of the luminance changes between the screen light and the face-reflected light to detect the liveness of the face.
We also study to leverage IoT devices to enhance the privacy of some daily operations. We find that the widely used personal identification number (PIN) is not secure and can be attacked in many ways. In some scenarios, it is hard to prevent attackers from obtaining the victim’s PIN. Therefore, we propose a novel system to secure the PIN input procedure even if the victim’s PIN has been leaked. The basic idea is that different people have different PIN input behavior even for the same PIN. Even though attackers can monitor the victim’s PIN input behaviors and imitate it afterward, the biological differences among each person’s hands still exist and can be used to differentiate them. To capture both PIN input behavior and the biological features, we install a tiny light sensor at the center of the PIN pad to transfer the information into a light signal. By extracting useful features from multiple domains, we can determine whether the PIN input is from the same person with high accuracy.

Besides designing new defense systems, we also show that sensory data and side-channel information can be leveraged to launch new types of attacks. We conduct a study on the network traffic of location-based AR applications. We find that it is feasible to infer the real-time location of a user using the short-time network traffic if the downloading jobs are related to the current location. By carefully deploying fake AR contents at some locations, our attack system can infer the location of the user with high accuracy by processing noisy network traffic data.

**Keywords:** Attack, privacy, security, mobile cyber-physical system.
To my beloved parents and wife
ACKNOWLEDGEMENTS

The past five years in the Center for Networked Computing (CNC) of Temple University CIS department will always be one of the my most important experiences. During these five years, I have benefited a lot from brilliant researchers and collaborations at Temple University. With these experiences, I have established a serious attitude towards research and also developed the necessary abilities to be a good researcher.

First, I would like to express my sincere appreciation to my advisor, Dr. Jie Wu. I cannot accomplish as much as I have without his selfless support, strict guidance, encouragement, and patience. During my Ph.D. progress, he has taught me how to think creatively and critically, provided me with great support, and shown me the necessity of a serious attitude towards research, which I will never forget. I also would like to thank Dr. Si Chen for his kindness, patience, and helpful guidance on doing system-based research.

I am also very grateful to my committee members Dr. Chiu C. Tan, Dr. Si Chen, Dr. Yan Wang, and Dr. Fauzia Ahmad. They have offered me sincere help, extensive knowledge, and valuable comments to my research from the preliminary examination to the dissertation defense.

It is my fortune to be the teaching assistant of Dr. Eugene Kwatny and Prof. Claudia Pine-Simon. They have offered me selfless support and shown me how to be a good teacher.

Finally, I would like to give my deepest and heartfelt thanks to my beloved parents and my better half. Thank you for letting me smile even if I think I cannot, for hugging me when I lose my voice, and for giving me unconditional trust. You are the reason
for all the successes in my life, and you mean more to me than you’ll ever know.
# TABLE OF CONTENTS

ABSTRACT ................................................................. ii  
DEDICATION ............................................................. v  
ACKNOWLEDGEMENTS .................................................. vi  
LIST OF FIGURES ........................................................ xiii 
LIST OF TABLES .......................................................... xvii 

CHAPTER

1. INTRODUCTION ....................................................... 1  
1.1 Motivation and challenges .......................................... 1  
1.2 Major contributions ................................................ 5  
1.3 Dissertation overview .............................................. 7  

2. DEFENSE SYSTEMS 1: VOICE LIVENESS DETECTION  
ON SMARTPHONE ....................................................... 9 

2.1 Introduction ......................................................... 9  
2.2 Literature review ................................................... 12  
  2.2.1 Voice-based Mobile Applications ......................... 12  
  2.2.2 Automatic Speaker Verification System .................. 12  
  2.2.3 Voice-Spoofing Attacks .................................... 13  
2.3 Preliminary ........................................................... 14  
  2.3.1 Attack model .................................................. 14  
  2.3.2 Background knowledge .................................... 14  
  2.3.3 Key insights ................................................ 16  
  2.3.4 Use case ..................................................... 17  
  2.3.5 Challenges .................................................. 18  
2.4 System design ...................................................... 19  
  2.4.1 Approach overview ......................................... 19  
  2.4.2 System pipeline ............................................. 20  
  2.4.3 Spectrum-based approach .................................. 21  
  2.4.4 Motion-based approach .................................... 22  
  2.4.5 Random vibration-based approach ....................... 24  
2.5 Evaluation .......................................................... 27
3. DEFENSE SYSTEMS 2: VOICE LIVENESS DETECTION ON AR HEADSETS 34

3.1 Introduction 34

3.2 Preliminary 37
   3.2.1 Human voice production and propagation 37
   3.2.2 Piezo contact microphone 39
   3.2.3 Attack model 39
   3.2.4 Use case 40
   3.2.5 Feasibility study and challenges 40

3.3 System design 42
   3.3.1 System overview 42
   3.3.2 Word segmentation and spectrogram generation 44
   3.3.3 Spectrogram enhancement 45
   3.3.4 Feature extraction and classification 46
   3.3.5 Decision combination 49

3.4 Evaluation 50
   3.4.1 Hardware 50
   3.4.2 Overall performance 50
   3.4.3 Influence of training dataset size 52
   3.4.4 Influence of the ratio of $\gamma$ relative to the maximum distance 53
   3.4.5 Influence of voting threshold 54
   3.4.6 Influence of sentence length 55
   3.4.7 Influence of background noise 55

3.5 Chapter summary 56

4. DEFENSE SYSTEMS 3: FACE FORGERY DETECTION 58

4.1 Introduction 58

4.2 Literature review 62
   4.2.1 Face reenactment techniques 62
   4.2.2 Face liveness detection in videos 62

4.3 Preliminary 63
   4.3.1 Face forgery using face reenactment 63
   4.3.2 Light metering of digital cameras 64
4.3.3 Face reflection of screen light .................................................. 65
4.3.4 Feasibility study ................................................................. 66
4.3.5 Challenges .............................................................. 66
4.4 System design ............................................................... 67
  4.4.1 Adversary model .......................................................... 67
  4.4.2 System Architecture ......................................................... 68
  4.4.3 Luminance extraction ....................................................... 69
  4.4.4 Preprocessing ............................................................. 71
  4.4.5 Feature extraction .......................................................... 73
  4.4.6 Fake video detection ......................................................... 76
4.5 Evaluation .............................................................. 79
  4.5.1 Implementation and dataset .............................................. 79
  4.5.2 Evaluation metrics .......................................................... 79
  4.5.3 Overall performance ......................................................... 80
  4.5.4 Influence of decision threshold ........................................... 81
  4.5.5 Influence of screen size ..................................................... 82
  4.5.6 Influence of number of detection attempts ............................. 82
  4.5.7 Influence of number of training data ..................................... 83
  4.5.8 Influence of sampling rate ................................................ 84
4.6 Chapter summary .......................................................... 84

5. DEFENSE SYSTEMS 4: SECURING PIN-BASED AUTHENTICATION SYSTEMS .................................................. 86

  5.1 Introduction .............................................................. 87
  5.2 Literature review .......................................................... 90
  5.3 Preliminary .............................................................. 92
    5.3.1 Ambient light sensor ..................................................... 92
    5.3.2 Attack model .......................................................... 93
    5.3.3 Feasibility study .......................................................... 94
    5.3.4 Challenges ............................................................. 95
  5.4 System Overview .......................................................... 96
  5.5 PIN Input Detection .......................................................... 98
    5.5.1 Coarse-grained PIN input detection ................................... 98
    5.5.2 Fine-grained starting and ending points detection ............... 99
  5.6 User Authentication .......................................................... 101
    5.6.1 Feature extraction .......................................................... 101
    5.6.2 Classification based on multiple additive regression tree ........ 104
  5.7 Evaluation .............................................................. 105
    5.7.1 Experimental prototype ................................................ 105
    5.7.2 Data collection .......................................................... 106
    5.7.3 System performance for normal users ................................ 107
    5.7.4 System performance against two types of PIN replay attack .... 108
6. ATTACK SYSTEM: BREAK LOCATION PRIVACY IN AR APPLICATIONS

6.1 Introduction
6.2 Literature review
6.3 Preliminaries and Problem Formulation
  6.3.1 Location-based Multi-player AR Overview
  6.3.2 Key Insight
  6.3.3 Adversary Model
6.4 Overview of the attack
  6.4.1 AR Content Generation and Deployment
  6.4.2 Recursive Region Detection
  6.4.3 Network Throughput Processing
6.5 Experimental Setup
  6.5.1 Implementation
  6.5.2 Data Collection
  6.5.3 Evaluation Metric
6.6 Experimental Results
  6.6.1 Performance of Location Detection
  6.6.2 Performance of Trajectory Construction
  6.6.3 Performance on A Real AR Application
  6.6.4 Influence of Different Paths
  6.6.5 Performance on Different Devices
  6.6.6 Performance on Large-scale Long-Term Tracking Simulation
  6.6.7 Influence of Traffic Noise
  6.6.8 Influence of Different Speeds
  6.6.9 Battery Consumption
6.7 Mitigations
6.8 Discussion
6.9 Chapter Summary

7. CONCLUSION

7.1 Summary of contributions
7.2 Future research

PUBLICATIONS
LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>The logic structure of the dissertation.</td>
<td>7</td>
</tr>
<tr>
<td>2.1</td>
<td>The differences between the human vocal system and a loudspeaker.</td>
<td>15</td>
</tr>
<tr>
<td>2.2</td>
<td>The spectra of audio signals collected from two microphones near the mouth, the throat, and the loudspeaker.</td>
<td>16</td>
</tr>
<tr>
<td>2.3</td>
<td>The use case of our liveness detection system.</td>
<td>18</td>
</tr>
<tr>
<td>2.4</td>
<td>System pipeline.</td>
<td>19</td>
</tr>
<tr>
<td>2.5</td>
<td>The spectra difference.</td>
<td>21</td>
</tr>
<tr>
<td>2.6</td>
<td>Filtered acceleration waveforms from a normal user and an attacker.</td>
<td>23</td>
</tr>
<tr>
<td>2.7</td>
<td>cumulative distribution function (CDF) of distances between acceleration sequences of normal users and the attacker.</td>
<td>24</td>
</tr>
<tr>
<td>2.8</td>
<td>The spectrum of the audio signal with random vibration injected.</td>
<td>25</td>
</tr>
<tr>
<td>2.9</td>
<td>Single-Sided Amplitude Spectrum w and w/o injected vibration.</td>
<td>26</td>
</tr>
<tr>
<td>2.10</td>
<td>A simple graphical user interface.</td>
<td>26</td>
</tr>
<tr>
<td>2.11</td>
<td>Performance of spectrum-based approach.</td>
<td>27</td>
</tr>
<tr>
<td>2.12</td>
<td>Performance of motion-based approach.</td>
<td>29</td>
</tr>
<tr>
<td>2.13</td>
<td>Influence of background noise.</td>
<td>31</td>
</tr>
<tr>
<td>2.14</td>
<td>Influence of different passphrases.</td>
<td>32</td>
</tr>
<tr>
<td>3.1</td>
<td>Human vocal system and two propagation paths of the voice.</td>
<td>36</td>
</tr>
<tr>
<td>3.2</td>
<td>Contact microphone and its frequency response.</td>
<td>39</td>
</tr>
<tr>
<td>3.3</td>
<td>The spectrograms of voices through air and internal body.</td>
<td>41</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
<td></td>
</tr>
<tr>
<td>3.4</td>
<td>System pipeline.</td>
<td></td>
</tr>
<tr>
<td>3.5</td>
<td>Spectrogram enhancement.</td>
<td></td>
</tr>
<tr>
<td>3.6</td>
<td>Correlation matrix for two voices from the same user.</td>
<td></td>
</tr>
<tr>
<td>3.7</td>
<td>Feature analysis.</td>
<td></td>
</tr>
<tr>
<td>3.8</td>
<td>Testbed for collecting internal body voice.</td>
<td></td>
</tr>
<tr>
<td>3.9</td>
<td>Overall performance of our system.</td>
<td></td>
</tr>
<tr>
<td>3.10</td>
<td>System performance with respect to training data size and distance threshold.</td>
<td></td>
</tr>
<tr>
<td>3.11</td>
<td>System performance with respect to voting threshold, sentence length, and background noise.</td>
<td></td>
</tr>
<tr>
<td>4.1</td>
<td>Face forgery detection in real-time video chat.</td>
<td></td>
</tr>
<tr>
<td>4.2</td>
<td>An example of face reenactment techniques [1].</td>
<td></td>
</tr>
<tr>
<td>4.3</td>
<td>Luminance of face-reflected light when screen light changes.</td>
<td></td>
</tr>
<tr>
<td>4.4</td>
<td>System architecture.</td>
<td></td>
</tr>
<tr>
<td>4.5</td>
<td>Facial feature localization and interested area extraction.</td>
<td></td>
</tr>
<tr>
<td>4.6</td>
<td>The spectrum of luminance signals w/ and w/o screen light change.</td>
<td></td>
</tr>
<tr>
<td>4.7</td>
<td>Preprocessing of luminance signals.</td>
<td></td>
</tr>
<tr>
<td>4.8</td>
<td>The luminance change in transmitted and received video respectively.</td>
<td></td>
</tr>
<tr>
<td>4.9</td>
<td>An example of LOF-based classification.</td>
<td></td>
</tr>
<tr>
<td>4.10</td>
<td>Monitors used in our experiments.</td>
<td></td>
</tr>
<tr>
<td>4.11</td>
<td>Overall performance for a single detection.</td>
<td></td>
</tr>
<tr>
<td>4.12</td>
<td>Impact of decision threshold.</td>
<td></td>
</tr>
<tr>
<td>4.13</td>
<td>True acceptance rates under different screen sizes.</td>
<td></td>
</tr>
</tbody>
</table>

xv
6.7 Noise removal and throughput accumulation. .......................... 133
6.8 The AR contents deployment. ............................................. 137
6.9 Performance of a single location detection. ......................... 139
6.10 System performance for three paths. ................................. 143
6.11 Location-detection accuracies on two devices. ..................... 145
6.12 Percentage of top $N$ locations. .................................... 147
6.13 Margin of error verses number of AR contents. .................... 147
6.14 Location detection accuracies in jogging scenario. ............... 150
LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Types of loudspeakers.</td>
<td>28</td>
</tr>
<tr>
<td>2.2</td>
<td>Users’ information.</td>
<td>28</td>
</tr>
<tr>
<td>2.3</td>
<td>Performance of vibration detection.</td>
<td>30</td>
</tr>
<tr>
<td>6.1</td>
<td>Popular mobile applications that ask for “read phone status and identity” permission.</td>
<td>125</td>
</tr>
<tr>
<td>6.2</td>
<td>Location detection accuracy for the overlapped area.</td>
<td>140</td>
</tr>
<tr>
<td>6.3</td>
<td>Location-detection accuracies with different distances between neighboring locations.</td>
<td>141</td>
</tr>
<tr>
<td>6.4</td>
<td>Performance of trajectory construction.</td>
<td>142</td>
</tr>
</tbody>
</table>
CHAPTER 1

INTRODUCTION

Cyber-physical systems (CPS) refer to a group of systems that combine the real physical world with cyber components. In traditional CPS (e.g. smart power grid), devices are usually huge and stationary. In recent years, due to the fast development of pervasive computing, sensor manufacturing, and artificial intelligence technologies, mobile CPS that extend the application domains of traditional CPS have become increasingly popular. In mobile CPS, devices have rich features, such as significant computational resources, multiple communication radios, various sensor modules, and high-level programming languages. These features enable us to build more powerful and convenient applications and systems for mobile users, but they can also be leveraged by attacks to launch new types of attacks. The security and privacy issues can exist in any application of mobile CPS. Therefore, in terms of the defense aspect, we focus on three important topics: voice liveness detection, face forgery detection, and securing authentication systems that are based on personal identification number (PIN). In addition, we also study the location privacy of AR applications from the attack perspective. This dissertation investigates major security and privacy threats in mobile CPS in three major applications and discusses the feasible defense strategy against such attacks.

1.1 Motivation and challenges

Although commercial mobile CPS are designed to be sufficiently secure, they still can be attacked in many ways. Apart from traditional attacks from the operating
system level, the rich sensor data and side-channel information can also be processed by the attacker to break the mobile CPS in two ways. First, since most mobile CPS regard all collected sensor data as trustworthy by default, attackers can manipulate the sensor data to fool the mobile CPS. Second, attackers can also leverage non-sensitive sensor data to infer sensitive information of mobile CPS users.

Among all commercial mobile devices, smartphone is an ideal platform of mobile CPS. Most smartphones have various sensors available in devices, including motion sensors, cameras, and audio components. In addition, they are also equipped with essential hardware to support different communication methods, such as Wi-Fi and Bluetooth, which enables smartphones to connect the physical worlds of users with cyber worlds. However, recent researchers show that current smartphones are vulnerable to various attacks. For example, since smartphone applications always assume the obtained sensor data to be trustworthy, researchers have shown that attackers can break current voice authentication and recognition systems with a very high success rate by launching voice replay attacks [2]. Moreover, the attacker can also infer victim’s sensitive information by installing a malicious application on the victim’s device and analyzing sensor data. Even if current operating systems have permission management systems to alert users how sensitive sensor data are collected, attackers can still gather useful information using low-permission sensor data. For instance, Ba et al. found that an attacker can eavesdrop the audio on the victim’s device only using the data of motion sensors [3].

Augmented reality (AR) devices, such as AR headsets, are a new type of mobile CPS that is getting more and more attentions. The basic idea of AR is to provide an immersive experience by overlaying virtual objects on the real scenes. On the one hand, most AR devices follow a similar design as that of smartphones in terms of hardware and software. Therefore, many attacks that were designed for smartphones can be launched on an AR device. However, due to the special usage scenarios of
AR devices, some defense systems designed for traditional smartphone applications may not work in AR scenario. For example, Zhang et al. proposed a system on smartphone to defend against voice replay attack [4]. The key insight is to leverage a microphone and a speaker as a Doppler radar to detect the articulatory gestures during speech. Unfortunately, such a system cannot be implemented on AR headsets since the speakers on these devices are usually near the ears and away from the mouth.

On the other hand, current AR devices, especially AR headsets are different from smartphones in three major aspects. First, AR devices need to continuously sense the surrounding environment using multiple sensors to ensure good AR experience while they are in use. Second, due to the limited battery capacity and computational resources, AR devices usually need to send the data to a server for outsourced processing. Third, the size of AR data (no matter sensor data and virtual objects) is usually much larger than that on smartphone. These major differences also introduce extra challenges to build defense systems on AR devices.

Besides the smart personal devices, many Internet of Things (IoT) can also be regarded as mobile CPS since they can both sense the physical world and be connected to the cyber world. For example, Raspberry Pi can be connected to a diverse types of sensors via different interfaces. Also, these data can be either processed locally on the board or transmitted to other terminals in the same network in both wired and wireless ways. Considering that the sensors in commercial mobile CPS devices are usually weak in terms of capabilities, IoT devices can be a great help for building better defense systems to protect the mobile CPS users.

In this dissertation, we summarize the major challenges in protecting the security and privacy of mobile CPS as follows:

- **Limited resource:** Due to the size constraint of commercial mobile CPS, the energy capacity and computation resources are very limited. If a defense system
uses too much CPU time, the energy drain will increase correspondingly, which will negatively impact the normal use of these devices. Therefore, the designers of defense systems should be careful about the computational complexity of their algorithms while still ensuring good defense performance.

• **Rich sensor data and side-channel information:** Mobile CPS are usually equipped with multiple sensors. These sensors can be triggered to sense the surrounding physical world. In addition, side-channel and context information of mobile CPS is also important and reflects how the mobile CPS interacts with users and surrounding environment. As researchers, it is challenging to identify how these information can be used to protect the security and privacy of mobile CPS and be aware of what kind of new attacks can be launched with the help of the data.

• **Performance and cost:** There is always a trade-off between performance and cost in modern systems. Since many defense systems on mobile CPS are data driven using artificial intelligence (AI) techniques, it is essential to be aware of what is the cost to get the required data and labels. In general, more data means higher and more robust performance, but the data collection and labeling procedures are usually expensive for a new user.

• **Generalization:** Generalization is another challenging objective when building a defense system. In this context, generalization means whether a defense system can still provide high performance if the attack methods slightly change or the defense system is used on an untested dataset. First, defense systems are built based on certain assumptions about attackers’ abilities. Therefore, it is essential for designers to consider how the defense performance changes when attackers change their attack behavior. For example, if it is cheap or easy for attackers to break the defense system by changing their attack
methods, the defense system needs to be re-designed to protect mobile CPS from this potential threat. Second, if a defense system is trained and tested based on the data of a certain group of users, designers should also evaluate the system to prove that the found patterns still hold for other groups.

1.2 Major contributions

In this dissertation, we carry out the study of the security and privacy issues in mobile CPS. The availability of rich sensor data and side-channel information, sometimes can be leveraged by designers to build better defense systems. Attackers can also launch new types of attacks using these information. Among all possible attacks, this dissertation focuses on four important threats to mobile CPS: voice replay attacks on smartphones, voice replay attacks on AR devices, face forgery in real-time video chat, and PIN leakage. The contributions of this dissertation can be summarized as follows:

- We study the voice replay attacks on smartphones and existing defense systems, and propose a new voice liveness detection system. To our best knowledge, our system is the first to leverage the throat voice as the side channel to detect the voice liveness. Moreover, we leverage two different sensors, microphone and motion sensor, to capture the information around the throat, and propose two separate detection schemes using these two types of data. In addition, to defend against strong attackers who can steal the victim’s throat voice and replay both voices to our system, we further propose a random noise injection scheme to detect whether the throat voice is stolen or newly produced by a live person.

- We also propose the first voice liveness detection system for AR headsets, which has not been given much attention. We find that the human voice can propagate through the internal body and propose to leverage the contact microphone to
collect this internal body voice around the temple. In addition, we design filters to remove the background noise in both voice and extract useful features in the time-frequency domain. To determine whether the voice is from a live person, we propose a low-cost algorithm to reduce the computation overhead and energy consumption.

- We identify face forgery attacks using the state-of-the-art face reenactment techniques. Based on our findings, we propose a novel fake facial video detection systems that is robust against this new face forgery attacks. Unlike existing algorithms that leverage a separate neural network, our system implements relatively low-cost feature extraction and classification methods. Moreover, our system can be generally used for detecting fake facial videos that are produced by other techniques as long as they cannot generate the corresponding luminance changes on facial area in real time.

- We also study the security of PIN along with existing attack and defense methods. Considering the limitations of current defense systems, we propose a new protection system for PIN-based system by leveraging a small light sensor. Compared with existing methods, our system claims high system performance for legitimate users and against attackers. Moreover, our system does not change the original interaction between users and PIN-based system. In addition, we consider the important biometrics in both PIN input behavior and the user’s biological features, which means that attackers cannot break our system even if they can imitate the behavior of the victim.

- We study location privacy in AR applications and reveal a potential security threat in current location-based multi-player AR applications. To show the practicality of such attacks, we design a fake AR content deployment strategy to create unique network traffic pattern and propose traffic processing algorithms
to infer the location from noisy traffic data. Compared with existing attack methods on location privacy, the new attack method does not need any location-related permission, which makes it harder to be detected by users. Second, our attack method has better performance on detecting single location and can also be used to detect the trajectory.

1.3 Dissertation overview

As shown in Fig. 1.1, the organization of the remaining chapters of this dissertation is as follows: In the first part of this dissertation, from Chapter 2 to Chapter 5, we use four defense systems we have designed to show how the rich sensory information can be used to build defense systems. Specifically, we focus on three applications and corresponding attacks in mobile CPS: voice replay attacks for voice-based applications, fake facial videos for video-based communication, and PIN replay attacks for PIN-based authentication systems. In terms of the voice-based applications, we first introduce our voice liveness detection system that is designed for smartphones in Chapter 2. Considering that such a defense system cannot protect future AR devices, we further propose a new voice liveness detection system for AR headsets by leveraging the internal body voice in Chapter 3. In Chapter 4, we design a
fake facial video detection system for the real-time video chat scenarios, which enables the user to know whether the face on the screen is real. In Chapter 5, we study the existing attacks and mitigation methods for PIN-based authentication systems. We propose a protection layer by leveraging the light signals received at the PIN pad to reject PIN inputs from attackers. In the second part of this dissertation, we study how the side-channel information on mobile CPS can be used by attackers to launch new types of attacks. More specifically, we propose a new attack system that aims to infer the real-time location of the user by analyzing the network traffic on the AR device of the user. Conclusion and future directions are given in Chapter 6.
The recent proliferation of smartphones has been the primary driving factor behind the booming of voice-based mobile applications. However, the human voice is often exposed to the public in many different scenarios, and an adversary can easily “steal” a person’s voice and attack voice-based applications with the help of state-of-the-art voice synthesis/conversion softwares. In this chapter, we propose a robust software-based voice liveness detection system for defending against voice spoofing attack. The proposed system is tailored for mobile platforms and can be easily integrated with existing mobile applications. We propose three approaches based on leveraging the vibration of human vocal cords, the motion of the human vocal system, and the functionality of vibration motor inside the smartphone.

2.1 Introduction

The recent proliferation of smartphones coupled with the demand for a convenient and non-intrusive way of communication and control have been the primary driving factors behind the booming of voice-based mobile applications. In addition to traditional voice over IP (VoIP) applications, e.g., Skype, which allows users to make voice calls to contacts, voice-based mobile applications have also become mainstream. These applications all provide a voice input interface, which allows users to submit their voices and receive information from that voice. For example, WeChat provides
“Voiceprint” [5] authentication interface, which allows users to log into their accounts by speaking passphrases. Besides, SayPay [6] offers a solution that fuses mobile payments with the human voice. These voice-based mobile applications can be quickly developed and implemented for existing smart devices as they require only a microphone, which is small and inexpensive [7].

However, unlike other human biometrics, the human voice is often exposed to the public in many different scenarios, e.g., people making a presentation in public, answering phone calls, talking loudly in a restaurant. As such, with the availability of high quality and low-cost handy recorders and other recording devices (e.g., smartphones), a malicious user can easily “steal” a person’s voice without being noticed. Several security issues are, therefore, caused by the leakage of people’s voices, which poses a severe threat to voice-based applications [8, 9, 10]. For instance, with state-of-the-art speech synthesis techniques (e.g., Adobe Voco [11]), an adversary could impersonate the victim to spoof the voice-based authentication system once they acquire enough victim’s voice samples. Since voice is considered as a unique biometric of a person, and thereby, it is characterized as a basis for personal authentication [12], these voice-spoofing attacks would result in severe consequences harmful to victim’s safety, reputation, and property.

The traditional technique for defending against voice-spoofing attacks is to implement an automatic speaker verification (ASV) system, which has already been deployed in many popular mobile applications, such as WeChat. The ASV systems employ unique vibration patterns of a user’s vocal chords and the sound-based feature created by other physical components (e.g., mouth) to assign a unique fingerprint. However, spoofing techniques against these systems have also progressed drastically [9, 13, 8]. Moreover, when detecting the attack, current ASV systems require prior knowledge of specific voice spoofing techniques used by the attacker [14], which is unrepresentative of the practical scenario. Therefore, the development of a generalized
defense system for voice-spoofing attacks is of the utmost importance. Recently, many liveness detection systems are proposed to fight voice-spoofing attacks by studying the differences between the human vocal system and loudspeakers. VoiceLive [15] can fight replay attack by capturing time-difference-of-arrival (TDoA) changes in a sequence of phoneme sounds to the two microphones of the phone. However, it needs the same relative location of user’s mouth during authentication, which is hard to satisfy in practice. A liveness detection system is proposed in [4] can detect a live user by leveraging the unique articulatory gestures of the user when speaking a passphrase. However, it cannot work if the attacker performs a jamming attack using high-frequency audio.

Considering the limitations of current solutions, we propose a robust software-based voice spoofing defense system which is tailored for mobile platforms and can be easily integrated with existing voice-based mobile applications. Our solutions use the unique vibration of human vocal cords and the movement of throat as key differentiating factors for liveness detection. Compared with existing ASV system, our solution does not assume any prior knowledge of the attacking method and is easy to operate. Moreover, our pure software solution is ready to use and can be seamlessly deployed on off-the-shelf smartphones.

We solve two challenges in the design of our system. First, in order to capture the vibration of vocal cords and the movement of the throat simultaneously, we need to use both the prime microphone (at the bottom) and front microphone. Since different people have different speaking habits and use different languages, it is difficult to extract a common pattern that can be used to detect the liveness of a speaker. To solve this problem, we perform spectrum subtraction of two audio signals and utilize the energy differences of different time slices and frequency band as a unique feature. Second, the sampling rate of the accelerometer-equipped on smartphones is only 50 Hz, which is not good enough to fully recover the human throat movement. To address
this issue, we extract multiple features from the acceleration readings to build a robust classification model and use it to determine whether the captured data is generated by human throat movement.

2.2 Literature review

2.2.1 Voice-based Mobile Applications

With advances in modern smartphones, voice-based mobile applications, i.e., mobile apps, have grown in popularity as these applications provide an intrinsically efficient, comfortable interaction interface to users. These existing voice-based mobile applications can be divided into two categories based on their functionalities: 1) voice communication and 2) voice control. For the first category, we have VoIP apps, by which people can make a voice call to anyone using the Internet (e.g., Skype, Google Voice). In addition, many voice instant messenger mobile apps have been developed in recent years, such as WeChat, WhatsApp, TalkBox, Skout, and iMessage. These apps allow users to record short voice messages and directly send them to others. Hence, this offers opportunities to attackers who are able to launch a voice-spoofing attack by imitating a victim’s voice, tone, and speaking style. This attack could harm victim’s reputation, safety, and property. The attacker could scam victim’s friends and family through fake phone calls and leave fake voice messages, etc.

2.2.2 Automatic Speaker Verification System

An automatic speaker verification system (ASV) is able to accept or reject a speech sample submitted by a user for claiming certain identity [16]. Recently, the development of ASV systems has made major progress as they are widely adopted by smartphones and online commerce [17, 18]. Existing ASV systems are divided into two types: text-dependent and text-independent. Text-independent ASV systems are
able to accept arbitrary utterances, i.e., different speaking habits and languages from speakers [19]. As a matter of fact, the text-dependent ASV is widely selected for authentication applications since it provides higher recognition accuracy with fewer required utterances. The current practice of building an ASV system involves two processes: offline training and runtime verification. During the offline training phase, the ASV system uses several speech samples provided by the genuine speaker to extract certain spectral, prosodic [20, 21], or other high-level features [22, 23] and uses them to create a speaker model. Then, in the runtime verification phase, the ASV system uses the trained speaker model to verify the incoming voice.

2.2.3 Voice-Spoofing Attacks

The voice-spoofing attacks aim to break the biometric identification of the victim. It can be divided into two categories: voice replay attack and voice synthesis or conversion attack. [24] shows that an attacker can overcome text-dependent ASV systems by concatenating speech samples from multiple short voice segments of the target speaker. Due to the simplicity of voice replay attacks, a few research papers have been published in developing relay attack countermeasures [24, 25, 26]. However, all these countermeasure systems suffer high false acceptance rate (FAR) compared to respective baselines. In [27], the authors demonstrate the vulnerabilities of ASV systems for voice synthesis attack (generate artificial speech from text input). [28] proposes the voice conversion attack in which the attacker converts the spectral and prosody features of his or her own speech and makes it to resemble to the victim’s speech. To detect voice synthesis and voice conversion attack, [29] exploits artifacts introduced by the vocoder to discriminate converted speech from original speech.
2.3 Preliminary

2.3.1 Attack model

The voice-spoofing attacks aim to attack the biometric identification of the normal user. In our attack models, an attacker is able to access victim’s smartphone and record the voice of the victim without being noticed. Also, an attacker can be equipped with one or more high-quality loudspeakers. Based on collected audio signals, an attacker can launch various attacks like replay attacks. The voice-spoofing attacks considered in our work can be divided into two categories.

**Simple replay attack.** In this type of attack, an attacker can use high-quality loudspeakers to replay collected victim’s voice or morphed voice, so that the attacker can impersonate the victim at a high degree of similarity. We assume that an attacker can access victim’s smartphone in the case of not being noticed.

**Strong replay attack.** In this type of attack, we assume that the attacker can capture more information besides the victim’s voice at the mouth. For example, the attacker can attack the database of current ASV system and fetch the voice signals at both victim’s mouth and throat. An attacker can leverage multiple loudspeakers to replay two audio signals to two microphones and imitate the human vocal systems better.

2.3.2 Background knowledge

In order to achieve robust liveness detection, we need to understand the structural differences between the human vocal system and loudspeakers. As shown in Fig. 2.1(a), the mechanism for producing the human voice can generally be subdivided into three parts: the lungs, the vocal folds, and the articulators. The lung first produces adequate airflow and air pressure to vibrate vocal folds. The vocal cords vibrate and chop up the airflow from the lungs into audible pulses that form the
laryngeal sound source. Then, the length and tension of the vocal cords are adjusted to produce ‘fine-tune’ pitch and tone. The articulators consisting of tongue, palate, cheek, lips further filter the sound generated from the larynx to strengthen it or weaken it. The vocal folds (vocal cords) are the primary sound source to produce voiced phoneme in the human vocal system. Besides voiced phoneme, there exist other sound production mechanisms produced from the same general area of the body, involving the production of unvoiced consonants, clicks, whistling, and whispering. The only difference between voice and unvoiced phonemes is that there is no vibration of the vocal cords for unvoiced phonemes. This fact suggests that the audio signals collected near the throat and the mouth can be different, and this difference can only be produced by the human speaker.

Strong attackers usually use high-quality loudspeakers for spoofing attacks. As shown in Fig. 2.1(b), the loudspeakers usually use an electromagnet to translate an electrical signal into an audible sound. The electromagnet is a metal coil that creates a magnetic field when there is an electric current flows through it. When electrical pulses pass through the coil of the electromagnet, the direction of the magnetic field is frequently changed. Also, there is a permanent magnet fixed firmly into
Figure 2.2: The spectra of audio signals collected from two microphones near the mouth, the throat, and the loudspeaker.

the loud speaker. With rapidly changed magnetic field, the coil is attracted to and repelled from the permanent magnet. As a result, the cone attached on the coil will vibrate back and forth, pumping sound waves into surrounding air and smartphone’s speaker, which means the two microphones of a smartphone around the loudspeaker will capture very similar audio signals.

2.3.3 Key insights

In order to resist two types of attacks that we consider, we need to leverage the structural differences between human vocal systems and loudspeakers. We observe that human voice can be divided into the voiced and unvoiced part. During voiced part, the vocal cords keep vibrating and generate low-frequency audio signals at the throat. The vocal cords stop vibrating during unvoiced part, while the other parts of the human vocal system generate different sounds. We collect voice signals when a
user says “Six” at two locations (the throat and the mouth) using two microphones, and the results are illustrated in Fig. 2.2. It is clear that the audio signal collected near mouth reserves the information of unvoiced parts, but most information of the unvoiced part is lost in the audio signal collected near the human throat. Also, both audio signals reserve the information of voiced part, while the audio signal collected near throat only contains the information at the low-frequency part. Different from human vocal systems, the cone keeps vibrating for both voiced and unvoiced parts in order to generate sounds. We use a loudspeaker to replay the voice of the user and collect the audio signals in the same way. Fig. 2.2 also shows the spectrum of the same audio signal played by a loudspeaker and captured by the prime microphone. We can observe that the spectrum contains much more information of unvoiced parts than that collected near the human throat.

When a person is speaking, the vocal cords vibrate at a relatively high frequency, and the throat also moves up and down in a relatively low frequency. Opposite to this, loudspeakers do not have the same movement pattern. If we put a motion sensor next to a human throat, the vibration of vocal cords and movement of a throat generate two different influences on the readings. Based on this observation, we argue that the influences generated by vocal cords and throat are hard to be imitated by loudspeakers. We will discuss the liveness detection using acceleration signals in Section 2.4.4.

### 2.3.4 Use case

In order to successfully defend users from spoofing attacks, our system requires users to put the bottom side of the smartphone on the throat while using the normal voice authorization systems, as shown in Fig. 2.3. We leverage two microphones that are available on most current smartphones. The prime microphone is used to capture the low-frequency voice cased only by the human throat, and the front microphone is
used to record human voice over the entire frequency band. Two audio signals are well synchronized by smartphones operating systems. The distance between the human throat and the prime microphone must be zero, and the distance between human lips and the front microphone is about 6 cm. Since the distance is pretty short, the time delay between two audio signals is less than 8 samples when the sampling rate is 44,100 samples per second. While speaking the passphrase, the user should put the bottom side of the smartphone on the throat. During this process, the user should be in stationary postures, like sitting and standing.

2.3.5 Challenges

Although we provide insights in Section 2.3.3, it is still challenging to perform liveness detection on a smartphone using only audio signals and acceleration readings. The first challenge is how to extract useful information from audio signals in two channels. Since different people have different speaking habits and use different languages, it is extremely hard to extract a common pattern that can be used to determine if the source is a real person. To solve this problem, we compute the Short-time Fourier transform (STFT) of two audio signals and get their spectrum subtraction. The spectrum subtraction is then treated as a picture, and the color
Figure 2.4: System pipeline.

represents the energy in corresponding time-frequency band. We use an image classification algorithm to determine the liveness of the speaker.

The second challenge is that current smartphones only provide acceleration reading at a sampling rate of 50 Hz. Since voice-based authentication only lasts for about 3 seconds, it is hard to extract human throat movement from limited acceleration readings (about 150 samples). To address this issue, we extract multiple features that describe acceleration signal in different aspects. The features are then used to build a robust classification model and determine if the acceleration reading is affected by human throat movement.

2.4 System design

2.4.1 Approach overview

The key idea underlying our liveness detection system is to fully leverage the nature of human vocal system in order to detect the liveness of the speaker. When a live speaker speaks a passphrase, the primary microphone only records the voice produced by the vibration of the vocal cords, while the front microphone records the voice produced by the entire vocal system. Based on this observation, we examine the spectrum property of audio signals and propose a spectrum-based approach to determine whether the input audio is from a real user in Section 2.4.3. Moreover, human throat will move up and down, and vocal cords will vibrate in high frequency.
Both movements generate different influences on the accelerator embedded in the smartphone. A motion-based approach is designed in Section 2.4.4 to find proper features and classification model to determine if an acceleration sequence is from a normal user. An attacker, who wants to perform replay attack, cannot imitate the human vocal system well and cannot get the same pattern on the audio spectrum and acceleration sequence. Furthermore, in case that an strong attacker can steal victim’s raw audio files from a database, we design a random vibration-based approach to inject random noise in the collected audio signals. By analyzing the number of injected vibrations, our system can recognize if the input audio signal is new or stolen from the victim.

2.4.2 System pipeline

The pipeline of data collection and processing is shown in Fig. 2.4. Our system captures audio signals in two channels and collects acceleration reading at the same time. The acceleration reading is further processed and analyzed to validate if the smartphone is touching human throat during data collection. A classification model is trained based on support vector machine (SVM) using proper features. The raw audio signals are processed by STFT to get the spectra. We compute the difference between the two spectra and use it as an input to match existing patterns. If the spectrum subtraction matches the existing patterns, the spectrum-based classification model will regard the user as a real person. In case that the attacker steals user’s voice recording from other databases, we inject a random and short vibration during data collection. The random vibration is then used to evaluate if the input is a new recording or a stolen recording. Then, three results are combined together to get the final validation results. A user is recognized as a real person if and only if all three decision components give positive results.
2.4.3 Spectrum-based approach

To distinguish if the voice is from a live speaker or a loudspeaker, we need to find features to represent the relationship and differences between two audio samples collected from two microphones. In order to capture features on both frequency domain and time domain, we perform STFT on two audio samples with a window size of 46ms based on:

\[
X(\tau, \omega) = \sum_{n=-\infty}^{n=+\infty} x[n]w[n - \tau]e^{-j\omega n}
\]  

(2.4-1)

where \(\tau\) is the time axis, \(\omega\) is the frequency axis, \(x[n]\) is the an audio sample, and \(X(\tau, \omega)\) is a complex function representing the phase and magnitude of the signal over time and frequency. Then, the spectrogram of the complex function \(X(\tau, \omega)\) is computed based on:

\[
spectrogram\{x[n]\}(\tau, \omega) \equiv |X(\tau, \omega)|^2
\]  

(2.4-2)

Fig. 2.2 illustrates the spectra of two audio samples when a user speaks “Six” to a smartphone, and we can find following observations that can help us detect the liveness of the speaker: 1) Since the vocal cords do not vibrate during unvoiced speech, the prime microphone loses most information for unvoiced part, while the front microphone can capture this information; 2) For the voiced part, the prime
microphone can only capture voice information at low frequency band. If the voice is from a live speaker, the differences of two spectra should contain most information of the voice except that in the low frequency band of voiced part, as shown in Fig. 2.5. Based on these observations, we compute the difference between two spectra and leverage its energy distribution as the feature to detect the liveness of a speaker. Due to unpredicted noise and speaking volumes of the speakers, it is hard to robustly extract the shape of energy distribution. To solve this problem, we treated the spectra difference as an image, and its energy represents the color. Considering the diversity of energy distribution due to various speaking manners of different people, all energy values (pixels in the image) are used to build the classifier. To eliminate the influence of different speaking time, we resize the spectra difference (the image) and convert them to vectors. The resulted vectors are used to build a binary SVM with nonlinear kernel function to determine whether the input spectra difference satisfies the observations we find.

2.4.4 Motion-based approach

When a user speaks a passphrase into the smartphone of our system, there are two kinds of movements involved. First, the throat will move up and down in a low frequency. In addition, the vocal cords will vibrate in high frequency for voiced phonemes. These two movements will generate different influences on the acceleration readings in the smartphone. To understand the influences of human speaking activity on the acceleration readings, we first collect the acceleration waveforms from normal users. Then, raw acceleration data is smoothed using a moving average filter with window size of 10. Fig. 2.6(a) illustrates the filtered acceleration waveforms under the influences of the human speaking activity. We can see that low-frequency throat movements generate 7 significant pulses by moving up and down. Also, vocal cords vibration affects the acceleration reading in high frequency, which is shown as small
spikes across the whole waveform. We further study the influence of a loudspeaker on the embedded accelerator and find that it is hard for an attacker to perform attacks using a loudspeaker. Fig. 2.6(b) shows the filtered acceleration waveforms under the influences of a loud speaker. We can see that the waveform contains much more significant spikes whose magnitudes are mainly within $[-0.05, 0.05]$. Dynamic Time Warping (DTW) is an efficient way to measure the similarity between two temporal sequences. However, it is hard to determine if an acceleration sequence is collected from a loudspeaker using only DTW algorithm. Fig. 2.7 shows the distributions of distances of acceleration sequence calculated by DTW between normal users and between a normal user and an attacker. We can see that two distributions are very similar. The distances between a user and an attacker are even smaller than those between normal users in some cases. To address this issue, we select 7 features to represent an acceleration sequence: (1) Variance; (2) Minimum; (3) Maximum; (4) Mean; (5) Skewness; (6) Kurtosis; (7) Standard deviation. We select the features based on Principal component analysis (PCA) and use selected features to train an SVM-based classification model. The model is then used to determine if an acceleration signal is from a live speaker or not.
Figure 2.7: cumulative distribution function (CDF) of distances between acceleration sequences of normal users and the attacker.

### 2.4.5 Random vibration-based approach

Even if our spectrum-based approach and motion-based approach can fight spoofing attacks effectively, we argue that there are stronger attackers who can hack the database and steal the voice at victim’s throat. Also, we assume that the strong attacker can leverage multiple speakers and imitate human vocal system perfectly with a high cost. In this case, our spectrum-based approach and motion-based approach cannot ensure good performance. To address this problem, we further introduce a random vibration strategy so that the strong attacker cannot fool our system even if the attacker can steal the raw audio file and imitate victim’s vocal system perfectly. Current smartphone operating system provides us the privilege to operate the vibration motor and define the vibration pattern. We fully leverage the vibration motor embedded in most smartphones. While recording, our system will randomly trigger the vibration motor for a given constant time $t$. Then, our system will detect the number of random vibrations in the received audio signals. If the number is larger than 1, the audio signal is classified as “stolen audio file” and the validation is rejected.

To effectively detect this attack, we need to locate the vibration accurately and determine the value of $t$. There is a trade-off in determining the value of $t$. If $t$ is too small, the intensity of the vibration may not be strong enough to be detected. If $t$ is too large, the noise generated by the vibration will influence the original validation.
process and our system. Based on our experiment, $t = 100\ ms$ gives us the best performance on two smartphones. Due to the high sampling rate provided by the current microphone, we can design a robust algorithm to detect the vibration of smartphone based on the audio signal. Fig. 2.8 shows the spectrum of the audio signal with injected vibration with the length of 100 $ms$ at 1 second. We can see that it is hard to detect the vibration under 15 kHz on the spectrum since the influence caused by vibration is buried by that of the human voice and background noise. However, the influence caused by smartphone’s vibration dominates the high-frequency part of the spectrum (17 kHz $\sim$ 20 kHz). Fig. 2.9 shows the single-sided amplitude spectrum from 17 kHz $\sim$ 20 kHz. It is clear that much more energy is in the given frequency band if there is a vibration.

Based on this insight, we design a vibration detection algorithm to locate the vibration at the frequency domain and validate the duration of each vibration. After getting the raw audio signal from the front microphone, we cut the audio sequence into frames with the equal size of 50 $ms$. Within each time frame, we perform STFT and calculate the sum of energy in the selected frequency band (17 KHz $\sim$ 20 KHz). If the sum of the energy is higher than a threshold $\tau$, a vibration is detected at the current time frame. After vibration detection on all time frames, we group the frames...
that contain a vibration as long as they are neighbors with each other. Then, we check the length of each group. The audio is recognized as collected from a normal user if and only if there only exists one group with the length of $N$. Otherwise, the sequence is recognized as stolen. In our experiment, we find that in some cases the vibration motor vibrates a little bit earlier than the random starting time we generate, and the pre-start will generate an overlap with the previous vibration. So, we set the $N = 3$ and $\tau = -15300$ in our system. Since people need at least 2.2 seconds to finish a 6-digit passphrase, the possibility that an attacker can get the same vibration location of the original audio signal is less than 4.3%.
2.5 Evaluation

2.5.1 Experiment methodology

Experiment setup. In order to evaluate the effectiveness of our system, we build a prototype on two smartphones with different sizes (LG Nexus 5 and MOTO Nexus 6). Both smartphones run on Android. The smartphones are used to capture audio signals in two channels. We design a simple graphical user interface (GUI), as shown in Fig. 2.10, to help users collect audio signals. The application starts capturing user’s voice in two channels as soon as the user presses the button and stops data collection immediately when the user releases the button. After data collection on smartphones, audio signals are sent to a local server for further validation. The server runs on a MacBook Pro with 2.9 GHz Intel Core i5 processor and 8GB 1867 MHz DDR3 memory.

Performance Metrics. In our experiments, we use the following performance metrics to evaluate the validation performance of our system. True acceptance rate is defined as the rate at which a live speaker is correctly accepted by the system and considered as a real person. True rejection rate is defined as the rate at which an attacker is correctly rejected by the system.
Table 2.1: Types of loudspeakers.

<table>
<thead>
<tr>
<th>Maker</th>
<th>Model</th>
<th>Number of trumpets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Willnorn</td>
<td>SoundPlus</td>
<td>2</td>
</tr>
<tr>
<td>Amazon</td>
<td>Echo</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 2.2: Users’ information.

<table>
<thead>
<tr>
<th>Sex</th>
<th>Age</th>
<th>Height (cm)</th>
<th>Average validation time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Female</td>
<td>28</td>
<td>162</td>
<td>2.2616</td>
</tr>
<tr>
<td>Male</td>
<td>27</td>
<td>172</td>
<td>2.9977</td>
</tr>
<tr>
<td>Male</td>
<td>22</td>
<td>180</td>
<td>3.3551</td>
</tr>
<tr>
<td>Male</td>
<td>27</td>
<td>185</td>
<td>4.7149</td>
</tr>
<tr>
<td>Female</td>
<td>25</td>
<td>165</td>
<td>2.7279</td>
</tr>
<tr>
<td>Male</td>
<td>24</td>
<td>187</td>
<td>3.6396</td>
</tr>
<tr>
<td>Female</td>
<td>23</td>
<td>175</td>
<td>3.9321</td>
</tr>
</tbody>
</table>

2.5.2 Performance of spectrum-based approach

To evaluate the performance of our spectrum-based approach, we collect 350 raw audio waveforms from 7 different users. These 7 users include 4 males and 3 females. Each user is asked to speak to the smartphone using the same 6-digit password 50 times. For each user, 5 audio waveforms are used as training data, and the remaining audio waveforms are used as validation data. Also, an attacker uses two loudspeakers to replay victims’ voice. The speakers we use are listed in Table 2.1. During replay attack, the relative location between the loudspeaker and the smartphone should remain the same as for normal users.

We observe that spectrum-based approach can achieve 100% true acceptance rate and true rejection rate for all users. We further evaluate how many training instances we need to build a strong classification model and if we can provide good validation accuracy without collecting training instances from the new user. Therefore, we only use the audio instances collected from one user as training data and perform evaluation on all users. Fig. 2.11 shows the evaluation results. We can observe that, with no less than 4 training instances, our system can accurately detect both live speakers and attackers with a accuracy of 100%. Also, our spectrum-based approach
does not need to collect much training data from a new user, which makes our system more practical.

2.5.3 Performance of motion-based approach

In this subsection, we evaluate the validation performance of our motion-based approach. Similarly, we collect 350 raw acceleration sequences from 7 different users. For each user, 5 acceleration sequences are used as training data, and the remaining are used as validation data. Also, 20 acceleration sequences collected from the attacker are used as negative instances. Fig. 2.12 illustrates the true acceptance of our motion-based approach. We can see that our system can achieve high true acceptance rate of at least 93.33% for most users and provides true rejection rate of 88.89%. To further improve the true acceptance rate, we can add more instances only collected from the new user. We argue that user can manually label wrongly predicted results, and our classification model can leverage new labeled data to build a better classification model for user 1. Experiment results show that the true acceptance rate can be improved to at least 95% after each user adds 5 more instances to the training set.

Figure 2.12: Performance of motion-based approach.
Table 2.3: Performance of vibration detection.

<table>
<thead>
<tr>
<th>Locations</th>
<th>Number of TAV</th>
<th>Number of FAV</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>40</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>40</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>40</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>39</td>
<td>0</td>
</tr>
</tbody>
</table>

2.5.4 Performance of random vibration-based approach

In this study, we investigate the performance of our random vibration-based approach when a strong adversary tries to fool our system by using the collected audio profile of the victim and imitating natural human voice using multiple speakers. First, we examine how accurately our system can detect the number of vibration in the audio signal. We let a user speak in front of our system 20 times. During each recording process, our system generates two non-overlapped vibrations and records the ground truth. We repeat the experiment in 4 different rooms, and the results are illustrated in Table. 2.3. The truly accepted vibration (TAV) is the vibration generated by the human vocal and correctly detected by our algorithm. The falsely accepted vibration (FAV) is the vibration generated by the background noise but wrongly detected. We can see that our vibration detection algorithm can achieve an accuracy of 100% on detecting non-overlapped vibration for the first three rooms. The fourth location is in a kitchen where there may exist high-frequency noise produced by electrical appliances. Several time frames could be wrongly recognized as containing vibration due to the high-frequency noise, which makes the duration of 4 vibrations longer than 150 ms and be rejected by our system. In this scenario, our system can still identify all the vibrations with an accuracy of 97.5%.

2.5.5 Influence of ambient noise

To evaluate the influence of ambient noise on spectrum-based approach, we place a loudspeaker at a distance of about 1 meter. We let the loudspeaker keep playing
audio from a talk show with different volumes. For each volume, we collect 40 audio waveforms from a user. We use the same classification model used in Section 2.5.2. We change the number of positive instances to evaluate the true acceptance rate, and the results are shown in Fig. 2.13. We can see that validation cannot be achieved with three positive instances when there is background noise. When we increase the number of positive instances to 5, we can get true acceptance rate of 82.5% in a low background noise environment. However, the validation performance is deficient in a noisy environment with a true acceptance rate of only 27.5%. This problem can be solved by involving more positive instances or increasing the weight of positive instances in the classification model. We can see that our system can achieve a true acceptance rate of 100% when seven positive instances are involved.

2.5.6 Influence of different passphrases

We also conduct an experiment to show the performance for different passphrases. In our system, we select 8 different passphrases, and a user is asked to repeat each passphrase at least 45 times. For each passphrase, 15 measurements including audio and acceleration are used for training, and the others are used for validation. We also let the attacker perform replay attack 45 times for each passphrase using recorded victim’s voice and use them as negative training and validation data. Fig. 2.14 shows the true acceptance rates for 8 passphrases. We can see that our system can achieve
a true acceptance rate of at least 93.2% for all 8 passphrases. Also, we examine the true rejection rate of our system on the selected 8 passphrases. Experimental results show that our system can provide true rejection rate of at least 86.7%.

2.5.7 Influence of different phones

To show that our system can be implemented on any smartphone equipped with two microphones, we evaluate our system on LG Nexus 5 and LG Nexus 6. The reason we choose these two smartphones models is that the sizes of these two smartphones differ a lot. We asked a user to speak to two smartphones 45 times. Similarly, five measurements on each smartphone are added to the pre-trained model in Sections 2.5.2 and 2.5.3, and the remaining are used as validation data. Experimental results show that our system can achieve a true acceptance rate of at least 95% on the two smartphones and get an acceptable true rejection rate of at least 88.75%.

2.6 Chapter summary

In this chapter, we propose a robust software-based voice spoofing defense system, which is tailored for mobile platforms and can be easily integrated with existing mobile applications. We propose three approaches based on leveraging the audio spectrum pattern, motion of the human vocal system, and the functionality of vibration motor.
This work in this chapter is published in the first paper of the publication list. We summarize our contributions as follows:

- We propose a robust software-only solution for defending against voice-spoofing attacks on smartphones with high accuracy.
- We select and combine advanced acoustic signal processing, mobile sensing, and machine learning techniques and apply them in detecting the unique vibration pattern when speaking.
- We develop a prototype and conduct comprehensive evaluations. Experimental results show that our spectrum-based approach can achieve both 100% true acceptance and rejection rates. Our motion-based approach can achieve mean accuracy of 96.8% and mean true rejection rate of 88.89%. Our random vibration-based approach can detect and locate the vibration with an accuracy of 97.5%. By combining three approaches we proposed, our system can detect a live speaker with a mean accuracy of 94.38% and detect an attacker with a mean accuracy of 88.89%.
CHAPTER 3

DEFENSE SYSTEMS 2: VOICE LIVENESS
DETECTION ON AR HEADSETS

Voice-based input is usually used as the primary input method for AR headsets due to immersive AR experience and good recognition performance. However, recent researches have shown that an attacker can inject inaudible voice commands to the devices that lack voice verification. Even if we secure voice input with voice verification techniques, an attacker can easily “steal” the victim’s voice using low-cost handy recorders and replay it to voice-based applications. To defend against voice-spoofing attacks, AR headsets should be able to determine whether the voice is from the person who is using the AR headsets. Existing voice-spoofing defense systems are designed for smartphone platforms. Due to the special locations of microphones and loudspeakers on AR headsets, existing solutions are hard to implement on AR headsets. To address this challenge, we propose a voice-spoofing defense system for AR headsets by leveraging both the internal body propagation and the air propagation of human voices.

3.1 Introduction

AR applications that overlay a user’s perception of the real world with digitally generated information are on the cusp of commercial viability. To provide better user experience, AR experiences are primarily delivered to AR users via wearable glass devices and head-mounted devices. For example, Microsoft, Google Vuzix, and other
companies have been working on bringing AR to us in the eyeglass form. Moreover, different from traditional human-computer interactions, most existing interactivity technologies (e.g., typing, tapping, clicking, and swiping) have become irrelevant and obsolete in the AR world. Because of the real-world interaction of AR experience, the input methods for AR headsets should fit what a human can understand. Therefore, most AR headsets adopt voice, eye gaze, and gestures as input methods. Among these three input methods, voice-based input is usually used as the primary input method for three reasons: 1) Voice is the primary way to deliver information in daily life, so voice-based input can provide immersive AR experience; 2) Many low-cost AR devices do not have capabilities to track eye gaze and recognize gestures; 3) Most gesture and gaze interfaces have problems with responsiveness and accuracy.

However, voice-based input suffers from various voice spoofing attacks. Recent researches [30, 31] have shown that an attacker can inject inaudible voice commands to the devices that lack voice verification. Moreover, unlike other human biometrics, the human voice is often exposed to the public in many different scenarios, e.g., people making a presentation in public. Even if we secure devices with voice verification techniques, an attacker can easily “steal” the victim’s voice using low-cost handy recorders and attack voice-based applications with the help of state-of-the-art voice synthesis/conversion software. Several security issues are, therefore, caused by the leakage of people’s voices and pose a severe threat to voice-based applications [8, 9, 10]. For instance, with a replay device, an adversary could impersonate the victim to spoof the Google Trusted Voice once they acquire enough victim’s voice samples. Since voice is considered as unique biometrics of a person, these voice-spoofing attacks would result in severe harmful consequences to victim’s safety, reputation, and property.

To defend against voice-spoofing attacks, the voice-based systems need to determine whether the voice is from the person who is using the AR headsets. To achieve this goal, traditional systems primarily use two solutions: 1) Check the
channel noises introduced by recording and the replay devices (loudspeakers); 2) Analyze the reverberation of replaying far-field recordings. However, these solutions have high false acceptance rates of up to 17%, which makes them unsuitable for commercial systems. Recently, many liveness detection systems have been proposed to fight against voice-spoofing attacks by studying the differences between the human vocal system and loudspeakers using phoneme location [15], articulatory gestures [4], magnetic fields of loudspeakers [32], and throat voice [33]. However, all of them are designed for smartphones. Considering the special locations of microphones and loudspeakers on AR devices, current liveness detection solutions cannot be implemented on AR headsets. For example, the approach proposed in [4] can fight replay attack by reusing a pair of microphone and loudspeaker as a Doppler radar. However, this system requires that both the loudspeaker and the microphone should be in front of the user’s mouth during the speech, which is hard to be ensured on AR headsets.

Considering the limitations of current solutions, we propose a voice-spoofing defense system for AR headsets by leveraging the internal body propagation of human voices. Our system determines whether the voice is from the person who is using the AR headsets by leveraging: 1) Both the internal body propagation and the air propagation of human voices; 2) A tiny and low-cost contact microphone
to collected internal body voice. First, human voices propagate through both the air and the internal body (skull). If two voices are from the same person, they should share common features in the frequency bands of human voices. Second, by attaching a contact microphone on the user’s head, we are able to collect the voice propagating only through the internal body. The small contact microphone can be easily integrated into existing AR headsets. To achieve our goal, we solve two challenges in the design of our system. First, the signal-to-noise ratio (SNR) of the voice propagating through the internal body is still low, which makes it hard to extract voice features from the raw time-domain signals. To address this issue, we transform the signal from the time domain to the time-frequency domain and leverage spectrogram enhancement techniques to extract the voice from raw signals. The second challenge is to measure the correlation and similarity between the internal body voice and the air voice of the user. In order to robustly measure the correlation and similarity between the two voices, we match high-energy blocks that exist in both spectrograms of two voices.

3.2 Preliminary

3.2.1 Human voice production and propagation

As shown in Fig. 3.1(a), the mechanism for producing the human voice can generally be subdivided into three parts: the lungs, the vocal cords, and the articulators (e.g. lips and tongue). The lung first produces adequate airflow and air pressure to vibrate vocal cords. The vocal cords vibrate and chop up the airflow from the lungs into audible pulses that form the laryngeal sound source. Then, the length and tension of the vocal cords are adjusted to produce ‘fine-tune’ pitch and tone. The articulators consisting of tongue, palate, cheek, lips further filter the sound generated from the larynx to strengthen it or weaken it. After the voices are produced
by the human vocal system, they mainly propagate through two media, as shown in Fig. 3.1(b). First, the voice propagates via the air and reaches the microphone, which is common for the use case of current voice input. Besides propagating through the air, the voice can also propagate through the speaker’s internal body, and that is why a person’s voice sounds different to them when it is recorded and played back. Although the tone of the voice received through the internal body is lower than that of the voice received through the air due to the special propagation medium, two voices should have a strong correlation and a lot of information shared. For the attacker who wants to issue a fake voice command obstruct the victim’s experience, the attacker’s voice reaches the AR device only through the air. Therefore, the internal body voice of the victim should not have much-shared information with the air voice.

Strong attackers can also use high-quality loudspeakers and recorders to break voice-based authentication. The loudspeakers usually use an electromagnet to translate an electrical signal into an audible sound. The electromagnet is a metal coil that creates a magnetic field when there is an electric current flow through it. When electrical pulses pass through the coil of the electromagnet, the direction of the magnetic field is frequently changed. Also, there is a permanent magnet fixed firmly into the loudspeaker. With rapidly changing magnetic field, the coil is attracted to and repelled from the permanent magnet. As a result, the cone attached on the coil will vibrate back and forth, pumping sound waves into the surrounding air and the smartphone’s speaker. Since the replay attacker can only record and replay the air voice of the victim, there is no internal body voice during the replay process. Moreover, since the internal body voice of a person is different from those of others even for the same word, a stronger replay attacker cannot impersonate the victim’s internal body voice by wearing the AR headset and saying the same words.
3.2.2 Piezo contact microphone

As shown in Fig. 3.2(a), contact microphone is a form of microphone that senses audio vibrations through contact with solid objects. Unlike normal air microphones, contact microphones are almost completely insensitive to air vibrations but transduce only structure-borne sound. By attaching a contact microphone near the speaker’s temple, we are able to collect the voice that propagates mainly through the body of the speaker. In addition, contact microphones have a wide frequency response, as shown in Fig. 3.2(b). Since the voiced speech of a typical adult will have a fundamental frequency for up to 255 Hz [34], the contact microphones have enough capability to capture the internal body voice.

3.2.3 Attack model

In our attack models, a malicious user aims to either spoof the voice verification system on the AR headset or obstruct the normal use of voice-based input. The capability of the attacker is limited in the sense of:

**Obstruction attack for voice commands.** In obstruction attack, a malicious user who can show up closely around the normal user aims to issue a voice command with high volume. For example, the malicious user can issue a ”remove” voice
command to clear the victim’s virtual objects. The malicious user can also issue a voice command to display redundant information in the field of vision of the normal user, which poses threats if the normal user needs clear sight (e.g. the normal user is driving). During the attack, we assume that the victim is not using the voice input, otherwise, the victim’s voice is expected to overshadow that of the attacker.

**Replay attack for voice-based authentication.** In this type of attack, we assume that an attacker can physically access the victim’s headset in the case of not being noticed. Moreover, the attacker can record or morph the victim’s voice and replay it to voice-based authentication system using loudspeakers. To achieve better attack performance, we assume that the attack can produce the corresponding internal body voice by shadowing the replayed voice of the victim.

### 3.2.4 Use case

In order to successfully defend AR users against two types of attacks, our system requires users to attach a contact microphone around the temple. Since the AR users need to wear the AR headset, this condition can be easily satisfied by integrating the contact microphone into the frame of the AR headset. We leverage the contact microphone to capture the internal body voice and use the existing normal microphone on current AR devices to collect the air voice. The distance between the normal microphone and the user’s mouth is about 10 centimeters. Since the distance is pretty short, the time delay between two audio signals is less than 13 samples when the sampling rate is 44,100 samples per second. While speaking, the user can be in any stationary posture, like sitting and standing.

### 3.2.5 Feasibility study and challenges

In order to defend against two attacks we consider, we need to fully leverage the relationship between voices through the air and the skull. Fig. 3.3 shows the
spectrograms of two voices when the user says “Five”. We can observe two facts: 1) There exists a strong correlation between two voices in both the time and frequency domains. If a normal user interacts with the headset using voice, we should observe a voice through the internal body is produced at the same time. 2) The voice that propagates through the internal body only reserves partial low-frequency features (200 Hz to 2000 Hz). If we can see high-energy blocks in the spectrogram of internal body voice, we should see high-energy blocks at the same location in the spectrogram of the air voice. These observations illustrate that it is feasible to defend against two attacks by measuring the correlation and similarity of two voices.

To achieve our goal, we solve two challenges in the design of our system. First, even with amplifier, the SNR of the voice that propagates through the internal body is still low, which makes it hard to extract voice features from the raw time-domain signals. To address this issue, we transform the signal from the time domain to the time-frequency domain and leverage spectrogram enhancement techniques to extract the features of two voices from their raw signals.

The second challenge is to measure the correlation and the similarity between the internal body voice and the air voice. This is difficult because both voices have different capabilities for capturing users’ voices. More specifically, the internal
body voice only contains partial low-frequency features, but it is insensitive to environmental noise. The mouth voice reserves much more features, but it can be easily influenced by environmental noise. In order to robustly measure the correlation between two voices, we first convert the two voices to spectrograms in the domain of three dimensions: time, frequency, and energy. The correlation and the similarity of two voices are measured by matching high-energy blocks that exist in both spectrograms.

3.3 System design

3.3.1 System overview

The key idea underlying our system is to fully leverage two propagation paths of the human voices. When the AR user says a voice command, the normal microphone will capture the user’s voice that propagates through the air, and the contact microphone on user’s head can record the voice that only propagates through the user’s body. By comparing the information in two voices, our system can determine whether the voice is from the normal user or from two types of attackers. For a new AR user, there are two stages to use the system. In the training stage, the new user is asked to say a few words using our system. These training instances are used to quickly build a classifier. After the training stage, the system is ready to be used. In the testing stage, our system will check whether the command is from the normal user who is using the AR headset using the trained classifier. If the voice is from the normal user, the user can interact with AR headset normally. Otherwise, the voice command will not be parsed to the AR headset for further verification.

The pipeline of data collection and processing is shown in Fig. 3.4. After collecting the user’s voices at two channels, we first segment the voice for each word to remove the interval between neighboring words. For the voice signals of each pair of words,
we transform the signals from the time domain to the time-frequency domain. Since both raw voice signals contain background noise, we further leverage spectrogram enhancement techniques to remove the noise and extract the information of the voices. Then, we measure the correlation between two enhanced spectrograms of each pair of words. If the correlation exceeds a threshold, the pair of signals is further checked for the second round. In the second round, we measure the similarity of two spectrograms. Here the similarity is defined as the proportions of shared information between two voices. If the proportions of shared information fit the trained classifier, the word is regarded to be from the normal user. To tolerate wrong classification results, the final detection result of a sentence (voice command) is determined by a voting procedure of all words in it. Only if the number of votes that represent the voices are from the normal user exceeds the voting threshold, the voice source is regarded as the normal user.
3.3.2 Word segmentation and spectrogram generation

Each audio signal includes two parts: the voice and background noise. The voice contains abundant features of the user’s voice, while the noise only records the acoustic noise in the background. In our system, we only focus on the user’s voice in order to reduce the influence of the acoustic noise in the background. Since the voice recorded by the normal microphone has much more features of the user’s air voice, we segment each audio sample into different words by performing Hidden Markov Model (HMM) based word segmentation techniques [35] on the air voice.

Also, we need to find features to measure the relationship and differences between two voices collected from two microphones to distinguish whether the voice is from a normal user. In order to capture features on time-frequency domain, we perform STFT on each word and each audio sample with a window size of about 22 ms based on:

\[
X(\tau, \omega) = \sum_{n=t_s}^{n=t_e} x[n]w[n - \tau]e^{-j\omega n} \tag{3.3-1}
\]

where \(\tau\) is the time axis, \(\omega\) is the frequency axis, \(x[n]\) is the an audio signal in the time range \((t_s, t_e)\), \(w[n]\) is the window, and \(X(\tau, \omega)\) is a complex function representing the phase and magnitude of the signal over time and frequency. Then, for each time and frequency frame, the spectrogram of the complex function \(X(\tau, \omega)\) is computed based on:

\[
E[f, t] = |X(\tau, \omega)|^2 \tag{3.3-2}
\]

where \(E[f, t]\) is the power of \(f^{th}\) frequency band and \(t^{th}\) time frame. \(f\) and \(t\) are positive integers with range \(1 \leq f \leq M\) and \(1 \leq t \leq N\). \(M\) is the number of frequency frames in generated spectrogram, and \(N\) is the number of time frames. In our system, we further convert power measurements to decibels for data processing.
3.3.3 Spectrogram enhancement

In real usage scenarios, the contact microphone cannot touch the skull directly, which leads to low SNR of recorded internal body voice even with an amplifier. The air voice is also influenced by background noise. To extract features from both voices, we leverage spectrogram enhancement techniques to extract high-energy clusters that are only produced by the user’s voice on the generated spectrograms. After obtaining the spectrogram of each word, we first apply frequency domain denoising method by subtracting the noise floor (non-voice part) from the spectrogram. Since the microphone of the AR headset is close to the user’s mouth, most power should distribute on the voice part as shown in Fig. 3.5(a). Therefore, the noise floor is set to 80% of the power in the spectrogram of each word. If the resulting magnitude becomes negative after subtraction, we set it to zero. Second, since the internal body voice collected from contact microphone contains strong noise under 800 Hz, we only reserve the spectrograms from 800 Hz to 2000 Hz for the following analysis. As shown in Fig. 3.5, most of the noise are removed from the spectrogram, and only the information of the voice are reserved.
3.3.4 Feature extraction and classification

Since two voices are generated from the same vocal system at the same time, we should be able to observe strong correlations between them for a normal user. Ideally, the subtraction of two spectrograms should be zero. In our system, we measure the correlation between two spectrograms instead of directly calculating the differences between them for two reasons. First, both voices have different capabilities for capturing users’ voices. More specifically, the internal body voice only contains partial low-frequency features, but it is insensitive to environmental noise. The mouth voice reserves much more features, but it is easy to be influenced by environmental noise. Second, even if two microphones are synchronized, there may still exist small synchronization bias in the collected voices. Similar to one-dimension cross-correlation measurement, given two spectrograms $S_1$ and $S_2$, we measure the correlation between $S_1$ and lagged copies of $S_2$ as a function of the horizontal lag $i$ and the vertical lag $j$. For this copy, assume that $S_1$ and the lagged copies of $S_2$ have an overlapped area of size $M \times N$, the correlation coefficient of the specific shift is:

$$\text{Corr}[i,j] = \sum_{k=1, l=1}^{k=M, l=N} O_1[k,l] \times O_2[k,l]$$

(3.3-3)

where $O_1$ is the overlapped part of $S_1$, and $O_2$ is the overlapped part of $S_2$. Hence, the positive integer $i$ is from 1 to $2M - 1$, the positive integer $j$ is from 1 to $2N - 1$. The best matching of two spectrograms is found if corresponding correlation coefficient is maximal. In our system, two voices are highly correlated, and the highest correlation coefficient must appear around the center of correlation matrix $\text{Corr}$, as shown in Fig. 3.6. Based on this observation, a word is detected to be from a live user if

$$\frac{|j - M|}{2M} < \lambda \quad \text{and} \quad \frac{|i - N|}{2N} < \lambda$$

(3.3-4)
where $\lambda$ is the decision threshold.

A pair of spectrograms that satisfy Equation 3.3-4 cannot ensure that two voices are from the normal user. Although we know two spectrograms are highly correlated from Equation 4, it is not clear how much information or features are shared between two spectrograms. Therefore, we further measure the similarity between two voices by finding the proportion of shared information. Based on our observations, the amount of shared information should make up a large proportion of either of two voices. In other words, if an entry is non-zero in the spectrogram of internal body voice, it is very likely to be non-zero in that of the mouth voice, and vice versa. To quantitatively describe how similar two spectrograms are, we first use the measured lags to calibrate our synchronization to get the best match. For each word, the proportion of the shared information that is in $S_1$ is defined as:

$$P_1 = \frac{\text{Sizeof} \{ (i, j) | S_1[i, j] > 0 \, \&\, S_2[i, j] > 0 \} \}}{\text{Sizeof} \{ (i, j) | S_1[i, j] > 0 \} } \quad \text{(3.3-5)}$$

Similarly, the proportion of the shared information that is in $S_2$ is defined as:

$$P_2 = \frac{\text{Sizeof} \{ (i, j) | S_1[i, j] > 0 \, \&\, S_2[i, j] > 0 \} \}}{\text{Sizeof} \{ (i, j) | S_2[i, j] > 0 \} } \quad \text{(3.3-6)}$$
The similarity between two voices is defined as the smaller one of $P_1$ and $P_2$.

Fig. 3.7(a) shows the values of the proportion of the shared information for both normal user and attacker. Ideally, the proportion of the shared information should be high for normal users. However, since different users have different speaking habits (e.g. different speeds of speech and different accents), the proportions of shared information may not always be a high value. Also, unpredictable noise during data collection may also influence the final results. Therefore, it is hard to determine the legitimacy of the speaker using a fixed threshold on each dimension. By studying the data distribution on 2-dimensional feature hyperplane, we find that data of normal users lies on a straight line, while that of attackers is far away from the line. Fig. 3.7(b) shows the distribution of distances from the data point to the straight line that is fitted using the normal user’s training data. We can see that over 95% of the normal user’s data points have the distance less than 2, while over 85% of the attacker’s data points have the distance larger than 2. This fact enables us to detect the legitimacy of the speaker by calculating the distance from the data point to the line that fits the training data. After collecting several training data from the user, we first fit a straight line using least squares, as the yellow line in Fig. 3.7(a). A word is considered to be from the normal user if

$$\frac{|aP_1 + bP_2 + c|}{\sqrt{a^2 + b^2}} < \gamma \quad (3.3-7)$$

where $P_1$ and $P_2$ are the features calculated using Equations 3.3-5 and 3.3-6, $a$, $b$, and $c$ are coefficients of a straight line $ax + by + c = 0$. $\gamma$ is the decision threshold and is set to the 95% largest distance of normal user’s training data. A word is considered to be from a normal user if and only if both of Equations 3.3-4 and 3.3-7 are satisfied.
3.3.5 Decision combination

AR users usually speak a sentence or passphrase that consists of multiple words to AR headsets. For example, the general voice authentication systems ask the user to speak a 6-digit passphrase. In order to give an accurate detection result for each sentence, we need to combine the results of multiple words after getting the correlation and similarity measurement of each of them. In a voting procedure, three questions need to be answered: 1) Who should be eliminated from voting; 2) What is the weight of each player; 3) What is the minimum number of votes needed to pass a vote. To answer the first question, the voter whose data cannot satisfy either of Equations 3.3-4 and 3.3-7 is eliminated from voting. Second, since both $P_1$ and $P_2$ reflect the propagations of shared information between two voices, the word with high values of $P_1$ and $P_2$ should have a higher weight for voting. Therefore, for each word in the voting procedure, we let the smaller value of its $P_1$ and $P_2$ be its weight. Third, to accurately reject the attacker and accept the normal user, for a sentence or a voice command with $n$ words, the minimal number of votes is set to $0.2 \times n$. If there is no result whose number of votes exceeds $0.2 \times n$, the user is regarded as the attacker.
3.4 Evaluation

3.4.1 Hardware

Our system consists of two components: a testbed for collecting internal body voice and a smartphone for collecting air voice. We implemented our testbed using a Raspberry Pi 3, an iRig HD 2 soundcard, and an AXL contact microphone. Besides, we used a Nexus 5 to collect user’s air voice and transmit it to the Raspberry testbed through WiFi. Both the smartphone and the Raspberry testbed were synchronized to the same server. Our experiments involved 8 volunteers (5 males and 3 females), and all of them were asked to repeat saying sentences of different lengths to our system. In order to make sure the contact microphone can capture the internal body voice during the data collection, we attached the contact microphone on a hat and asked each volunteer to wear it. Each volunteer wore the hat in their own way and was in a comfortable posture they prefered. For data analysis and processing, the data was then transmitted to a desktop computer with Intel(R) Core(TM) Devil’s Canyon Quad-Core i7-8700K @ 4.00 GHz CPU and 16 GB of RAM.

3.4.2 Overall performance

We first evaluated our system performance for normal users and against two types of attacks. In this experiment, we used the voices of 40 words collected from the
normal user as the training data. The correlation threshold $\lambda$ was set to 0.1, and the distance threshold $\gamma$ was set to the 95% largest distance of normal user’s training data. We asked each user to say a 5-word sentence 50 times. Moreover, we repeated this procedure for 10 times to study the variance of true acceptance rates of different volunteers, and the experimental results are shown in Fig. 3.9. We can observe that our system can correctly accept the normal user with mean accuracy of 97% for all users. Even in the worst case, our system can still achieve a high accuracy of 92.3% for normal users. By studying normal users’ data that is wrongly rejected, there are two main reasons that degrade the performance. First, there are two volunteers who speak softly, which makes their voice is easier to be covered by background noise. Second, volunteers’ activities may cause slight movement of the hat, which introduces high-energy noise to the internal body voice and reduces similarity between two voices.

We further evaluated how accurately our system can reject two types of attacks. To collect the data for the obstruction attack, we let a volunteer speak loudly while the normal user (another volunteer) was wearing the hat. To collect the data for the replay attack, we used a Nexus 6 smartphone record the victim voice at a distance of 0.5 meters. Then, we used the loudspeaker of a smartphone to replay the victim’s voice to our system. At the same time, the replay attacker said the same sentence to our system while wearing the hat. Moreover, we made sure the genders of the victim
and the replay attacker are the same. We leveraged the fitted straight line for the victim to determine the legitimacy of the attacker’s data, and the results are shown in Fig. 3.9. We can see that our system can provide high accuracy against both types of attacks. More specifically, our system can provide a mean accuracy of 99.2% and 98% for defending the obstruction attack and replay attack, respectively. The accuracy of successful defenses is not 100% for two reasons. First, some internal body voices in the training dataset contained noise, which increased the distance threshold. Second, the slight movement of the user’s head may also introduce random high-energy influence to the spectrogram. In rare cases, the filtered spectrogram of noise was similar to that of some words (e.g. “eight”). As a whole, our system can provide high-security protection for users against obstruction attack and replay attack while still ensuring good user experience for normal users.

3.4.3 Influence of training dataset size

In practice, we want the number of training data to be as small as possible to reduce the training cost for new users. Therefore, we evaluated how much training data is needed by our system in order to provide both high-security protection and good user experience. Fig. 3.10(a) shows the system performance with different sizes of the training dataset. We can see that the average accuracy for the normal user is improved...
a lot by using more data for training since we have more knowledge about the distribution of the normal user’s data. By contrast, the average accuracy of successful defense against either of two attacks is almost the same by using different numbers of training data. The reason behind this is that the distribution of the attacker’s data is significantly away from that of the normal user. Therefore, our system can accurately reject two types of attacks even if the training data is limited. Overall, our system can provide both high-security protection and good user experience after collecting the voices of 20 words from the normal user, which is low-cost and easy to be used for new users.

3.4.4 Influence of the ratio of $\gamma$ relative to the maximum distance

In our default experimental setting, the distance threshold $\gamma$ is set to the 95% highest distance in the training data. In real scenarios, there is a trade-off on
determining the value of $\gamma$. A small distance threshold can provide extremely high true rejection rate against two types of attackers, but it also makes it hard for normal users to use our system. A high distance threshold can ensure good user experience, but more attackers are wrongly accepted. In this subsection, we study what is the proper value of $\gamma$ for different users. Fig. 3.10(b) shows the system performance with different values of $\gamma$. It is clear that the average accuracy for normal users rises with the increase of $\gamma$, while the average accuracy of successful rejection drops. When $\gamma$ is the 95% highest distance in the training dataset, the true acceptance rate and the true rejection rate are nearly equal. Therefore, we let the $\gamma$ be equal to the 95% highest distance in the training dataset to balance the need for security protection and user experience.

### 3.4.5 Influence of voting threshold

The performance of our system relies on a successful voting procedure. Hence, a proper voting threshold is important. Similar to the distance threshold, there is also a trade-off on determining the value of the voting threshold. If the voting threshold is too small, all normal users can be accepted, but some attacker may also be wrongly regarded as the normal user. If we assign a high value to the voting threshold, all attackers can be successfully rejected, but the user experience of normal users is ruined. In this subsection, we study what is the proper value of the voting threshold. Here we use $c \times n$ to represent the voting threshold where $c$ is a constant and $n$ is the number of words in a sentence (voice command). We evaluated the performance for 5-word sentences using the default parameters and adjusted the value of $c$, and the results are shown in Fig. 3.11(a). We can see that the average accuracy for normal users drops rapidly when $c$ is larger than 0.2. Moreover, our system can provide good security protection after $c$ reaches 0.2. Therefore, we let the $c$ be equal to 0.2 in our default system setting.
3.4.6 Influence of sentence length

We also evaluated the system performance for sentences of different lengths. Here the sentence length means the number of words in the sentence. When the length of the sentence is short, the wrong classification of a few words may dominate the voting procedure and give the incorrect detection result. For longer sentences, the voting procedure can tolerate a few wrong predictions by involving more players. In this subsection, we study what is the minimum sentence length to ensure good security protection and user experience, and the results are shown in Fig. 3.11(b). We can see that the system performance is improved with more number of words in a sentence. When the sentence length is 6, our system can provide average accuracy of about 100% for both accepting normal users and rejecting attackers. Moreover, with more numbers of words in a sentence (voice command), the variance of both true acceptance rate and true rejection rate are reduced, as shown in the error bar in Fig. 3.11(b). This fact implies that the robustness of our system is improved by saying a voice command with more words. Considering most voice commands supported by current AR applications have lengths of at least 3 words (e.g. open the navigation), our system can provide good enough security protection and user experience for them.

3.4.7 Influence of background noise

Since our system records the air voice using a normal microphone, the background acoustic noise (e.g. conversation or music) may cover the features in the air voice and degrade the performance for normal users. To evaluate the robustness of our system against background noise in terms of accepting normal users, we asked one volunteer to speak a 5-word sentence to our system. During the data collection, we used two loudspeakers to simulate different noise levels from 45 dB (average home noise) to 70 dB (inside a car at 60 mph). We did not consider greater noise in our evaluation for two reasons: 1) Most voice-based AR applications are not designed
for noise environment (e.g. video call); 2) The performance of voice recognition and authentication systems can also be degraded by strong noise. Fig. 3.11(c) shows the evaluation results. We can observe that our system can achieve a high accuracy of at least 99.5% for all noise levels. We found that the reason why our system can still provide good performance in a noisy environment is that the AR users will subconsciously raise their volume in a noisy environment, which makes the features of their voices more significant than those of background noises. By applying spectrogram enhancement techniques, these background noise can be largely removed.

3.5 Chapter summary

Voice-based interaction is usually used as the primary interaction method for AR headsets due to its good user experience and performance. AR users rely on accurate and secure voice input to communicate with AR headsets. However, recent researches have shown that an attacker can easily perform various attacks with the help of state-of-the-art voice synthesis/conversion software. To secure the voice input on AR headsets, we propose a robust and low-cost solution for defending against voice-spoofing attacks on AR headsets with high accuracy. Our system leverages a contact microphone to record the internal body propagation of the voice. A user legitimacy is determined by measuring the correlation and similarity between the internal body voice and air voice. To our best knowledge, our system is the first to protect the voice input for AR headsets. This work in this chapter is published in the second paper of the publication list. We summarize our contributions as follows:

- We show that it is feasible to capture the internal body propagation of human voices using a low-cost contact microphone. We also present an approach to extract voice features from noisy internal body voice.
- We propose a robust and low-cost solution for defending against voice-spoofing
attacks on AR headsets with high accuracy. To our best knowledge, our system is the first to protect the voice input for AR headsets.

• We develop a prototype and conduct comprehensive evaluations. Experimental results show that our system can successfully defend against obstruction and replay attacks with an accuracy of at least 98%.
CHAPTER 4

DEFENSE SYSTEMS 3: FACE FORGERY DETECTION

With the rapid popularity of cameras on various devices, video chat has become one of the major forms of communication, such as online meetings. However, recent progress in face reenactment techniques enables attackers to generate fake facial videos and use others’ identities. To protect video chats against fake facial videos, we propose a new defense system to significantly raise the bar for face reenactment-assisted attacks. Compared with existing works, our system has three major strengths. First, our system does not require extra hardware or intense computational resources. Second, it follows the normal video chat process and does not significantly degrade the user experience. Third, our system does not need to collect training data from attackers and new users, which means it can be quickly launched on new devices. We develop a prototype and conduct comprehensive evaluations.

4.1 Introduction

In the past few years, thanks to fast internet speeds and the powerful processing capacity of personal electronic devices, video chat has become a major form of communication. Compared with text-based or audio-based communication, video chat enables users to observe the real emotions and activities of each other without physically being together, which makes the information delivered more accurate and the relationship establishment more efficient. Therefore, many video chat software
(e.g. Skype [36] and WebEx [37]) are released for various applications, such as conference meeting, interviewing, and making friends. Based on a report from Statista, the estimated number of Skype users is expected to be 1.67 billion in 2020 [38].

There are two major channels in real-time video chat: image and audio. By default, both channels are regarded as real information since they are generated in real-time, which is why video chat is used as an alternative way to validate the identity of a user in practice. However, since a malicious user can easily get the victim’s videos and voice from social networks, both channels can be counterfeited with the development of AI-assisted techniques. For example, the recorded voice of the victim can be replayed to pass through current voice-based authentication systems. Similarly, recent research in face reenactment shows that the facial expressions on one face can be transferred to any other face in real-time. These facts enable the malicious user to easily use the victim’s identity, which poses a serious threat to legitimate users. Even if the voice replay attack can be efficiently countered by using voice liveness detection techniques [39, 4, 32, 40], attackers can still fool legitimate users by generating fake image channels.

To defend against fake face videos, various face liveness detection systems are designed using either artifact detection-based methods [41, 42, 43, 44, 45] or challenge-response-based methods [46, 47]. The basic assumption of the artifact detection-based method is that fake facial images must have imperfect artifact detections. By extracting proper features, the fake facial images can be detected using various classification models. However, in order to gain enough knowledge for building a robust classifier, artifact detection-based methods have to collect fake videos in advance, which usually involves significant training. Moreover, artifact detection usually requires lots of computational resources to achieve better feature extraction and classification, which is not available on resource-limited devices. Challenge-
response-based methods are based on the nature of human activities. For example, FaceLive can detect the media-based facial forger by correlating the head movement measured by motion sensors and head pose change recorded in videos [46]. However, the face reenactment attacker can still easily break FaceLive by faking the sensor data since it can have enough knowledge of the target video. Moreover, since the detection is done on the attacker side, the attacker can even send the legitimate user a wrong detection result. Recently, Tang et al. [47] proposed a new liveness detection method by randomly flashing pre-designed pictures (e.g. white and black scenes) on a screen and analyzing the face-reflected light. Nevertheless, their work also relies on a neural network for accurate classification. Moreover, the flashing pictures replace the original video frames, which will degrade the user experience between two legitimate users.

Considering the limitations of existing solutions, we propose a defense system for real-time video chat against fake facial videos generated by face reenactment techniques. As shown in Fig. 4.1, our system requires no extra sensors except the screen and camera that are available on all videotelephony devices. Specifically, the screen is used to emit light signals, and the camera works as a sensor to measure the relative luminance (simplified as luminance [48] in this chapter) of the light that is reflected from the untrusted user’s face. The key insight behind our system is that the luminance of the face-reflected light is proportional to that of the screen light for a legitimate user. Since the face reenactment attacker cannot generate the real-
time face reflection in a photo-realistic fashion, the legitimate user can detect the face forgery by: 1) introducing luminance changes in the transmitted video through changing the area of light metering; and 2) measuring the correlation between the luminance changes of the screen light and face-reflected light.

To achieve our goal, we solve three major challenges in the design of the system. The first challenge is to robustly extract the luminance information of face reflection from the videos. To address this issue, we leverage the facial landmark detection algorithm to locate the lower part of the nasal bridge as the area of interest and calculate the luminance information using only the color information within this area. Second, the luminance signals are noisy and cannot be directly used for correlation measurement. To solve this problem, we remove the noise components using signal processing techniques and extract the significant light change from filtered signals. The last challenge is to extract useful features from the filtered signal and build a classifier for robust and accurate detection. In our system, we extract four features that describe the luminance change behavior and trend from the filtered signal. A local outlier factor-based classifier is trained on selected features for the final decision.

Compared with existing works, our system has three major strengths: 1) low-cost: our system does not require extra hardware or intense computational resources; 2) good user experience: since the luminance change in the transmitted video is made by controlling the exposure level, both users can still see each other’s face with only limited loss of video information; and 3) zero training effort: our system does not need to collect training data from either a new user or attackers, which means our system can be quickly launched on new devices.
4.2 Literature review

4.2.1 Face reenactment techniques

Face reenactment is a group of techniques that can transfer facial expressions from a source face to a target face. Traditionally, the work of face reenactment is done offline due to high computational resource required [49, 50, 51, 52, 53, 54, 55, 56, 57]. For example, Garrido et al. [49] proposed a system that can transfer facial expressions when both the source and target faces are from the same person. They further improved their work to transfer facial expression among different people in [50]. Recently, there has been research supporting online face reenactment, opening this technology up to a wider range of applications. A famous work Face2Face [1] is proposed to achieve online face reenactment with about 27.6 frames per second. This fact implies that face reenactment techniques can be executed during real-time video recording, which largely improves attackers’ capability for launching face forgery attack in real-time video chat.

4.2.2 Face liveness detection in videos

Considering the serious threats introduce by fake facial videos, various systems are proposed to detect fake faces in videos, which is also referred to as face liveness
detection. Overall, current fake face detection methods can be grouped into two categories: artifact detection-based and challenge-response-based. Artifact detection-based methods aim to exploit artifacts that are introduced during the synthesis process using both low-level and high-level features [41, 42, 43, 44, 45, 58]. However, all existing artifact detection-based methods have two major limitations. First, they need to collect enough training from target face forgery techniques, which is usually expensive and hard to satisfy in practice, particularly for unknown techniques. Second, the detection procedure requires intensive computational resources (e.g. a graphics processing unit), which is not suitable for battery-limited devices.

Different from artifact detection-based methods, challenge-response-based methods leverage the nature of human activities (e.g. head movement [46]). However, the face reenactment attacker can still easily break FaceLive by faking the data of motion sensors in advance since it can have enough knowledge of the target video. Moreover, since the detection is done on the attacker’s end, the attacker can even send the legitimate user a wrong detection result. A recent work detects a fake face during face authentication by randomly flashing well-designed pictures on a screen [47]. However, their challenge-response-based method has to alter the displayed content on the screen, which largely influences the user experience of video chat.

4.3 Preliminary

4.3.1 Face forgery using face reenactment

The goal of face reenactment techniques is to animate the facial expressions of the target video by a source actor and re-render the manipulated output video in a photo-realistic fashion. Fig. 4.2 shows an example of face reenactment technique reported in [1]. We can see that the facial expression in the source video is transferred to the person in the target video with high quality. Compared with other real-time
face forgery techniques (e.g. face swapping), face reenactment creates fewer artifacts while achieving high frame rates (up to 47.5 Hz in [56]). For a legitimate user in video chat scenarios, it is hard to detect the face reenactment attacks with high accuracy. Although face reenactment techniques have made great success in face forgery, their nature also gives us the insight to defend against them. Since face reenactment techniques only focus on transferring the facial expression, the luminance change of the output video is the same as the target video, which means the attacker cannot have the correlated luminance change of face-reflected light. Even if the face reenactment attacker can use the source actor to observe the luminance change and generate the change in the output video, the extra computational overhead will largely reduce the frame rate make real-time attack infeasible.

4.3.2 Light metering of digital cameras

In general, the camera controls the shutter speed and aperture by predicting how much light is actually hitting the subject. Current cameras provide users with various ways to meter light. Among them, spot and multi-zone metering modes are most used and widely available. In multi-zone metering, the camera measures the light intensity in multiple points in the scene and then combines the results to find the setting for the ideal exposure. Therefore, multi-zone metering can produce balanced exposure for most scenes and is used as the default mode for most cameras. Alternatively, with spot metering, the camera will measure only a very small area of the scene. By default, this small area is at the center of the scene, but the user or application can easily select a different off-center spot. If the spot is moving from a relatively low-luminance area to a high-luminance area, the camera will let less light in, which leads to a diminished brightness in the darker area. Similarly, if the spot is moving from a relatively high-luminance area to a low-luminance area, the luminance of the scene rises. Hence, by moving the metering spot between high-luminance and low-
luminance areas, the legitimate user can easily control the overall luminance of its
video. Since the exposure only changes the brightness of each pixel, this method can
reserve partial information (e.g. the face of the legitimate user) in the scene, which
ensures a certain level of user experience.

4.3.3 Face reflection of screen light

When the untrusted user watches the legitimate user’s facial video, the camera can
capture the screen light that is reflected by the face of the untrusted user. Here, we
model the face reflection of screen light based on the Von Kries coefficient law [59].
For a single type camera, a diagonal model can be described as:

\[ I_c(x) = E_c(x) \times R_c(x), \quad c \in \{R, G, B\}, \quad (4.3-1) \]

where \( x \) is a pixel on the face, \( c \) is the light with different colors (red, green, and
blue), \( I_c \) is the luminance of corresponding color, \( E_c \) is the illuminant spectral power
distribution of the screen light on \( x \), and \( R_c \) is the reflectance of pixel \( x \). Therefore,
if we focus on a pixel with the same reflectance and change the light luminance, then
we have:

\[ \frac{I_c(x)'}{I_c(x)} = \frac{E_c(x)'}{E_c(x)}, \quad c \in \{R, G, B\}, \quad (4.3-2) \]

where \( I_c(x)' \) and \( E_c(x)' \) are the luminance and illuminant spectral power distribution
after the change of screen light. From this equation, we can observe that the
luminance of the face reflection is proportional to that of the screen light, which
serves as the basic insight of our system.
4.3.4 Feasibility study

To achieve our goal, we first show that the luminance of the face-reflected light is highly correlated to that of the screen light. Specifically, we made a video that flashes between white and black with a frequency of 0.2 Hz and displayed this video on a Dell 27-inch Light-emitting diode (LED) Monitor. We asked a volunteer to sit in front of the monitor while using the front camera of an iPhone 7 to record his facial video. Fig. 4.3 shows the faces when the screen shows black and white color, respectively. We can clearly observe that the luminance of the face-reflected light increases when the color changes from black to white. As a reference, the luminance value of the nasal bridge increases from around 105 to around 132. Moreover, this fact is true for all types of screens including LED, liquid crystal display (LCD), and organic LED (OLED) since they all reduce the amount of emitted light when displaying darker scenes. This simple case implies that the luminance of the face-reflected light does change proportionally to that of the screen light, which shows the possibility of detecting fake faces using the correlation between two luminance signals.

4.3.5 Challenges

Although we can observe the corresponding luminance change of the face-reflected light while the screen’s color changes between black and white, it is still challenging to apply this insight to real video chat scenarios for fake forgery detection. First, the face of the untrusted user will likely be moving in the scene and can be partially occluded by other objects (e.g. hair and sunglasses), which introduces extra noise to the luminance signals of the face-reflected light. To address this issue, our system only extracts the luminance information from the lower part of the nasal bridge since this area can be robustly located using the facial landmark detection algorithm and is the least likely part to be obfuscated.

The second challenge is to obtain the luminance change information from the
noisy luminance measurements. The raw luminance signals contain various types of noise. For example, dynamic scenes in the video will introduce high-frequency noise to the raw luminance signal of the screen light. Additionally, the luminance change in practice is weaker than the ideal case in feasibility study. To remove the noise and robustly locate each luminance change, we designed a series of filters and apply them on the raw luminance signals in order.

The last challenge is to extract useful features from the filtered signal and build a classifier for robust and accurate face forgery detection. To solve this problem, we select four features that describe when and how the luminance signal significantly changes. To reduce the training cost while still ensuring good performance, we build a strong classifier without collecting training data from the attacker and new users using the local outlier factor model.

4.4 System design

4.4.1 Adversary model

In our adversary model, the attacker aims to impersonate others using face reenactment while video chatting with victims. The capability of the face reenactment attacker is limited in the sense that: 1) the attacker has already or is able to set up a video chat connection with the victim; 2) the attacker can generate fake facial videos with high quality in real-time using any face reenactment technique; and 3)
the attacker can redirect the input stream of the current video chat software (e.g. Skype) to the fake facial videos using a virtual web camera. When these tasks are performed, we suppose the victim cannot visibly identify the fake facial video as a forgery.

The objective of our system is to significantly raise the bar to such face reenactment attacks. To break our defense system, the attacker needs to reconstruct the face-reflected light on the fake face with high quality based on the relative locations of the head, camera, and the screen in real-time. For this, the attacker has to: 1) introduce an extra image processing layer for each frame to reconstruct the face-reflected light; and 2) have enough computational resources to ensure the real-time attack. Therefore, our system is difficult to attack.

### 4.4.2 System Architecture

The key idea underlying our system is to measure the luminance correlation between the screen light and the face-reflected light. When a legitimate user is using videotelephony with an untrusted user, the camera can capture the screen light that is reflected by the untrusted user’s face. By comparing the luminance of the screen light and the face-reflected light, we can determine if the face is from a real person or
generated by face reenactment techniques. There are two major phases for using our system: a training phase and a detection phase. In the training phase, our system will learn the decision strategy based on the knowledge in the legitimate users’ data. After that, our system is ready to be used for detection. Our detection methods can be triggered multiple times during the real-time video chat. If the untrusted user is detected as an attacker, an alert will be sent to the legitimate user to avoid further loss.

Fig. 4.4 shows the detailed process of our system in five steps. A legitimate user Alice wants to validate whether the facial video sent from the untrusted user Bob is real or fake. To do this, Alice gets her own facial videos using a camera in step 1 and sends the real-time facial video via the internet to Bob in step 2. On Bob’s side, his device receives Alice’s video and displays it on his screen, which means that the luminance of the screen light largely depends on the content in Alice’s video in real-time. At the same time, as illustrated in step 3, Bob is recording his facial video whose luminance change should be influenced by not only the ambient light in Bob’s environment but also his screen light. By receiving Bob’s video in step 4, Alice can get the luminance information of both Bob’s screen light and Bob’s face-reflected light. In our system running on Alice’s device, we first extract the luminance information in both videos and apply filters to the raw signals to extract only significant light changes.

4.4.3 Luminance extraction

The goal of our system is to detect the liveness of the face in the video by measuring the correlation between luminance signals of the screen light and face-reflected light. Therefore, we first need to robustly extract these two types of luminance information from two videos respectively. Since we are only interested in the overall luminance of the screen light, we first compress each frame of the transmitted video into a single
pixel, and use the luminance value of the compressed pixel to represent the overall luminance of the transmitted video. The luminance of a pixel is defined as:

\[ C = 0.2126R + 0.7152G + 0.722B, \]  

(4.4-3)

where \( C \) is the luminance value calculated using linear Red Green Blue (RGB) values. The coefficient of each color is assigned based on the human visual perception of brightness.

However, not all facial parts can be used to measure luminance changes. For example, the users may blink their eyes or talk during the recording. Such activities will introduce a lot of variances between neighboring frames. Also, users may wear glasses that reflect light from other sources, which will introduce much noise to the luminance measurements. Based on our preliminary study, we find that the lower part of the nasal bridge has the most stable images and is hard to be occluded in most cases. Moreover, the luminance changes caused by different screen lights at this area are easy to detect. Therefore, we extract only the lower part of the nasal bridge from each frame of the video for luminance measurement.

When a legitimate user receives the video from the untrusted user, our system extracts frames with a sample rate of 10 Hz. For each frame, we detect the location of the lower part of the bridge by using a facial recognition API for Python [60]. As shown in Fig. 4.5, the facial recognition API can report four locations on the nasal bridge and five locations on the nasal tip. Since the sampled frames can vary in size depending on camera hardware, we use the locations of the nasal bridge and nasal tip to extract the area of interest. As shown in Fig. 4.5, given the coordinates of the nasal bridge \((a_1, b_1)\) and nasal tip \((a_2, b_2)\), the side length of the interested area is \( l = |b_1 - b_2| \). A square whose center is at \((a_1, b_1)\) is extracted from the frame to calculate its luminance. We use the same methods to get the luminance information
from the area of interest. Fig. 4.7(a) shows the luminance signal generated from the lower part of the nasal bridge, and we can see significant rising edge and falling edge appear when the luminance of the screen light significantly changes (green areas).

### 4.4.4 Preprocessing

As shown in Fig. 4.7(a), the raw luminance signals contain various noise. For the transmitted video, the noise is mainly from the object movement in the scene. For the face reflection in the received video, the noise can be introduced by external light sources. Moreover, the inaccurate face localization can lead to jittering in the interested area, which further influences the luminance extraction of the face reflection. Hence, the raw luminance signals need to be filtered before being used for feature extraction.

Fig. 4.6 illustrates the spectrum of the luminance signals of the face-reflected light. It is clear that high-frequency noise exists across entire frequency bands, while screen light changes influence the luminance of the face-reflected light with low frequency under 1 Hz. Based on this observation, we first use a low-pass filter with a cut-off frequency of 1 Hz. As shown in Fig. 4.7(a), most high-frequency components are removed while the overall trend is retained. In our system, we only consider significant luminance change in both luminance signals for two reasons. First, only significant luminance change in the transmitted video can generate luminance change.
Figure 4.6: The spectrum of luminance signals w/ and w/o screen light change. in the interested area of the received video. Second, the significant luminance changes in the received video are robust to noise and easier to detect. However, it is hard to locate each significant luminance change in the filtered signal since low-frequency noise still exists. To locate all significant light change in the filtered signal, we leverage a moving window with length of 10 samples and calculate the short-time variance within each window. The basic insight is that the low-frequency noise within a window only generates a low variance. Moreover, the variance value in the moving window can reach local maxima in two cases: 1) the luminance rapidly increases to a high value; and 2) the luminance drops to a much lower value. Therefore, each significant luminance change can be located by finding the local maxima in the variance signal.

Nevertheless, the variance signal cannot be directly used for locating significant light change. As shown in Fig. 4.7(b), low-frequency noise can either generate small spikes in the variance signal or split a significant luminance change into multiple lower, neighboring peaks. To remove small spikes, we apply a threshold filter on the variance signal with a cut-off threshold of 2. To group neighboring lower peaks into one significant luminance change, we further smooth the variance signal by applying a moving window with a length of 30 samples and calculating the root-mean-square value in each window. Then, we leverage a Savitzky-Golay filter [61] with a window length of 31 samples using polynomial fitting and a moving average filter with a window length of 10 samples to further smooth the signal, and the result is shown in Fig. 4.7(c). Finally, the traditional peak finding algorithm is applied.
The raw and filtered luminance signal

Cut-off threshold: 2
Split to two small peaks

Variance signal

Significant luminance change

Smoothed variance signal

Figure 4.7: Preprocessing of luminance signals.

on each smoothed variance signal respectively. Since the luminance variation range of the screen light is much larger than that of the face-reflected light, the minimal prominence of the peaks is set to 10 and 0.5 for the screen light and face-reflected light, respectively.

4.4.5 Feature extraction

In our system, we consider both the similarities of luminance change behaviors and the correlation of luminance change trends. The luminance change behavior is a vector where the value of each element is the time when a significant luminance change happens. Therefore, the luminance change behavior focuses on the timestamps when significant luminance changes happen while ignoring the trend of the signal. The
luminance change trend is the smoothed variance signal after preprocessing and is used to describe how the luminance changes over time.

4.4.5.1 Luminance change behavior

If both luminance signals are legitimate, there is a strong correlation between them. In other words, for any significant luminance change in one signal, we can always find a matched luminance change in another one. To quantitatively describe how similar two luminance change behaviors are, we define two behavior similarity metrics \( z_1 \) and \( z_2 \). The proportion of matched luminance changes in the transmitted video \( z_1 \) is defined as:

\[
z_1 = \frac{1}{N} \times F(T, R),
\]

where \( N \) is the number of significant luminance changes in the transmitted video, \( T \) is the preprocessed luminance signal of the transmitted video, \( R \) is the preprocessed luminance signal of the received video, and \( F(T, R) \) is a function whose output value is the number of matched luminance changes in the transmitted video. Similarly, the proportion of the matched luminance change in the received video \( z_2 \) is defined as:

\[
z_2 = \frac{1}{M} \times G(T, R),
\]

where \( M \) is the number of significant luminance changes in the received video and \( G(T, R) \) is a function whose output value is the number of matched luminance changes in the received video. For a legitimate user, both values are expected to be 1 or very close to 1, while the values of an attacker should be close to 0. Fig. 4.8 shows two luminance signals collected from a legitimate user. It is clear that, for each luminance change in one signal, we can always find a matched luminance change in another one,
Figure 4.8: The luminance change in transmitted and received video respectively, which means both $z_1$ and $z_2$ are equal to 1.

4.4.5.2 Luminance change trend

In the luminance change behavior, we only consider when significant luminance changes happen while ignoring the trend of the signal. In the worst case, the attacker’s signal can have the same luminance change behavior but with considerably different shapes of luminance signals. Therefore, besides considering the similarity between two luminance change behaviors, we also evaluate the correlation of their trends. To remove the mismatch introduced by network delay, we first estimate and remove the delay based on the average time difference between matched luminance changes. Since we only consider the trend of the luminance signal instead of absolute values, we further normalize each smoothed variance signal to $[0, 1]$. Then, each signal is cut into two segments with equal length. For each pair of segments of two signals, we leverage Pearson correlation coefficient [62] to measure the correlation of their trends. Specifically, the correlation coefficient $corr(x, y)$ between a pair of signal segments is defined as:

$$corr(X, Y) = \frac{1}{L} \sum_{i=1}^{L} \frac{(x_i - \bar{X})}{\sigma_X} \frac{(y_i - \bar{Y})}{\sigma_Y}$$  \hspace{1cm} (4.4-6)
where $L$ is the number of segments, $X = (x_1, x_2, \ldots, x_L)$ and $Y = (y_1, y_2, \ldots, y_L)$ are a pair of normalized signal segments of length of $L$, $\bar{x}$ is the mean value of $x$, $\bar{y}$ is the mean value of $y$, and $\sigma$ represents the standard deviation. The value ranges of $\text{corr}(x, y)$ is $[-1, 1]$. Ideally, $\text{corr}(x, y)$ should be 1 if two smoothed variance signals are positively correlated. In other words, the larger the $\text{corr}(x, y)$, the more positive correlation exists between two smoothed variance signals. Since we have two correlation coefficients calculated from two pairs of segments, we only use the smaller one as the third feature $z_3$. Besides, we also use the maximum dynamic time warping (DTW) distance (denoted as $z_4$) between each pair of segments as the fourth feature to describe the correlation of luminance change trends. Since the range of $z_4$ is much larger than the other three features, we divide it by 30 to reduce its influence in the classification.

4.4.6 Fake video detection

4.4.6.1 Fake video detection for a single video clip

To do the classification, a naive idea is to collect training data from both legitimate users and face reenactment attackers. However, it will involve much training cost to collect data from every new legitimate user. Moreover, it is even harder to get the
data from all possible face reenactment attackers. Therefore, we need to build a
classifier that can have good classifications performance using only the data of a
limited number of legitimate users. In our system, we build a strong classifier using
the local outlier factor (LOF) model [63] due to its fewer requirements on parameter
adjustment and good performance. Specifically, the dataset sent to the LOF model
consists of two parts: the dataset collected from legitimate users and one new data of
the untrusted users. Since the attacker’s features are distinct from those of legitimate
users along at least one dimension, the attacker’s data appears as an outlier in the
whole dataset.

Given a feature vector \( z = [z_1, z_2, \ldots, z_K] \) of the untrusted user’s data, the local
reachability density (LRD) of a feature vector \( z \) is defined as:

\[
LRD(z) = \frac{1}{|N_k(z)|} \sum_{r \in N_k(z)} \min \{ k\text{-dis}(r), d(z, r) \},
\]

where \( N_k(z) \) are the \( k \) nearest neighbors (legitimate users’ data), \( r \) is a legitimate
user’s data that is also the \( k \) nearest neighbor of \( z \), \( k\text{-dis}(r) \) is the distance of the
object \( r \) to the \( k^{th} \) nearest neighbor, and \( d(z, r) \) is the Euclidean distance between
feature vectors \( z \) and \( r \) on the feature hyperplane. LOF model determines whether
the signal is from an attacker based on comparing the local densities of \( z \) and its
\( k \)-nearest neighbors using

\[
LOF_k(z) = \frac{\sum_{r \in N_k(z)} \frac{LRD(r)}{LRD(z)}}{|N_k(z)|}.
\]

Since the attacker’s features are distinct from those of legitimate users along at least one feature dimension, the attacker’s data point should be away from the cluster for legitimate users, which means its values of \( LOF_k(z) \) are larger than 1 on the feature hyperplane. Based on this observation, our system determines whether the signal is
generated by the attacker by setting a threshold $\tau$. If the value of $LOF_k(z)$ is larger than $\tau$, an attacker is claimed to be detected. Fig. 4.9 illustrates an example of LOF-based classification using two features $z_1$ and $z_2$. The darkness of the background represents the value of $LOF_k(z)$. The darker the background, the larger the $LOF_k(z)$. We can observe the $LOF_k(z)$ values of legitimate users are all less than 1.5, while that of the attacker is 2. By setting a threshold $\tau = 1.8$, the attacker can be accurately detected. In our system, the decision threshold $\tau$ is set to 3, and the number of neighbors is set to 5.

4.4.6.2 Decision combination for multiple rounds of detection

Since our solution does not require intense computational resources, it is possible to trigger our system multiple times during the video chat to tolerate single wrong classification. To combine the detection results of multiple attempts, we involve them in a majority voting game where each player has equal weight. Considering the final result is produced based on $D$ detection attempts, an untrusted user is regarded as a face reenactment attacker if its votes exceed $0.7 \times D$. The coefficient 0.7 is determined based on the detection accuracy of each single detection, which is reported in Section 4.5.3.
4.5 Evaluation

4.5.1 Implementation and dataset

Like the video chat scenario, our system consists of two components: a legitimate user (Alice in Fig. 4.4) who triggers the detection and an untrusted user (Bob in Fig. 4.4) with unknown legality. We implemented our testbed using a Dell 27-inch LED monitor with 85% brightness to display the video from the legitimate user. For the untrusted user who is also legitimate, we used a Google Nexus 6 smartphone to act as the camera for recording facial videos. For the untrusted user who is a face reenactment attacker, we first collect its facial videos using a Google Nexus 6 smartphone. The recorded facial videos are then fed to the driving model of ICface [64] for generating fake facial videos. The reason we use ICface is that it generated the most visually convincing results of any open-source facial reenactment method. In total, ten volunteers (four females and six males) with diverse skin colors (both dark skin and light skin) are involved in our experiments. To simulate the behavior of the legitimate user, we asked volunteers to record their daily video chat while changing the metering area by touching the smartphone screen. The collected facial videos are segmented into clips with equal length of 15 seconds. For the behavior of the untrusted user, we asked ten volunteers to act as both a legitimate user and a face reenactment attacker, respectively. For each role of each user, we replayed 40 video clips to them. For data analysis and processing, the data was then transmitted to a desktop computer with Intel(R) i7-8700 @ 3.2 GHz CPU and 32 GB of RAM.

4.5.2 Evaluation metrics

To evaluate the performance of our system, we use four metrics as follow: 1) true acceptance rate is used to describe how accurate our system can accept a legitimate user, and it is defined as the number of accepted attempts by the number of total...
Figure 4.11: Overall performance for a single detection.

4.5.3 Overall performance

4.5.3.1 System performance for legitimate user

A good face forgery system should provide high usability for legitimate users, which means the true acceptance rate should be as high as possible. To examine whether our system can be trained without collecting data from new users, we trained two classifiers using each volunteer’s own data and another volunteer’s data, respectively. For each volunteer, we repeated this experiment for 20 rounds to obtain the average true acceptance rate. Within each round, we randomly picked 20 instances for training and tested the system using the other 20 instances. Fig. 4.11 shows the true acceptance rates for a single detection attempt. We can observe that our system can provide an average true acceptance rate of at least 87.75% when the classifier is trained using each volunteer’s own data. Even if the classifier is trained using other’s
data, our system can still achieve an average true acceptance rate of at least 89.5%, which implies that our system can be quickly launched for new users without training in practice.

4.5.3.2 System performance against attacker

In this experiment, we randomly picked 20 instances from each volunteer as training data, and evaluated the true rejection rate of a single detection attempt against fake facial videos generated by ICFace. As illustrated in Fig. 4.11, our system can successfully reject the face reenactment attacker with average accuracy of at least 88.25%. For some volunteers (e.g. user 2), the mean true rejection rate can reach 97.25%, which means that our system can already provide high-security protection with only one detection attempt. Although our system can still make the wrong classification with a low possibility, our decision combination strategy can tolerate a single mistake and further improve system accuracy and robustness, which will be discussed in Section 4.5.6.

4.5.4 Influence of decision threshold

The value of the decision threshold influences the system performance a lot. If the decision threshold is too high, the legitimate users can all be passed by our system, but many attackers can also be missed. If we improve the security level by setting the decision threshold to a small value, many legitimate users will be rejected, which largely reduces usability. In this experiment, we evaluate the proper value of the decision threshold. We adjust the decision threshold from 1.5 to 4 while using 20 randomly-picked instances for training. Fig. 4.12 illustrates the mean false acceptance rate and false rejection rate for different values of the decision threshold. When the decision threshold is between 2.8 and 3, our system achieves balanced false acceptance rate and false rejection rate, which means the equal error rate of our system is about
5.5%. Therefore, we set the default value of the threshold to 3 in our system.

4.5.5 Influence of screen size

The performance of our system largely relies on the amount of light emitted from the screen. If the luminance of the screen light is low, the luminance change of the face-reflected light would be lower due to the light scattering. In this experiment, we evaluated the system performance by using screens with different sizes shown in Fig. 4.10, and the results are shown in Fig. 4.13. Better system performance is achieved by using a larger screen, which is in line with our expectation. Even with the smallest screen in our testbed, our system can still achieve an average true acceptance rate of about 85% for a single detection. Besides, we evaluate the system performance on a 6-inch smartphone screen. Experimental results show that our system can achieve similar performance only when the user’s face is very close (about 10 cm) to the screen. When the screen is moved too far away, the luminance of the screen light is not strong enough to generate significant luminance change on the user’s face.

4.5.6 Influence of number of detection attempts

Due to the influence of noise, our system may wrongly accept an attacker or reject a legitimate user with a low possibility. To tolerate the wrong detection for a single video clip, our system combines the results from multiple detection attempts through
a majority voting procedure. Fig. 4.14(a) shows the system performance under a different number of detection attempts when 20 instances are used for training. We can observe that both true acceptance rate and true rejection rate are significantly improved by combining multiple detection results whether the classifier is trained using the user’s own data or others’ data. Moreover, the variances of accuracy (both true acceptance and true rejection rate) are largely reduced, which means the system robustness is improved by considering multiple detection attempts.

4.5.7 Influence of number of training data

To quickly launch our system in practice, we want to reduce the training cost by as much as possible even if our system is trained using other’s data. Therefore, we performed an experiment using the data collected from one volunteer for evaluating how many training instances are needed for good system performance, and the results are shown in Fig. 4.14(b). When the classifier is trained with eight instances, our system can already provide an average true acceptance rate of 92.25% for normal users and an average true rejection rate of 91% for attackers. By involving up to 20 training instances, the average true acceptance rate and true rejection rate are slightly raised to 94.75% and 95.75%, respectively. Additionally, the standard deviations of both true acceptance rate and true rejection rate are largely reduced by up to 8.8%, which shows the system robustness is largely raised by having more knowledge about
4.5.8 Influence of sampling rate

The computation overhead of our system largely depends on the sampling frequency of each video. High sampling rates can provide us more information about the luminance change, but the image and signal processing overhead also multiply. To find the lowest viable sampling rate for our system, we collected data from one volunteer and vary the sampling rate among 5 Hz, 8 Hz, and 10 Hz. As illustrated in Fig. 4.15, our system can still provide mean detection accuracy of at least 95.25% for both legitimate users and attackers when the sampling rate is only 8 Hz. When the sampling rate drops to 5 Hz, the mean true acceptance rate slightly decreases to about 86%, while the mean true rejection rate rapidly drops to only 48%. Therefore, our system requires a sampling rate of at least 8 Hz to ensure security.

4.6 Chapter summary

In this chapter, we propose a defense system for real-time video chat against fake facial videos generated by face reenactment techniques. The key insight behind our system is that the luminance of the face-reflected light is proportional to that of the screen light. Therefore, we can detect face forgery by measuring the
correlation between the luminance signals of the screen light and the face-reflected light. Compared with existing works, our system has three major strengths. First, it does not require extra hardware or intense computational resources. Second, our system does not replace original video frames and can ensure a certain level of user experience. Moreover, our system does not require the training data of attackers and new users, which means it can be quickly launched on any videotelephony device.

This work in this chapter is published in the third paper of the publication list.

We summarize our contributions as follows:

- This is the first work where the luminance of face-reflected light is used to defend against face reenactment attacker.

- We propose robust solutions for extracting luminance signals from videos and finding significant luminance change from noisy luminance signals.

- We extract four strong features from the filtered signals to describe the luminance change behavior and trend. Moreover, we propose a local outlier factor-based classifier to detect fake faces in videos without collecting training data from either a new user or attackers.

- We develop a prototype and conduct comprehensive evaluations. Experimental results show that our system can provide an average true acceptance rate of at least 87.75% for legitimate users and reject face reenactment attacker with mean accuracy of at least 88.25% for each detection.
Nowadays, personal identification number (PIN) is one of the most popular methods for identity verification. However, recent researches show that attackers can easily recover victims’ PINs in spite of the large number of combinations PIN provides. Existing protection approaches require alteration of the original interaction between the user and PIN-based authentication systems, or still fail if the attacker can observe and mimic the victim’s input behavior. Considering these limitations, we propose a defense system called LightDefender to protect current PIN-based systems from PIN replay attacks using a single ambient light sensor. Specifically, we protect the PIN input by leveraging the biometrics in the received light intensity that is influenced by input behaviors and biological features. To our best knowledge, our work is the first one to protect PIN input using the light intensity. Different from existing approaches, LightDefender does not change the original interaction methods between the user and PIN-based authentication systems, and the extra hardware cost is low. In addition, by leveraging biological differences (e.g. finger length) among different users, LightDefender still claims high-security protection against strong attackers who can mimic the victim’s input behaviors.
5.1 Introduction

User authentication is an important procedure for a system to verify the identity of the user. Among all authentication approaches, personal identification number (PIN) is one of the most popular ones because of its combination of both usability and security. A PIN is a numeric or alpha-numeric password used in the process of authenticating a user accessing a system. A 4-digit PIN can result in 10,000 possible combinations. Therefore, PIN is widely used to withdraw cash from an ATM, unlock a mobile device, open a door, and so on. However, recent researches show that attackers can easily recover the victims’ PINs in spite of a large number of combinations PIN provides. The attacks can be grouped into three categories. First, to achieve good usability, users tend to pick context-related PINs (e.g. birthday), which largely decreases the randomness of PINs and makes it easier for the attacker to hit the PIN [65]. Second, the attacker can perform shoulder-surfing attacks or leverage a camera to record the victim’s input procedure [66, 67, 68, 69, 70]. Third, recent researches show that the attacker can use various sensor information and side channel (e.g. acceleration and acoustic signals) to help in recovering victims’ PINs [71, 72, 73, 74]. Therefore, it is essential to have a defense system that adds protection to PIN-based authentication systems, especially if the PIN has been leaked.

To defend PIN users against potential threats of PIN leakage, most existing approaches focus on preventing the attacker from acquiring the victim’s PIN, and they can be classified into two categories: challenge-response-based approaches [75, 76, 77] and indirect input-based approaches [78, 79, 80]. In challenge-response-based approaches, the user is asked to input the correct response that is calculated using the PIN based on a random challenge. However, by repeating the challenge procedure, the attacker can still gather useful information about the original PIN based on multiple challenge-response pairs [81, 82]. To address this issue, various
Figure 5.1: LightDefender system scenario.

solutions are proposed to prevent the attacker from observing the challenge-response pairs by using secure secondary channels [83, 84, 85, 86, 87, 88, 89, 90, 91, 92]. However, these approaches suffer from low usability and high learning cost. Similar to the challenge-response-based approaches, indirect input systems ask users to input on a secondary interface. However, indirect input-based approaches still alter the original interaction methods. To ensure the good usability of the defense system, other researchers propose to defend against PIN leakage by leveraging the input behavior (e.g. velocity and direction) during PIN input [93]. However, since they only consider simple time-domain features (e.g. velocity magnitude and directions), they fail to defend against the attackers who can mimic the victim’s input behavior through shoulder-surfing and recording attacks [94].

Considering the limitations of existing approaches, we propose a defense system that aims to protect the current PIN-based authentication systems from PIN replay attacks. A defense system should meet three key requirements. First, the defense system should have high usability, which means that it cannot significantly change the original interaction methods between the user and the PIN-based authentication systems. Second, the defense system should provide high-security protection to PIN input against strong attackers who can mimic the victim’s input behavior. Third, the extra hardware cost should be as low as possible. In this chapter, to meet the above requirements, we propose a new system called LightDefender to defend against
PIN replay attacks using a single ambient light sensor. In PIN replay attacks, an attacker already has the victim’s PIN via some way and aims to break PIN-based authentication systems by inputting the victim’s PIN. Specifically, we protect the PIN input by leveraging the biometrics in the received light intensity that is influenced by input behaviors and biological features (e.g. finger length). Different from existing approaches, LightDefender does not change the original interaction method between the user and PIN-based authentication systems, and the extra hardware cost is low. In addition, compared with input behavior-based approaches, LightDefender provides protection against strong attackers who can mimic the victim’s input behaviors.

As shown in Fig. 5.1, LightDefender consists of two major components: a low-cost light sensor and a light source. The ambient light sensor lies in the center of the PIN pad or keyboard and converts the received light intensity to the output voltage. The light source is over the light sensor and continuously emits visible light. When a user inputs a PIN, the palm and fingers block partial incident light, which generates different light intensities received at the light sensor. Therefore, even if an attacker can “replay” the victim’s PIN to authentication systems, the light intensity signal is different from that of the victim as long as the attacker does not follow the victim’s input behavior. Moreover, even if an attacker can record and mimic the victim’s input behavior by performing shoulder-surfing and recording attacks, the received light intensity of the attacker is still distinct from that of the victim because of biological differences (e.g. finger length and width) between the attacker and the victim. These biological differences also introduce variances to the received light intensity. In this chapter, we investigate the possibility of protecting PIN input using the biometrics in the raw output voltage of a single ambient light sensor, while eliminating the influences of noise (e.g. movement of nearby people). In particular, we develop a mechanism to detect the fine-grained starting and ending points of the PIN input only based on the raw output voltage signals. Then, we extract 34 features
from the raw output voltage signals in the time domain, the frequency domain, and
the time-frequency domain. These features are used to build a classification model
that is used to determine whether the input is from the normal user.

5.2 Literature review

**PIN leakage.** Although PIN is proposed as an authentication method with
high security, recent researches show that it can be reconstructed using various
techniques. In general, these attacks can be grouped into three categories: statistics-
based approaches [65], camera-based approaches [66, 67, 68, 69, 70], and side-channel
information-based approaches[71, 72, 73, 74]. For statistics-based approaches, a recent
study shows that the most common numbers follow some patterns and tend to be
based on some context (e.g. birth date) [65]. In camera-based approaches, the
attacker can reconstruct the PIN with high accuracy based on the video-based side-
channel information. Moreover, recent researches show that sensors in the victim’s
mobile and wearable devices can reveal its sensitive PIN [71, 72, 73]. Other works
are also proposed to infer the PIN by using various side-channel information (e.g.
acoustic signals) [74, 72, 70].

**Defence against PIN leakage.** Considering the threats of PIN leakage, various
systems are designed to either prevent the attacker from acquiring the victim’s PIN.
These systems can be further classified into two categories: Challenge-response-based
approaches [75, 76, 77, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92] and Indirect input-
based approaches [78, 79, 80]. Challenge-response-based approaches [75, 76, 77]
are all based on the insight that the attacker who does not know the mapping
function cannot recover the victim’s PIN based on the newly constructed password.
However, by repeating the challenge procedure, the attacker can still gather useful
information of the original PIN based on multiple challenge-response pairs [81, 82].
(a) A circuit diagram of LDR-based light sensor. (b) 2-D illustration of incident light when the finger moves vertically and horizontally.

Figure 5.2: Human vocal system and two propagation paths of the voice.

To solve this problem, various solutions are proposed by delivering the random challenge through various secure secondary channels that are invisible to the attacker [83, 84, 85, 86, 87, 88, 89, 90, 91, 92]. Although their approaches achieve high accuracy on defending against shoulder-surfing attackers, they usually come with low usability and introduce extra cost to users for learning the new system.

Similar to the challenge-response-based approaches, indirect input systems prevent the attacker from observing the PIN input procedure by leveraging a secondary input interface. However, indirect input-based approaches still alter the original interaction method of the PIN input, and the secondary interfaces usually introduce high hardware cost (e.g., Google glass). There are also many systems that try to authenticate the user by leveraging the biometrics in input behaviors or keystrokes [93, 95, 96, 97]. However, they only consider simple features mainly in the time domain such as velocity magnitude and device acceleration. It is still possible for an attacker to perfectly mimic the victim via shoulder-surfing attacks [94].
5.3 Preliminary

5.3.1 Ambient light sensor

A light sensor generates an output signal indicating the intensity of light by measuring the radiant energy that exists in a very narrow range of frequencies basically called “light”, and which ranges in frequency from “Infra-red” to “Visible” up to “Ultraviolet” light spectrum. Among all types of light sensors, the photoconductive cell using Light Dependent Resistor (LDR) is the most common. The LDR is made from a piece of exposed semiconductor material that changes its electrical resistance based on the received light intensity. Fig. 5.2(a) shows a circuit diagram of an LDR-based light sensor. We can acquire the light intensity level by measuring the voltage $V_{out}$ at their junction. The output voltage $V_{out}$ is determined based on:

$$V_{out} = V_{in} \times \frac{R_{LDR}}{R_{LDR} + R_1},$$

where $V_{in}$ and $V_{out}$ are input and output voltage, respectively. $R_1$ is a series resistor, and $R_{LDR}$ is the light dependent resistor. When the light intensity is low, the resistance of the light dependent resistor reaches a high value, which produces a high output voltage. In contrast, the output voltage is low when the received light intensity is high. Since the value of $R_{LDR}$ will never be zero or infinity, the LDR sensor is expected to measure the light intensity from 0 to infinity.

In our system, we embedded an LDR-based light sensor in the middle of a keyboard. The underlying principle of light-based gesture recognition systems [98, 99, 100] and our light-based defense system are fundamentally similar: the hand can reflect or block the light, which further influences the received light intensity at the sensor. The light is from a fixed light source that is over the PIN pad (e.g.
attached on the shield) and emits lights with consistent intensity. Fig. 5.2(b) shows how the vertical and horizontal movements of the finger influence the incident light on a 2-D plane. The yellow region illustrates the space in which the light can reach the ambient light sensor. The finger and hand are modeled as a line. We can observe that, if the finger moves away from the light sensor vertically, more light reaches the light sensor, which produces higher light intensity. Similarly, if the finger moves away from the light sensor horizontally, the received light intensity rises since more direct light reaches the light sensor. Moreover, due to the biological differences (e.g. finger length, finger width, and palm size) among different users, the received light intensities are also different even if the fingers and hands of two users are at exactly the same location.

5.3.2 Attack model

In our attack models, the attacker aims to break PIN-based authentication systems (e.g. ATM machine) by “replaying” the victim’s PIN to authentication systems. The capability of the attacker is limited in the sense of:

A simple PIN replay attack. In this attack model, the attacker can acquire the victim’s PIN by using non-vision techniques (e.g. motion sensors). Therefore, the attacker only has the victim’s PIN without knowing how the victim inputs the PIN. To break the PIN-based authentication system, the attacker inputs the PIN with a random input behavior (e.g. different fingers).

A strong PIN replay attack. In this type of attack model, we assume that the attacker can use vision-based techniques (e.g. a hidden camera) to infer the victim’s input behavior, which means that the attacker knows not only PIN but also the victim’s input behavior. To break the PIN-based authentication system, the attacker inputs the PIN while imitating the victim’s input behavior.
5.3.3 Feasibility study

To validate our observations, we build a sensing platform by embedding an LDR-based ambient light sensor in the middle of a keyboard. A light-emitting diode (LED) bar is installed over the keyboard to act as the major light source so the user’s hand and fingers can block the light as long as they are over the keyboard. The feasibility experiments are done in an office room where multiple light sources exist. The details of the platform setup are shown in Section 5.7. We first asked a volunteer to input a six-digit PIN ("146928") twice, and the measured output voltage signals are shown in Fig. 5.3(a). We can observe that the user’s input behavior introduces much greater influences to the raw output voltage measurements than other factors (e.g. the activities of other people in the same room). Moreover, the output voltage patterns of the same user are consistent overall. Although the same user cannot perfectly reproduce the same pattern (small variations in the red box) still exist, we can still extract useful knowledge (e.g. overall shape and frequency) from the raw output voltage to match the patterns from the same user.

Moreover, we first asked the simple replay attacker to input the victim’s PIN in its own way on the same testbed, and the raw output voltage is shown in Fig. 5.3(b). We can see that the output voltage pattern is distinctive from that of the victim because
their input behaviors (e.g. finger used and habitual hand) are different. Especially in
the second phase of the input behavior (green box), the peak-to-peak distance of the
victim’s data is much higher than that of the attacker’s data. We also collected data
from the strong replay attacker, as shown in Fig. 5.3(c). We used a camera to record
the whole process of the victim’s PIN input. The strong replay attacker is required
to watch the video until it is confident to imitate the victim’s input behavior. We
can see that the strong replay attacker is able to produce a similar voltage pattern
to the victim in terms of the overall shape, but the amplitude is still significantly
distinct from that of the victim. In the first stage of the input behavior (purple box),
the average amplitude of the victim’s data is about 0.5 V, while that of the strong
replay attacker is about 0.4 V. The reason behind this is the biological differences
between the victim and the attacker. For example, even if two different fingers are
at the same location, different finger length and width determine the amount of light
that is blocked, which produces different light intensity received at the light sensor.
Therefore, even if the strong replay attacker can perfectly imitate the victim’s input
behavior, it cannot produce the same voltage pattern as long as its hand and fingers
are biologically different from those of the victim.

5.3.4 Challenges

**Fine-grained input detection.** In order to defend against the PIN replay attack
using an ambient light sensor, we first need to extract the sensor signal that is
influenced by the PIN input. In general, the PIN input procedure consists of three
phases: moving hands over the keyboard, inputting the PIN, and moving hands back.
A naive solution is to acquire the key pressing time from the PIN-based authentication
system, but this approach only reserves the sensor signal in the second phase while
losing all information in the other two phases. To extract the sensor signals that
contain the information in all three phases, we proposed an energy-based input
detection approach based on the insight that PIN input has greater influences on the sensor values than other factors.

**Accurate classification model using proper features.** After getting the raw output voltage data of the whole PIN input procedure, we need to extract features that are consistent for the same user and distinctive between the user and the attacker. Moreover, the classification model should be robust to the collinearity of extracted features because features are heterogeneous across different domains. To address this problem, we extract features from the time domain, frequency domain, and time-frequency domain of the raw output voltage signal. To leverage the collinearity of features from three domains, we use a multiple additive regression tree for classification.

### 5.4 System Overview

We build a system that mainly contains two major phases: the enrollment phase and the authentication phase. The processes of both phases follow the pipeline shown in Fig. 5.4.

**Enrollment phase.** In the enrollment phase, the user is asked to repeat inputting its PIN several times. Since the user is not able to give the accurate starting and ending time of the PIN input, LightDefender processes the raw signal to extract the output voltage signal that is influenced by the PIN input. Considering the frequency of PIN input is at least 1 Hz, LightDefender first removes the influence of background
noise by filtering the output voltage signals through a high-pass filter and detects the coarse-grained location of the PIN input by studying the short-time energy of filtered signals. Then, LightDefender detects the accurate starting and ending time of the PIN input by analyzing the short-time energy around the coarse-grained location based on a threshold. The extracted output voltage signals are used to extract features that can represent the identity of the user. These features are trained together with attackers’ data (collected in advance) in the database to build a strong classifier.

**Authentication phase.** After collecting enough training data from the user, the system is ready to be used for authentication. The system can be used by the normal user or an attacker. For each authentication attempt, we first detect and extract the PIN input in the same way as in the enrollment phase. After that, we extract the same 34 features of the new input and send it to the multiple additive regression tree-based model. An attacker is detected and rejected if the classification model recognizes the new input as from an attack.
5.5 PIN Input Detection

5.5.1 Coarse-grained PIN input detection

To defend against PIN replay attackers using the ambient light sensor, we first need to accurately detect the starting and ending time of the PIN input behavior. In general, the PIN input behavior can be segmented into three stages: 1) Moving the hand over the PIN pad; 2) Inputting PIN; 3) Moving the hand back. A simple solution is to acquire the pressing time of the first and last keys from the authentication systems. However, this solution will lose the sensor information in the first and the third stages. Moreover, the output voltage of the ambient light sensor is also constantly influenced by other factors (e.g. human activities nearby), which makes it hard to detect the starting and ending points using a threshold.

To address this issue, instead of directly detecting the fine-grained starting and ending points, we first find the coarse-grained location of PIN input gesture in the noisy output voltage signals using the finding algorithm in [101]. The accurate starting and ending points are then detected around the coarse-grained locations. Fig. 5.5(a) shows the raw output voltage when a user walks to a PIN authentication system and inputs a PIN. It is clear that the PIN input behavior will introduce much greater influence than other human activities. Moreover, as shown in Fig. 5.5(b), most background noise has a frequency of less than 1Hz, while the PIN input still has information with a frequency of larger than 1 Hz. Based on these two observations, we find the coarse-grained location of PIN input by analyzing the short-time energy of noisy output voltage signals. In order to remove the pulses caused by background noise, we apply a 3-order high-pass filter on the raw signals with a cut-off frequency of 1 Hz, and the filtered signal is shown in Fig. 5.5(c). We can see that pulses introduced by the PIN input are much more significant in the filtered output voltage signal. To further remove the pulses caused by background noise, we apply a threshold filter on
the output of the high-pass filter. The threshold is set as the mean of the high-pass filter’s output, excluding the highest 40% and the lowest 40% of the measurements. All measurements whose values are lower than the threshold will be 0 after passing through the threshold filter. To find the coarse-grained location of the PIN input, we apply a moving window to the filtered output voltage signals and compute the short-time energy within each window. The window size is set to 1.8 seconds in our system for two reasons. First, it is the minimal time to input a 6-digit PIN. Second, by using the minimal time as the window size, we can ensure that the signal within the window is only influenced by PIN input. Since the pulses introduced by the PIN input are much more significant in the filtered output voltage signal, the starting point of the moving window must be within the PIN input procedure when the short-time energy within the window reaches the highest value. Therefore, the coarse-grained location of the PIN input can be detected by solving:

$$\arg \max_s ([g_{s}, g_{s+1}, \ldots, g_{s+w}]) ([g_{s}, g_{s+1}, \ldots, g_{s+w}])^T$$

where $s$ is the coarse-grained location of the PIN input, $G = [g_1, g_2, \ldots, g_n]$ is the filtered output voltage signal, $n$ is the length of the filtered signal $G$, $w$ is the size of the moving window, and $([g_{s}, g_{s+1}, \ldots, g_{s+w}]) ([g_{s}, g_{s+1}, \ldots, g_{s+w}])^T$ computes the short-time energy of the window starting from the $s^{th}$ sample to the $(s + w)^{th}$ sample. Fig. 5.6(a) shows the short-time energy of windows starting from different samples. We can see that the short-time energy reaches its highest value at 19.4 seconds, which is exactly during the PIN input.

5.5.2 Fine-grained starting and ending points detection

Since the detected coarse-grained location lies within the procedure of PIN input, the accurate starting and ending points must show near the coarse-grained location.
Moreover, we find that the output voltage values are pretty stable before and after the PIN input because the user will not move before and right after the PIN input. Therefore, the values of these two stable stages should be close to zero after filtering the raw output voltage signal with high-pass and threshold filters. This observation enables us to detect the accurate starting and ending points by checking short-time energy changes before and after the coarse-grained location. To detect accurate starting and ending points of PIN input, we first apply a moving window on the filtered signal in coarse-grained location detection. To achieve better granularity, we set the window size to 0.3 seconds, and the result is shown in Fig. 5.6(b). We can see that the short-time energy is very low (close to zero) until the PIN input starts. Since the most noise is removed in the filtered signal, we can accurately detect the starting and ending time by finding the first and the last points whose energy exceeds a threshold around the coarse-grained location. In our testbed, the threshold is set to 0.00001 V².s, and both starting and ending points should be within 5 seconds from the detected coarse-grained location.
Figure 5.7: The Wigner-Ville distribution of the victim and the strong attacker.

5.6 User Authentication

5.6.1 Feature extraction

To train a strong classifier that can detect the replay attacker, we need to extract useful features from output voltage signals that are influenced by the PIN input. Here, useful features are those that are consistent for the same user but distinctive between the normal user and the attacker. In our system, we select 34 different features from the time domain, the frequency domain, and the time-frequency domain.

Features in the time domain. We extract six features from the time domain, including the maximum, the average amplitude, peak-to-peak distance, variance of the signal, root-mean-square (RMS) level, and the average dynamic time wrapping
(DTW) distances between the new data and the templates that are selected from the normal user's pre-collected data. Specifically, the maximum, the average amplitude, and peak to peak distance describe the overall amplitude of the raw output voltage, which is mainly influenced by biological features such as finger length and width. The RMS level and variance are used to describe the trend of the signal. The DTW distance is used to measure whether the new data has a similar shape as the user's template that is collected during the enrollment phase. Since we only consider the overall shape of the detected output voltage signal, we normalize each output voltage signal individually over the range of the ADC to eliminate the influence of voltage value. Moreover, we smooth each raw output voltage signal using a moving average filter with a window length of 20 samples.

**Feature in the frequency domain.** To capture the features in the frequency domain, we perform a fast Fourier transform (FFT) on the extracted output voltage signals. Six features are extracted from the FFT result, including skewness, kurtosis, mean value, median value, variance, and peak-to-peak distance. These features describe the rhythm of how the user presses the key and blocks the incident light.

**Feature in the time-frequency domain.** Besides extracting features from the time and frequency domains individually, we also study how the PIN inputs influence the output voltage signal in each time and frequency frame. We first apply maximal overlap discrete wavelet transform (MODWT) using the Haar wavelet down to the fourth level on the raw signal and perform multiresolution analysis on the MODWT matrix. The reason we choose MODWT rather than classic discrete wavelet transform is that MODWT can achieve translation-invariance by removing the downsamplers. We extract the mean value, peak-to-peak distances, RMS, and variance from the results of the multiresolution analysis as features. Also, we calculate the Wigner-Ville distribution of the raw signal. Compared to a short-time Fourier transform, the Wigner-Ville distribution function can furnish higher clarity. For a discrete signal
\[ G = [g_1, g_2, \ldots, g_n] \text{ with } n \text{ samples, the Wigner-Ville distribution is defined as:} \]

\[
WVD_G(t, f) = \sum_{k=-n}^{n} G(t + \frac{k}{2})G^*(t - \frac{k}{2})e^{-j2\pi fk}, \quad (5.6-3)
\]

where \( t \) is the time vector, \( f \) is the frequency vector, and \( G^*(t - k/2) \) is the complex conjugate of \( G(t - k/2) \). Fig. 5.7(a) shows the Wigner-Ville distribution of the output voltage signal influenced by the PIN input. We can observe that the PIN input influences the output voltage signal mainly in the low-frequency bands. Therefore, we further check the Wigner-Ville distribution of the voltage signals influenced by PIN inputs of the normal user and the strong replay attacker respectively, and the results are shown in Figs. 5.7(b) and 5.7(c). It is clear that the energy distribution is distinctive in the low-frequency bands in two aspects. First, the entries with the lowest amplitude appear at different locations in two Wigner-Ville distributions. As shown in Fig. 5.7(b), in the user’s distribution, the entry with the lowest amplitude is at the later stage of the PIN input. While in the strong attacker’s distribution, the entry with the lowest amplitude appears at the middle stage of the PIN input. Second, the energy distribution in each frequency band is very distinctive between the user and strong replay attacker, which means that we can detect the attacker by checking the standard deviation of the energy distribution in each frequency frame. Therefore, we extract the location of the minimal amplitude and its amplitude value as three features. Moreover, we calculate the standard deviation of the energy distribution for each frequency frame under 2 Hz and include them into the feature vector. To deal with different frequency resolutions caused by different lengths of signals, we resize each Wigner-Ville distribution to the same size so that the first 15 frequency frames exactly cover the frequency range from 0 Hz to 2 Hz.
5.6.2 Classification based on multiple additive regression tree

To determine whether the extracted features are from the normal user or any type of PIN replay attackers, we train a binary classifier based on Multiple Additive Regression Tree (MART). Compared with other machine learning models, the gradient boosting-based approach has three major advantages. First, MART is robust to various types of features with different scales and units, which exactly exists in our feature vectors. For example, the value of the maximal amplitude is in the range from 0 to 1, while the values of DWT features can be less than 0.001. Second, features extracted from different domains may not be totally independent of each other. By using MART, the classifier can effectively deal with the colinearity of features across various domains.

The basic idea of MART is to build a strong classifier using a set of weak classifiers. Different from other gradient boosting approaches, MART specializes the gradient boosting approach to the case where each weak classifier is a regression tree. Here we use the formulation of MART in [102]. After $M$ rounds, the estimation $F(x)$ of the strong classifier is an additive expansion of the form

$$F(x) = \sum_{m=0}^{M} b_m h(x; a), \quad (5.6-4)$$

where $h(x; a)$ is a weak classifier with parameters $a = \{a_1, a_2, \ldots, a_K\}$ and feature vector $x = \{x_1, x_2, \ldots, x_N\}$. In each iteration, the coefficients $b_m$ and the parameters $a_m$ are jointly fit to the training data in a forward “stage-wise” manner. Starting with an initial guess $F_0(x)$, the coefficients $b_m$ and the parameters $a_m$ in the $m^{th}$ iteration can be found by solving the following problem:

$$(b_m, a_m) = \arg \min_{b, a} \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + bh(x_i; a)), \quad (5.6-5)$$
where $y_i$ is the diagnosis variable, and $L(y, F)$ is the loss function that is used to define lack-of-fit. Therefore, the estimation of the strong classifier after the $m^{th}$ iteration is expressed as

$$F_m(x) = F_{m-1}(x) + b_m h(x; a).$$ (5.6-6)

In our system, we implement the MART-based classifier using the library of scikit-learn [103]. Specifically, we choose the deviance function as the loss function, and the learning rate is set to 0.1. Since the MART-based classifier is fairly robust to over-fitting, we set the number of iterations to 5000 to achieve better performance. For each regression tree, the maximal depth is set to 4, and the number of features to consider when looking for the best split is set to 4.

5.7 Evaluation

5.7.1 Experimental prototype

Since commercial PIN pads or keyboards are not equipped with an ambient light sensor, we built a prototype to mimic the layout and structure of PIN pads that are widely used on ATM machines. As shown in Fig. 5.8, our prototype consists of five components: a prototype PIN pad (made by cardboard), an LDR-based ambient light sensor (about $1), an analog-to-digital converter (ADS1115 16 bit and 4-channel
analog-to-digital converter), a data sink and processing center (Raspberry Pi 3 b+), and a light source (WORKRITE ERGONOMIC VERANO LED array). Since the Raspberry Pi only accepts digital signal from GPIO input, we used a 16-bit converter to convert the analog output to digital signals. On the Raspberry Pi board, we used a Python script and public library to read the sensor data with a frequency of 100 Hz. The LDR-based light sensor is attached in the middle of the PIN pad that is placed under the light source. We implemented our prototype in a shared office room where different human activities exist.

5.7.2 Data collection

Our experiments included 10 participants (4 males and 6 females) aged from 22 to 29. All participants are university students who have no knowledge of our system details. We asked each participant who acts as the normal user to randomly choose a 6-digit PIN and input it on our prototype in a comfortable way 43 times. Among them, three instances are used as the template for calculating DTW distances and 20 randomly picked instances are used as training data to build the MART-based classifier. Additionally, for each normal user, we asked three other participants to act as an attacker. During a simple PIN replay attack, we only gave each of the three attackers the PIN of the victim. Each attacker input the victim’s PIN on our prototype in its preferred way 10 times, so we have 30 instances for the simple PIN replay attack. During a strong PIN replay attack, we showed each of the three attackers not only the victim’s PIN but also the videos of the victim’s input behavior. When each attacker was confident enough to mimic the victim’s behavior, the strong PIN replay attack was launched 10 times. For each attacker in both simple and strong PIN replay attacks, five randomly picked instances are used as training data and the remaining five instances are used for testing. Therefore, the training dataset of each user has 20 instances from the normal user, 15 instances from each simple PIN replay
To evaluate the performance of our system, we used three metrics, including true acceptance rate (TAR), true rejection rate (TRR), and authentication time. The true acceptance rate is defined as the rate at which a normal user is successfully accepted by the system. Similarly, the true rejection rate is defined as the rate at which an attacker is successfully rejected. The authentication time is defined as the number of PIN input attempts needed to pass our system.

5.7.3 System performance for normal users

We first evaluated the system performance for normal users by repeating the experiment 20 times with randomly picked training data and testing data. Fig. 5.9 shows the average true acceptance rate for 10 participants. We can observe that our system successfully accepts a normal user with an average true acceptance rate of 95%. For user 5, 6, and 8, the average true acceptance rate can reach near 100%. We further study why user 10 has a lower true acceptance rate than other users. We found that the user 10 used the most complex input behavior with 4 fingers in our experiments, which makes her input behaviors less consistent than those of other users and leads to lower true acceptance rate. However, even in the worst case (user 10), our system can still accept the normal user with an average accuracy of at least 89%.
5.7.4 System performance against two types of PIN replay attack

With attackers’ training data. Similarly, we used the same classifier in Section 5.7.3 and repeated the experiment. The experimental results are illustrated in Fig. 5.9. It is clear that our system can provide high true rejection rates of about 98% and 96% for both types of PIN replay attacks, respectively. Especially for users 2 and 6, our system can reject all attackers with nearly no errors. We also found that the system performance can decrease to 91% against attackers when the input behavior of a normal user is simple and easy to mimic. For example, user 7 only used his index fingers to input the PIN while other fingers are holding up, which makes it easier for strong attackers to produce similar patterns of received light intensity. Moreover, our system can achieve similar performance for 4-digit PINs with a mean true acceptance rate of about 98.7%, and both types of attackers can be rejected with an accuracy of nearly 100%.

Without attackers’ training data. We also evaluated the system performance against attackers whose data is not in our training dataset, which is more common in practice. Fig. 5.10 illustrates the true rejection rates against two types of attackers. We can see that our system can still ensure high-security protection for users against simple PIN replay attackers with a mean true rejection rate of 96.8%. Moreover, even if strong attackers can imitate the victim’s input behaviors, our system can still reject them with mean accuracy of 93.6%
5.7.5 Authentication time

The system performance above is for a single PIN input attempt. In practice, to achieve good usability, PIN-based authentication systems usually allow the user to input its PIN for up to three or five times. Therefore, we studied the system performance within the maximum number of input attempts. Fig. 5.11(a) shows the authentication time distribution of the normal user and two types of PIN replay attackers. If the attacker cannot break our system within ten attempts, its authentication time is set to 10 times to avoid an infinite number. We can see that all normal users can be correctly accepted by our system within two input attempts, while any type of PIN replay attacker is falsely accepted with a possibility of no more than 2%. Even if the attacker can launch the PIN replay attack at most five times, our system can still provide a high true rejection rate of at least 94.5%.

5.7.6 Influence of the size of training dataset

In practice, we need to control the size of the training dataset to reduce the cost in the training phase. Therefore, we further studied what is the minimal size of the training dataset needed from the normal user. In our system, we assume that we can collect the attacker’s training data in advance for any possible PIN. In this
experiment, we randomly selected a normal user and adjusted the training dataset size from 1 to 20 while the training dataset size of two types of attackers was fixed to 30 instances. To eliminate the influence of extremely imbalanced training data, we made the normal user’s training dataset size constant at 20 by randomly duplicating the training instances. Fig. 5.11(b) shows the average true acceptance rate and true rejection rate against two types of PIN replay attacks. We can see that true acceptance rate rises with more training instances from the normal user, while the system performance against two types of PIN replay attacks is relatively stable (over 99.7% and 97.3%, respectively) no matter the amount of training instances from the normal user. Specifically, with 9 normal user’s training instances, our system can already provide an average true acceptance rate of 91.75%.

5.7.7 Influence of light conditions

To further evaluate the system performance under lower light intensities, we used an ANNT LED Desk Lamp as the new light source that contains an LED array and can emit lights of five levels from 315 lux to 610 lux. Fig. 5.12(a) shows the true acceptance rates and true rejection rates under six different light intensities. We can see that the system performance is not influenced by the light intensity of the light source within the range (315 lux to 825 lux) we considered. When the received illuminance is only 315 lux, our system can still correctly accept the normal user and reject the strong PIN replay attacker with an accuracy of at least 98%. Based on our experiment, the average received illuminance is about 350 lux under the fluorescent lamp. Therefore, the light intensity required in our system is comfortable and acceptable for users.
**Figure 5.12:** Influences of different light intensities and hand conditions.

### 5.7.8 Influence of gloves and wet hands

In our default settings, we assume users always interact with our system using dry hands. However, in practice, users may use our system in various conditions, e.g. wet hands in the summer. To evaluate the robustness of our system against various hand conditions, we asked a participant to input his PIN when his hand is wet and in purple nitrile gloves, respectively, and used the classifier that is trained using dry hands to make prediction. Fig. 5.12(b) shows the evaluation results. We find that our system can still correctly accept the user who used the wet hands with an average true acceptance rate of 93.75%. However, the true acceptance rate drops to about 10% if the user inputs its PIN while wearing gloves. By checking raw output voltage signals, we found that although the overall shapes of output voltage signals are still consistent, the gloves result in higher received light intensity than dry hands. In other words, the gloves change the biological features of the user, which makes the original classifier wrongly detect the user as a strong attacker.

### 5.7.9 Influence of sampling rate

As we discussed in Section 5.5.1, the influences of PIN input on output voltage signals are mainly in the frequency bands from 0 Hz to 2.5 Hz, which means a sampling
rate of 5 Hz is enough to capture the information of PIN input in theory. Although we use a high sampling rate of 100 Hz to capture as much information as possible, it is always good to reduce the sampling rate for saving energy. In this experiment, we evaluated the system performance under different sampling rates, and the results are shown in Fig. 5.13. It is clear that the system performance, especially the true acceptance rate, is improved with the greater sampling rate. When the sampling rate is 5 Hz, the obtained information is enough for our system to provide a high true acceptance rate of 90.5%. By including the high-frequency features, we can achieve an average true acceptance rate of 98% with a sampling rate of 100 Hz.

5.8 Chapter summary

In this chapter, we propose a new system called LightDefender to defend against two types of PIN replay attacks by leveraging the biometrics in the received light intensity that is influenced by input procedure. The key insight is that different input behaviors and biological differences result in different output voltage signals. These differences can be reused as biometrics to authenticate users right after the input procedure. Different from existing approaches, LightDefender does not change the original interaction methods between the user and PIN-based authentication systems, and the extra hardware cost is low. This work in this chapter is published in the fourth paper of the publication list.

Our contributions are as follows:

• Our work serves as a feasibility assessment to show that the light intensity...
influenced by the PIN input contains rich biometric information and can be used to verify the identity of the user. To our best knowledge, our system is the first to use ambient light to protect the PIN input.

- We propose a mechanism to accurately detect the starting and ending point of the PIN input by analyzing the raw output voltage signals. In total, 34 features are extracted and used to build a multiple additive regression tree-based classification model for the final decision.

- We develop a prototype and conduct comprehensive evaluations. Experiments with 10 volunteers show that LightDefender can achieve an average true acceptance rate of 95% for normal users. Moreover, LightDefender can correctly reject two types of PIN replay attackers with an average true rejection rate of at least 93.6% even if no data of new attackers is available.
CHAPTER 6

ATTACK SYSTEM: BREAK LOCATION

PRIVACY IN AR APPLICATIONS

AR applications that overlay a user’s perception of the real world with digitally generated information are on the cusp of commercial viability. AR has appeared in several commercial platforms like Microsoft HoloLens and smartphones. They extend the user experience beyond two dimensions and supplement a user’s normal 3D world. A typical location-based multi-player AR application works through a three-step process, wherein the system collects sensory data from the real world, identifies objects based on their context, and finally, renders information on top of the user’s senses. However, because these AR applications frequently exchange data with users, they have exposed new individual and public safety issues. In this chapter, we develop ARSpy, a user location tracking system solely based on network traffic information of the user, and we test it on location-based multi-player AR applications. We demonstrate the effectiveness and efficiency of the proposed scheme via real-world experiments on 12 volunteers and show that we could obtain the geolocation of any target with high accuracy. We also propose three defense methods to mitigate these side channel attacks. Our results reveal a potential security threat in current location-based multi-player AR applications and serve as a critical security reminder to a vast number of AR users.
6.1 Introduction

AR applications connect the physical world and the cyber world by overlaying digitally generated information on a user’s perception of the real world. Common AR applications use a marker, which is sufficient for AR projects where users can remain stationary, to trigger AR content. Location-based AR applications, in contrast, heavily rely on users’ physical locations. Typically, they use GPS (BLE beacons for the indoor environment) and simultaneous localization and mapping (SLAM) techniques to determine a user’s location and to detect a device’s orientation. Utilizing location information to enhance an AR application helps to create a more immersive experience by relying on physical proximity to automatically trigger AR content. As the first significant success in location-based AR, Niantic Lab’s Pokemon Go [104], a smartphone game combing location-based real-time tracking and AR, attracted more than 45 million daily users within just a few days of its launch; it has been downloaded 800 million times since then. However, the potential of location-based AR lies far beyond smartphone games, and it is being applied more consequentially in both consumer and business-to-business settings. For example, Gatwick Airport has installed 2,000 indoor navigation beacons, which will enable AR path-finding at the airport [105]. Moreover, many third-party AR services such as Wikitude and Motive.io, provide a full-featured software development kit (SDK) that allows developers to build location-based AR applications without concern for technical details, such as motion tracking, proximity calculation, or scale estimation. In fact, with increasing shift to hands-free devices, such as head-mounted displays or smart glasses, location-based AR is becoming a new information-delivery paradigm.

While the technology underlying AR applications is booming, little thought has been given to how these systems should protect the privacy of users. The AR devices continuously receive input from the environment through video, audio, and
Network throughput
Fake AR contents
Victim’s trajectory
Victim
Victim’s AR device
AR provider

Figure 6.1: An example show how the attacker infers the victim’s trajectory using network traffic.

other sensors, and the continuous network connectivity will expose new security and privacy issues, especially in scenarios where AR users can also upload AR contents to the server (e.g. AR-based message board). Existing AR systems protect users’ geolocations by encrypting the two-way transmission between users’ devices and server using HyperText Transfer Protocol (HTTP) and HTTP Secure (HTTPS) protocols. Even if the attacker can capture the network packets in the middle of transmission, the geolocation of the user is regarded as safe if the attacker cannot decrypt the network packets. However, it is known that the attributes of encrypted traffic, often referred to as side-channel information, can leak some sensitive information about the communications. Such side-channel information leaks have been studied by [106] (secure shell), [107] (voice-over-IP), and [108] (web application).

Several existing studies conducted by various research groups have shown anonymity issues in encrypted web traffic. It has been shown that even when a user visits a web page through HTTPS channel, that page can still be identified due to the distinct size of a page and corresponding resource objects (e.g., images) [108]. Despite the importance of this side-channel threat in an encrypted channel, there is currently no study in the AR application domain for understanding its gravity and mitigation solutions.
In this work, we explore the security threat model of AR devices and demonstrate a new side-channel threat caused by location-based multi-player AR applications’ unique combination of a high volume of real-time data, outsourced geolocation processing, and open privilege of uploading AR contents. We show that an adversary can covertly learn an AR device’s location and track the user in real-time by simply relying on monitoring the device’s network throughput. Different from getting GPS information from the victim’s device, the attacker can acquire network traffic information without using any location-related permissions, which means our attack methods are hard to be noticed by the victim in system permission level. Our attack model is proposed based on the following observations: 1) Location-based multi-player AR applications interact with a cloud database and cache AR contents when the victim is within a certain distance from them. 2) Many Location-based multi-player AR applications allow any user to upload or delete their AR contents to the database, such as WallaMe [109] and World Brush [110]. Therefore, as shown in Fig. 6.1, an attacker can also use the AR applications to upload fake AR contents of a specific size to the database in advance. Then, the attacker can observe a unique network traffic pattern on the victim’s AR device when the victim is close to that location. By properly determining the size and location of each AR content, the attacker can locate users and reconstruct the victim’s trajectory with high accuracy. Based on two observations, we propose a fake AR contents generation and deployment strategy. A network throughput processing method is also provided to extract a victim’s location information from the raw network throughput. Extensive experiments on our self-built AR application based on Android platform, simulation testbed, and a real location-based AR application show that our attack methods can reveal a victim’s location with high accuracy. Three mitigation solutions are also proposed to defend against this side-channel attack.
6.2 Literature review

Mobile augmented reality. The basic idea of augmented reality was proposed in the 1960s [111, 112]. Since the 1990s, researchers have become increasingly interested in this area, and many AR devices and frameworks have been proposed to overcome challenges to tracking and registration in the hopes of properly aligning virtual and real objects, user interfaces and human factors, and auxiliary sensing devices. The increasing capabilities of mobile devices, affordable high-speed Internet access, and breakthroughs in computer vision and cloud computing have only recently made AR a reality. Many mobile augmented reality (MAR) applications have been designed and implemented towards the following demands: 1). Tourism and navigation[113, 114, 115, 116]; 2). Advertisement [117, 118]; and 3). Entertainment [119]. In [113, 114], researchers propose a MAR prototype for campus exploration. The application can display information about surroundings while users are walking.

Augmented reality security. Lately, several researchers have focused on the security, privacy and safety concerns associated with AR systems [120, 121, 122].
However, most of the existing publications are focused either on input privacy [123, 124, 125] or output safety [122, 126, 127]. Only a few publications [128, 129, 130] have addressed output privacy of AR systems. Different from existing works, we point out a novel side channel that allows attacker to track an AR user even if the network traffic is encrypted.

**Fingerprinting and traffic analysis.** There is a large body of research on the side-channel attack on encrypted network traffic for traditional websites [131, 132, 133]. In [133], the authors evaluate a state-of-the-art method for detecting a website and conclude that webpage detection is infeasible. X. Cai et al. [132] proposed an attack method that can guess which of 100 web pages a victim was visiting with an accuracy of at least 50%. A more recent work [131] shows that it is possible to identify encrypted video streams in high precision. Besides website information, traffic analysis can also be used to infer application-specific sensitive information, such as health conditions [134], or other contextual information [135]. A recent work [136] is proposed to detect AR users’ locations by monitoring the network throughput. However, their solution only considers an small area (three locations) and involves much training cost. Prior works also cover mitigations [137, 138] and counter-mitigations [139].

**Location leakage through sensory data.** In the past few years, researchers did a lot of works on inferring locations using various types of sensory data and side channel information [140, 141, 142, 143]. For example, Liang et al. proposed a system to infer the locations of a user using motion sensors [140]. However, their system requires pre-collecting enough training data from the same user for the same path. Therefore, their system can fail to work when the user changes the movement behavior. Besides using sensory data from a single source, researchers also seek to predict the next location of a user using multiple sensors and context information. For instance, Do et al. predict the next location of the user using current context
consisting of current location, time, application usage, etc. [141]. However, such a model can only work when the behavior of the user is relatively stable. To reduce the impact of dynamic behaviors of a user, Tiwari et al. design an attack model that can infer location-related information of a user using the network traffic when the user is using Google Maps [142]. However, they can only provide good performance on path detection over time but fail to detect the real-time location of a user. Compared with existing work, our system does not need to collect any training data from the target user. In addition, our attack model does not rely on any consumption of the behaviors of the victim. Moreover, compared with existing works that also leverage network traffic, our system is specifically designed for AR applications and can achieve better system performance on single location detection in real time.

6.3 Preliminaries and Problem Formulation

6.3.1 Location-based Multi-player AR Overview

A typical location-based multi-player AR application runs on a mobile AR device. Users can utilize the equipped camera to record the surrounding real scene, combine the geolocation data from multiple sensors including GPS and gyroscope, and load the AR data information in real time. Then, they can make an integrated display of the acquired AR contents, such as texts, images, sounds, videos, and models. A typical location-based AR application structure is shown in Fig. 6.2. The sensor data (e.g., video and GPS information) is sent to the SDK-enabled logic layer of location-based AR applications. Location-based AR application processes the raw sensor data and requests corresponding AR contents from cloud dataset that is maintained by content providers. Then, the requested AR contents are downloaded to location-based AR applications.

For location-based AR systems, the location-based AR contents are typically stored
in a *cloud database* that is maintained by independent developers (e.g., *content provider*). There are several reasons to move AR contents storage and geolocation processing to the cloud server. First, for business reasons, since the AR service mediates all AR content retrieval, the AR application developer can inject ads, charge content providers, and keep usage statistics easily. Second, to facilitate geolocation-based channel launching, recognition of trigger GPS location is done at the server, because this involves matching against proprietary databases using proprietary algorithms. Third, the geolocation contents are always considered as “hot” data, which keep changing all the time. The centralized location processing removes the need to replicate and update the service’s geolocation content database on millions of devices, which is a computationally intensive task and would profoundly impact the actual performance of low-powered mobile devices.

Location-based AR applications are different from traditional location-based applications in terms of the content size. In general, the network traffic volume of location-based AR applications is much higher than most conventional location-based applications such as weather applications and navigation applications. Due to the large size of the AR contents, the AR applications only cache those AR contents that are within a certain distance from the AR user, which enables the attacker to estimate the AR user’s location by detecting a distinctive pattern in network traffic. Moreover, the network throughput of AR applications is much larger than that of traditional applications, and that is why 5G network is proposed to fulfill the network requirements of AR applications. In the AR scenarios, the large network throughput is much more normal than the traditional smartphone scenarios. This fact gives us a change to disguise our applications as an AR application that does not have location services, so that the network traffic introduced by fake AR contents cannot be easily noticed by the victim.

To support location-based multi-player AR experience, users can upload or delete
their AR contents with real-world GPS coordinates to the cloud database and also download AR contents when they reach those real-world GPS coordinates. Moreover, location-based AR applications must continuously analyze the device’s GPS location in order to download AR contents at the GPS location and to anchor AR objects on the screen. Cognizant of the need to facilitate the development process, several AR service providers have supplied AR client software and SDK to the developers to help them build AR applications quickly. We find that most of them (except EasyAR) provide location-based (geo API or GPS) and cloud-based (content and cloud API) services to enable location-based multi-player AR experience. Moreover, most of them issue a free license, which means more developers will use these SDKs to build location-based AR applications. Therefore, without mitigation solutions in the SDK level, the location-based and cloud-based services can be used to infer the real-time location information.

6.3.2 Key Insight

Conceptually, a location-based AR application is quite similar to a traditional desktop application. They both work on input data from the user or the database, and their state-transitions are driven by their internal information flows (both data flows and control flows). The only fundamental difference between them is that an AR application’s input points, program logic, and program states are split between the AR devices and the server, so a subset of its information flows must go through the network. We refer to them as data flows. Data flows are subject to eavesdropping on the wire and in the air, and thus often protected by HTTPS and Wi-Fi encryptions.

After the user submits the location to the server, the returned geolocation-based AR content is typically segmented at the application layer. In order to estimate the victim’s location based on the network traffic, the throughput patterns should always exist when a victim is walking along the path. Fig. 6.3 shows the downloading
throughputs of every 10 seconds when the victim who uses WallaMe walks along a path. On the path, 1, 2, and 4 AR contents (posts with pictures) are deployed at 3 different locations, respectively. Each burst represents a downloading job of AR contents when the victim reaches the location, and there is no significant network traffic between neighboring areas. We can see that the sizes of each burst and inter-burst intervals remain the same for the same AR content deployment at a different time. In fact, even if packets are encrypted using either the transport layer security (TLS) or secure sockets layer (SSL) protocol at the transport layer, their sizes and times of arrival are visible to the adversary. SSL/TLS is a separate protocol that inserts itself between the application protocol and the transport protocol (TCP) that enables applications to be only as secure as the underlying infrastructural components. This feature has been reported by many traffic analysis literature [131, 144]. Therefore, if the observable traffic feature is correlated with the segmentation in the application-layer, they can leak information about the content of the AR message.

The attacker’s goal is to infer the victim’s geolocation information from the encrypted data traffic. In other words, an attack can be thought of as an ambiguity-
set reduction process, where the ambiguity-set of a piece of data is the set containing all possible values of the data that are indistinguishable to the attacker. How effectively the attacker can reduce the size of the ambiguity-set quantifies the amount of information leaked out from the communications - if the ambiguity-set can be reduced to $1/R$ of its original size, we say that $\log_2 R$ bits of entropy of the data are lost. Similar modeling of inference attack has also been discussed in prior research, e.g., elimination of impossible traces in [145].

6.3.3 Adversary Model

In this study, we consider a capability-restricted attacker that is aiming at revealing the location of users of a specific type of AR applications, called location-based multiplayer AR. These AR applications allow users to publish or delete their own AR contents (e.g. images and messages) at any location. The attacker’s capability is restricted in the following senses:

- **It only has the access right no more than that of a standard AR user (except that it can manipulate its geolocation).** Manipulating geolocation is low-cost and easy to implement on many platforms. For example, Android allows users to manipulate location as long as developer options are activated. By manipulating its geolocation, the attacker can deploy AR contents using different accounts at any location without physically being there.

- **It can trick victims into installing its malicious applications that only require non-location-related permissions to monitor network throughput of the targeted AR application.** In our adversary model, the attacker can only trick AR users into installing malicious applications that only require non-location-related permissions, which is a common assumption in side-channel attacks (e.g. the remote attack in [131]). There are two major ways to monitor the network throughput on current smartphone systems: 1) through internal system permission; 2) adding a
virtual private network (VPN). On Android platform, the malicious application can get the network throughput of a specific application by using “read phone status and identity” permission that is widely required for many popular applications. Table 6.1 shows some popular applications that ask for “read phone status and identity” permission and their number of installation. We can observe that this permission is common, and it is hard for users to notice its potential risk of location leaking. Besides, the malicious application can also pretend as data usage monitoring applications that are popular on all platforms. For instance, My Data Manager [146] claims it is trusted by over 14.8 million uses worldwide on Apple Store. In general, these applications get network throughput by setting up a VPN. Any downloading data stream must pass the VPN before being received by an application. By using either of these two methods, the malicious application can get the real-time network throughput of the targeted AR application.

In summary, the location-based multi-player AR applications that may be used to track users’ location must have the following features: 1) high volume of real-time data; 2) outsourced geolocation processing, and open privilege of uploading AR contents. Although such systems is only a small portion of all types AR applications, their users are enough to attract attackers to launch attacks.

<table>
<thead>
<tr>
<th>Application</th>
<th>Number of installation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tmall</td>
<td>1,000,000+</td>
</tr>
<tr>
<td>Youku</td>
<td>10,000,000+</td>
</tr>
<tr>
<td>Facebook</td>
<td>1,000,000,000+</td>
</tr>
<tr>
<td>Twitter</td>
<td>500,000,000+</td>
</tr>
<tr>
<td>Uber</td>
<td>100,000,000+</td>
</tr>
</tbody>
</table>
6.4 Overview of the attack

There are three parts to the location-based side channel attack: AR users (victim), AR cloud database, and malicious user (attacker). As shown in Fig. 6.4, a complete attack can be divided into five steps. 1). The attacker uploads several specially crafted geo-objects with a fake location to the cloud database. 2). The victim posts his/her current location to query the database. 3). The database returns several geo-objects back to the victim including the crafted objects. 4). The victim downloads these objects and creates a unique traffic pattern. 5). The attacker utilizes the malicious application to keep monitoring victim’s traffic pattern and uses the reported pattern to reveal the location of the user.

6.4.1 AR Content Generation and Deployment

We first consider the simple attack scenario. In this scenario, the attacker already knows the small region where the victim is and wants to further infer the accurate location of the victim. We will discuss how to locate the victim in a large region in Section 6.4.2 for more general attack scenarios. To achieve this goal, we propose two AR content deployment strategies with different granularity and deployment cost.

**AR content generation.** There are two file formats that have been heavily adopted for displaying 3D models in AR: GL Transmission Format (glTF) and USDZ. Both are open-source format and can be generated from traditional 3D assets. The size of an AR content can be easily controlled by either adding a hidden surface or tweaking the image files (png format) that have been mapped to the 3D Model.

**Coarse-grained location detection.** To locate the victim in a detected region, the basic idea is to cut the region into several non-overlapped areas. Each area is a circle whose center is the location of AR contents and radius is the searching range of the AR application. Moreover, the size of AR content in one area is distinct from that
in any other areas. When a victim shows up in any area, corresponding AR contents will be downloaded to the device, and the attacker can infer the victim’s location based on the size of a downloading job in the network throughput. Although this strategy can locate the victim in an area with limited size of deployed AR contents, it has two key limitations. First, it cannot cover all locations in the small region since the searching area of each AR content is a circle. In some cases, victims in the region may not show up in the searching area of any AR content, so the coarse-grained location detection strategy fails to detect the location. Second, the localization granularity is relatively coarse. Without more information or deployment, we cannot infer more fine-grained location information of the victim within each non-overlapped searching area.

**Fine-grained location detection.** To address the limitation of the course-grained location detection strategy, we also propose a fine-grained location detection strategy with more deployment cost to improve the localization granularity and coverage.

The location-based AR applications set a physical sensing range for each geo-content. For instance, in Pokemon Go, the AR content “Fort” is only reachable if the
user’s distance is less than 38 m. In order to further enhance the localization accuracy and thus break the limit, we utilize a space partition attack algorithm similar to [147]. The basic idea is to divide the target area into four non-overlapping regions and thus pinpoint the victim in the space to precisely one of the regions. Fig. 6.5(a) shows an example of the space partition. Assuming the covered area of an AR content is a box, given the maximum sensing range $R$, we can place fake AR contents at the origin (illustrated as red star in Fig. 6.5(a)) to cover a large area (highlighted in yellow). To improve the localization accuracy, the attacker can also place fake AR contents at four corners (illustrated as light red star) of the highlighted area. By doing this, the attacker can locate the victim in each smaller yellow box and further enhance the accuracy to $R/2$. We could repeat this partition for multiple rounds until the expected accuracy is achieved. The whole algorithm is summarized in Algorithm 1. For the simplicity of problem presentation, we consider the area where the victim is as the box rather than the circle.

We then study the case in which the small region is fully covered by the geo-AR content. We assume that each geo-AR content is capable of covering a fixed radius
Algorithm 1 Space partition algorithm for fine-grained localization

In: Initial location \(I=(c_X, c_Y)\) and resolution \(\delta\)
Out: Location set \(P\)

1: Initial a queue \(Q \leftarrow (c_X, c_Y, \delta)\)
2: while \(\delta \geq \text{threshold}\) or \(Q\) is not empty do
3: \((c_X, c_Y, \delta) \leftarrow \text{pop} \ Q\)
4: \(P \leftarrow (c_X \pm \delta, c_Y \pm \delta)\)
5: \(Q \leftarrow (c_X \pm \delta/2, c_Y \pm \delta/2)\)

\(r\) around it. Therefore, we can model each geo-AR content as a disk with radius \(r\).

In order to cover the entire two-dimensional plan with these disks, the appropriate optimization metric should be the amount of geo-AR content used per unit area (e.g., density). We first introduce the strip-based deployment strategy (shown in the highlight part of Fig. 6.5(b)). The strip-based strategy places the geo-AR contents along a line such that the distance between the centers of any two adjacent circles is \(r\). This strategy is good for tracking a user along a given path.

In order to tile the entire plane, we need to place the geo-AR content using the strip-based strategy repeatedly. Given a 2D plane, for every even index \(k\), place a strip of geo-AR content oriented in parallel to the \(x\)-axis such that the point \((0, k(\sqrt{3}/2 + 1)r)\) is the center of a geo-AR content constituting the strip. For every odd index \(k\), place a strip of geo-AR content oriented parallel to the \(x\)-axis such that the point \((r/2, k(\sqrt{3}/2 + 1)r)\) is the center of a geo-AR content in the strip. Next, we do a similar process along the \(y\)-axis. For every odd integer \(k\), we place two geo-AR contents at \((0, k(\sqrt{3}/2 + 1)r \pm \sqrt{3}/2r)\). The full geo-AR content displacement pattern is shown in Fig. 6.5(b). It can be verified that our solution provides connected coverage to the entire two-dimensional region.

To support fine-grained localization with complete coverage, the key challenge is to propose a special AR content size sequence so that we can accurately locate the victim in any overlapped area. To address this issue, we design an AR content size generation algorithm based on super increasing sequence.
Let the sizes of crafted AR contents at different geolocations \( W = (w_1, w_2, ..., w_n) \) be a super increasing sequence. Then

\[
  w_k > w_{k-1} + ... + w_2 + w_1, \quad \text{for all } 2 \leq k \leq n
\]  

(6.4-1)

where each element \( w_i \) in set \( W \) is the size of AR contents deployed at a geolocation. Therefore, each AR content \( w_k \) has its unique size. Moreover, the combination of multiple AR contents is also unique. This property allows the attacker to place overlapped AR content, which greatly enhances the precision of our attack method.

The size of each AR content \( w_i \) can be computed based on the Algorithm 2. Note that \( c \) is a constant value picked up by the attacker to avoid overflow. Once \( W \) is generated, the attacker can then execute an AR content generation function to generate a set of location-based AR contents based on the given size \( w_i \). Note that this is an application-specific function, so the attacker may need to further alter the size (by adding or subtracting a constant value \( p \)) of each content or deployment multiple AR contents at a single location to achieve a successful attack based on the limitation of the AR application.

We can also notice that the fine-grained location detection is a special case of coarse-grained location detection. In coarse-grained location detection, each non-overlapped searching area must be a circle, while each non-overlapped searching area can be of any shape. Although fine-grained location detection can achieve better granularity and coverage, it will also introduce more deployment cost since more non-overlapped searching areas are introduced. In real-world attack scenarios, the attacker can pick either strategy based on the trade-off between performance and cost.
Algorithm 2 AR content generator

**In:** Size $n$, Constant number $c$

**Out:** Set $W$

1. **for** $i$ **in** range(1, $n$) **do**
2. \[ w_i \leftarrow \text{sum}(w_0, w_1, ..., w_{i-1}) + \text{random}(1, c) \]

Figure 6.6: Example of recursive region detection.

### 6.4.2 Recursive Region Detection

In real-world scenarios, it is usually hard for attackers to estimate the small region where the victim shows up. If we keep using proposed AR content deployment strategies for a large region, both of them will produce unlimited AR content size at some locations, which produces abnormal network traffic that the victim can easily notice and makes the attack unfeasible. In order to reduce the maximal size of the AR contents deployed at each location, we first narrow the search area by repeating partitioning a large area into four non-overlapped regions. For each partition, we deploy AR contents with the same size at all locations in each partitioned region, and the distance between neighboring AR contents is twice the length of the searching range of the AR application to avoid overlapped areas. To robustly distinguish four small regions based on the network throughput, four different sizes of AR contents for four different regions are generated based on Algorithm 2. Assuming the victim is moving, once a victim shows up in any region, our attack model can quickly identify the victim’s region. Then, our attack model deletes all AR contents and further
repeats this process in the detected region until we can finally locate the victim within a much smaller region (e.g., a block) for further accurate localization and tracking. Fig. 6.6 shows an example of our hierarchical localization. The number of possible locations of the victim can be reduced to 4 after repeating the process twice.

6.4.3 Network Throughput Processing

*Noise removal and throughput accumulation.* The collected network traffic of AR applications contains noise. On the one hand, the noise comes from various link conditions or other data exchange except downloading AR contents between the AR application and the server. On the other hand, based on our experiments, the network throughput not only counts the bytes of the content in packets but also counts the bytes in packet header or other information within the packets. So, the raw network traffic data cannot be directly used to parse the real-time locations. In order to accurately track victim’s location using network traffic, we first eliminate small traffic that cannot be caused due to downloading AR contents from the server based on a threshold $\tau$. Moreover, we need to accurately estimate the network throughput downloaded at each location in order to infer the victim’s location based on the special throughput pattern. Since the AR contents are not downloaded immediately, we need to accumulate the network throughput within a moving time window of length $T$ in order to accurately estimate the size of AR contents deployed at each location. The length of the moving time window is set as:

$$T = \left\lceil \frac{\max(W)}{\lambda} \cdot F_s \right\rceil + 1 \quad (6.4-2)$$

where $\max(W)$ is the maximal size of AR contents, $\lambda$ is the average downloading speed of the AR application measured based on the history, and $F_s$ is the sampling rate of network traffic monitoring. As we observe in Section 6.3.2, there is no
significant network traffic between two neighboring locations, which means that the size of AR contents at each location is a local maxima in accumulated throughput sequence. Based on this observation, we find the size of AR contents by finding the local maxima in the accumulated throughput sequence. Each local maxima represents a single location where the AR contents are deployed or a location that is within the searching range of AR contents deployed at multiple locations. Fig. 6.7 shows an example of our network traffic processing. The raw network throughput is collected from WallaMe when the victim passes three locations where 1, 2, and 3 pictures are deployed respectively. After throughput accumulation, we can observe three local maximas (blue markers) in accumulated throughput that correspond to three downloading jobs in raw network throughput.

**Localization.** The localization algorithm works as follows: given a local maxima $S$ in accumulated throughput sequence and generated AR content size set $W$, we aim to infer the location $X = (x_1, x_2, ..., x_m)$ of the victim. $X = (x_1, x_2, ..., x_m)$ means the location that is within the searching range of AR contents deployed at $m$ different locations, and $m$ is a positive integer. For the coarse-grained strategy, $m = 1$. For the fine-grained strategy, since the integers in set $W$ form a super increasing sequence,
Algorithm 3 Localization algorithm

In: A local maxima in accumulated throughput sequence \( S \), generated AR content size set \( W \)

Out: Inferred location \( X \)

1: \( X = \emptyset \)
2: \( n \leftarrow \text{sizeOf}(W) \)
3: for \( i \) in range\((n, 1)\) do
4: if \( S > w_i \) then
5: \( X = X \cup x_i \)
6: \( S \leftarrow S - w_i \)
7: else
8: \( x_i \leftarrow 0 \)

we can prove that the inferred area \( X = (x_1, x_2, ..., x_m) \) is unique if \( X \) exists. The location \( X \) can be computed by the following localization algorithm (Algorithm 3).

After obtaining the location of each local maxima, we further calibrate the localization results. If the victim is detected in the overlapped area of a set of locations, we will double check the physical distance among locations in the set. If the searching ranges of locations in the set do not have a common overlapped area, we argue that the received throughput is noisy and the victim is at the feasible overlapped area of a subset of locations whose total size is maximal. The trajectory of the victim is recovered once all of its locations on the path are obtained. However, due to inaccurate GPS data, a victim may receive the data that is deployed by the attacker more than one time. To remove redundant information, for a sequence of continuous and identical location estimations, we only reserve one of them.

6.5 Experimental Setup

6.5.1 Implementation

We built a real testbed in order to effectively evaluate the attack methods we propose. Our testbed included four parts: an Android application for monitoring network traffic, an Android application for imitating the behaviors of current AR
applications, a customized location provider for location spoofing, and a back-end server that receives the requests from all AR clients and returns corresponding data. Simple graphical user interfaces (GUI) are designed to help subjects to collect data. We illustrate the system design and implementation in detail in the following paragraphs.

**Network traffic monitoring.** The core part of our system is accurately monitoring the network traffic of a specific application. To achieve this goal, we studied the feasibility of monitoring network traffic on Android platform. NetworkStatsManager can provide access to network usage history and statistics of other applications, which enables an attacker to implement a listener in another application on a victim’s device. Although NetworkStatsManager needs “read phone status and identity” permission that is a protected permission, a lot of popular Android applications ask for this permission, as shown in Table. 6.1. This fact enables the attacker to hide this listener in a popular application without being noticed by the victim. To get the real-time network, we created a background service that can log the total network usage every second. The throughput of each second was acquired by calculating the difference between neighboring entries in the log file.

**Location spoofing.** Location spoofing is used to generate mock locations, so that the attack can deploy fake AR contents at any location without physically being there. Moreover, other AR users can also deploy AR contents, which may change the pattern of network traffic on the victim’s device and break our attack model. To address this problem, the attacker needs to know the size of AR contents deployed by AR users at those locations the victim may appear, which can also be solved by leveraging location spoofing. Before deploying fake AR contents, the attacker first sends fake geographical locations where he/she wants to deploy fake AR contents to the server. The attacker monitors the network traffic that reflects the size of AR contents deployed by normal AR users. Based on the size of existing
AR contents, the attacker rearranges the size of fake AR contents deployed at each location. Most of the Android applications acquire location via a location provider (e.g. “network” or “GPS”) from the location system service. However, it is possible to add customized location providers under certain circumstances such as debugging. The attacker needs to enable “Allow mock location” option in the developer options of their Android device before getting access to the mock location API. This API asks for five variables: latitude, longitude, altitude, speed and accuracy. Typically, the location-based AR applications only utilize the latitude and longitude values to determine the user’s current location.

**Location-based AR application.** We first studied several state-of-the-art AR applications and SDKs (e.g., Google AR and Wikitude) and found that these AR applications and SDKs send local information (e.g., locations and images) to the server using a simple HTTP(S) GET requests. After getting the requests from the client, the AR server serializes returned information into a structured data (e.g. JavaScript Object Notation, Extensible Markup Language, and Protocol Buffers) using HTTP protocol and returns it to the AR application. Extended studies show that all existing AR applications and AR SDKs are based on the same mechanism. Therefore, our AR application is equivalent to most existing applications or future applications developed using current SDKs in terms of data transmission and communication.

Therefore, we built a location-based Android application to imitate the behavior of current AR applications. The application keeps collecting GPS information, sends it to our back-end server using GET request, and receives corresponding data from the server. We further tested it and ensured that our AR application has the same behavior of network traffic and mechanism for data transmission. To ensure the GPS locations sent to the server are accurate, we only sent a GET request to the server when the accuracy of measured GPS data was within 8 m. Although our self-built AR application had most features of real AR applications, we could not perfectly
simulate and reproduce all behaviors of real AR applications. To further show that our attack model is feasible to be launched on real AR applications, we also evaluated our attack model on WallaMe.

**Back-end server.** We implemented our server on a public IP address based on HTTP(S) protocol. The back-end server receives requests from all mobile clients, analyzes their geographical location, and returns corresponding AR contents. For each request, we first compared its GPS location to those of all deployed AR contents. If the user appeared around one or more AR contents, we would generate a temporal file with the corresponding size in milliseconds and return it in the response.

### 6.5.2 Data Collection

To evaluate the performance of our attack model, we conducted various experiments on our testbed on a campus, as shown in Fig. 6.8. On the path, we uniformly picked 8 locations on the map. The distance between neighboring locations was about 60 m. The searching range of each location was set to different values to evaluate the performance of our attack strategies on detecting single location and detecting the overlapped area. For coarse-grained location detection, the searching
radius is about 20 m, and the size of AR contents at the first location was set to 1 KB and was increased by 1 KB for each of the following locations. For fine-grained location detection, the searching radius is about 45 m, and we deployed AR contents whose total sizes follow the rule of super increasing sequence at each location. The minimal size of deployed AR contents was also 1 KB. We control the size of AR contents at each location by 2 ways: 1). Deploying more AR contents with equal size. 2). Changing the size of a single AR content by adding more information (e.g. pictures with specific sizes) to the content. 12 healthy volunteers with their ages ranging from 21 to 26 were involved in the study. Among 12 volunteers, we asked 8 of them to use our self-built AR application and the other 4 of them to use WallaMe for testing. We collected 10 trials from each volunteer, and each trial lasted for about 10 minutes. During each trial, the volunteer was required to pick a path and pass different locations while opening two applications (the AR application and the network monitoring application) and connecting their devices to their personal hotspot. The server returned corresponding AR contents based on the real-time location of the user without introducing extra traffic. Besides recording the network traffic of the AR application during each trial, we also logged the received GPS coordinates as the ground truth.

6.5.3 Evaluation Metric

The location reported by our system is not the real geolocation with 2-dimension coordinates but a non-overlapped area. The size of the reported area is determined how the attacker perform coarse-grained and fine-grained location detection. Therefore, instead of using the distance as a metric, we evaluate the system performance based on how accurately our system can correctly locate the victim in a area. Here a correct detection means that our attack system detect that the user is in an area when the user is exactly in that area. The location detection accuracy
Accu is defined as:

\[ \text{Accu} = \frac{L_{\text{correct}}}{L_{\text{all}}} \]  

(6.5-3)

where \( L_{\text{correct}} \) is the number of correctly detected locations (areas) and \( L_{\text{all}} \) is the number of all locations (areas) that the victim has passed.

6.6 Experimental Results

6.6.1 Performance of Location Detection

Single location (non-overlapped area) detection. Since the performances of both the recursive region detection and localization strategies are based on how accurately we can detect the victim at a location, we first evaluated the performance of our attack model on the single location detection using our self-built AR application. We removed the noise in the raw data and processed it with our location detection algorithm. Then, we compared the detection results with the ground truth. The location detection accuracy is defined as the number of correct detections divided by the total number of detections.

Fig. 6.9 shows the single locations detection accuracies for all locations by using either the coarse-grained location detection strategy or the fine-grained location...
Table 6.2: Location detection accuracy for the overlapped area.

<table>
<thead>
<tr>
<th>Location with</th>
<th>More AR content</th>
<th>Less AR contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>100%</td>
<td>99%</td>
</tr>
</tbody>
</table>

detection strategy. Evaluation results show that our coarse-grained location detection strategy can locate the victim in non-overlapped areas with a mean accuracy of about 94.6%. Since the size differences of AR contents deployed at different locations are much greater in fine-grained strategy by using *super increasing sequence*, it can provide a better average accuracy of about 97.1%. Moreover, we notice that the location detection accuracies are slightly lower for those locations where more AR contents were deployed. The reason is that the downloading process of large files is easier to be influenced by unstable network, which breaks the special patterns in network throughput.

**Overlapped area detection.** In our fine-grained location detection strategy, the victim may also appear in the searching areas of AR contents deployed at multiple locations. Instead of assigning the victim to one location, we would like to locate the victim in the overlapped area accurately. To evaluate how accurately our system can detect the overlapped area, we evaluated how accurately the victim can be located in the overlapped area of two searching areas. Since the sizes of AR contents deployed by the attacker are unique at different locations, one location should have more AR contents deployed than the other one. We repeated the experiment for 100 times, and Table. 6.2 shows the detection accuracy for the location with more AR contents and the location with less AR contents. We can see that our location detection model can accurately detect the location with more AR content, and the percentage of the wrong prediction for the location with less AR contents is no more than 1%. These results show that it is feasible to locate the victim even if he/she is in an overlapped area.

**Granularity of location detection.** There is a trade-off in how densely
Table 6.3: Location-detection accuracies with different distances between neighboring locations.

<table>
<thead>
<tr>
<th>Distance (meter)</th>
<th>70</th>
<th>27</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>100%</td>
<td>98%</td>
<td>60%</td>
</tr>
</tbody>
</table>

The attacker should deployed the AR contents in a small region. If we deploy AR contents at many different locations, we can estimate victim’s locations with a better granularity, but the location detection accuracy may not be good due to inaccurate GPS coordinates. Also, the network usage is also higher, which makes our attack easy to be noticed by the victim. In fact, the fewer locations where we deploy AR contents, the better the detection accuracy is expected to be, but more details of the victim’s trajectory are lost. In order to study how densely the AR contents can be deployed with a good detection accuracy in our attack model, we adjusted the distance between neighboring locations and studied its influences on location detection using our self-build testbed. In this experiment, the searching range of AR contents at each location was about 20 m. We asked a volunteer to walk along the same path for 10 times. Along the path, we deployed AR contents at as many locations as possible with the distances between neighboring locations were about 70 m, 27 m, and 13 m, and the results are shown in Table 6.3. When the distance is larger than 27 m, ARSpy can achieve an excellent location detection accuracy of at least 98% since at most 14% of the searching area is overlapped with those of neighboring locations. The accuracy drops to 60% when the distance is about 13 m. Considering the deviation of GPS measurements is from 3 m to 8 m in outdoor environment, the noisy GPS data cannot reflect the real-time location of the victim relative to each location where AR contents were deployed. If GPS data is inaccurate, the server would not consider the victim is at that location. Therefore, the victim’s device would not download the AR contents deployed by the attacker, and the attacker cannot track the victim based on the network throughput.
Table 6.4: Performance of trajectory construction.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Avg Levenshtein distance</th>
<th>Exact fit</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARSpy</td>
<td>96.72%</td>
<td>0.0345</td>
</tr>
<tr>
<td>RG</td>
<td>13.75%</td>
<td>0.7665</td>
</tr>
</tbody>
</table>

6.6.2 Performance of Trajectory Construction

It may also be beneficial for the attacker to know the actual route through which the victim traverses on his way to the destination. For this purpose, we also calculate for each constructed trajectory the Levenshtein distance [148] between it and the actual trajectory. The Levenshtein distance is a standard metric for measuring the difference between two sequences. It equals the minimum number of updates required to change one sequence to the next. The distance is normalized by the length of the longer trajectory of the two. This allows us to measure the accuracy of the algorithm for estimating the full trajectory the user traversed. For each estimation, we also note whether it is an exact fit with the actual route (i.e., zero distance). The percentage of successful localization of destination, average Levenshtein distance, and percentage of exact full route fits are calculated for each type of estimated route. To benchmark the results, we note in each table the performance of a random guess algorithm which outputs merely a random but feasible route.

We evaluate the performance of the trajectory construction using the dataset for single location detection, and Table 6.4 illustrates the trajectory construction performance of ARSpy and the random guess (RG)-based attack. Compared with the random guess-based attack model, ARSpy can achieve much better trajectory construction performance. Moreover, ARSpy can accurately predict the destination of the victim with a high accuracy of 96.72%. Although the percentage of exact full route fits is 77.5%, we can note that the average normalized Levenshtein distance is only 0.0345, which means only one or two locations are wrongly detected for a path with eight locations even if the constructed trajectory does not fit the real trajectory.
Figure 6.10: System performance for three paths.

The high percentage of successful localization of destination and the low percentage of exact full route fits show that ARSpy can accurately track the victim for a more prolonged path.

6.6.3 Performance on A Real AR Application

Experimental results show that our attack methods can achieve high performance on our self-built AR application. To show the feasibility of our attack methods on a large range of real AR applications, we evaluated the performance on WallaMe.
WallaMe is a free AR application that allows users to take a picture of a surface around them and add information (e.g. words, stickers, and photos) on them. Once the picture is posted, it will be geolocalized and visible by everyone passing by. Also, the user who uploads the picture can make the pictures private, which means that the picture is visible only to specific groups of people. In this experiment, we evaluated how accurately three locations can be detected by our attack methods. Specifically, we deployed 1, 2, and 4 pictures at three locations, and the sizes of those pictures are nearly equal. Since the searching radius of WallaMe is about 1 block, the distance between each possible next location and the current location is about 180 m to generate overlapped areas between two neighboring locations. We ask four volunteers to pass these three locations for 20 times. Based on the accuracies of GPS measurements, each volunteer was regarded at a single location or the overlapped area of two locations by WallaMe. Experimental results show that our system can successfully detect the single location or the overlapped area with an average accuracy of 90%, which implies that our attack model is also feasible to be launched on existing AR applications.

6.6.4 Influence of Different Paths

To show the generalization of our attack model, we further evaluated the location detection accuracy for three other paths. As shown in Fig. 6.10(a), a volunteer was asked to walk through each region along the colored path. The three paths were carefully selected to cover different types of outdoor scenarios. For example, the regions of the black path and the yellow path have high buildings of at least eight floors. We used these two regions to evaluate the attack performance with inaccurate GPS measurement due to high buildings. The region of the red path has low building of at most four floors. Across all experiments in this subsection, we used our coarse-grained location detection strategy to deploy AR contents at different locations. The
Figure 6.11: Location-detection accuracies on two devices.

distance between neighboring locations is about 50 m, and the searching radius of each location is set as 20 m. The experimental results are shown in Fig. 6.10(b). We can see that our system can provide location detection accuracy for at least 85.7% even if the GPS measurements are influenced by the high buildings.

6.6.5 Performance on Different Devices

In our attack model, we assume that the attacker does not know what smartphone the victim uses. Therefore, we further conducted experiments to evaluate the effectiveness of our attack model on different devices using the deployment configuration in Fig. 6.8 using the coarse-grained strategy. We chose Google Nexus 5 and Nexus 6 as devices in this experiment. During the experiment, we asked a volunteer to hold two devices while walking along the path for ten times. Fig. 6.11 shows the location detection accuracy on two devices. We found that Nexus 6 has a better performance than Nexus 5. The reason is the sampling rate of GPS data. In most cases, both of the two devices can receive a GPS coordinate every 1 second. However, Nexus 5 needs to wait for more than 1 second to get the next GPS coordinate at a probability of 1.39% in our experiments, while Nexus 6 only has this issue at a probability of 0.93%. Moreover, the maximal delay of receiving the next GPS coordinate on Nexus 5 was 22 seconds, while that on Nexus 6 was only 12 seconds.
Considering the walking speed is about 1.4m/s, a victim using Nexus 5 is more likely to miss a location due to the long delay. Even if the victim’s device may lose some GPS coordinates, our attack model can achieve high location detection accuracy on both of the two devices, and at most one location was missed in each trial. The high location detection accuracy indicates that our attack model is feasible on different devices, which means that the attacker can deploy this attack on any victim who is using AR applications.

6.6.6 Performance on Large-scale Long-Term Tracking Simulation

According to [149], the top N locations inferred from human mobility data can be used to reveal the identity of a user. For instance, top two locations may link to user’s home and work locations, top three locations may correspond to user’s home, work and shopping locations. [150] shows that the human mobility traces are highly unique and more than 95% of the individuals could be uniquely identified based on the top four locations. In this subsection, we discuss the performance on long-term tracking of the top N locations inferred by the user’s network throughput data. The simulation is based on the GeoLife Dataset [151], which contains GPS trajectories of 182 users in a period of over three years. We replayed the GPS data (as ground truth) to simulate user moving trajectory in a location-based AR application that we created and then used ARSpy system to launch attack and infer user’s locations. Our simulation assumes that the attacker has some pre-knowledge of the target, and knows the city that he/she lives in. We set the detection range of each AR content equal to 1.2 km. Shown in Fig. 6.12, our AR attack method is able to deduce at least top four locations for more than 50% of the user data and achieves 86% detection rate for the top two (and above) locations. This means that the attacker can infer these users’ home and workplace solely based on the network throughput data.

Fig.6.13 illustrates the relationship between the number of deployed AR contents
According to the results, to track an individual user in a city, the attacker needs to deploy at least 1000 fake AR contents to the server to bring down the error to around 30 m. However, the actual number depends on the detection range of the AR contents and the size of the tracking area. The results show that our proposed algorithms can be used for long term tracking and is able to correctly infer top $N$ locations of the user with high accuracy, which means that the attacker can track a target even if the server puts a restriction on the AR content update rate.

### 6.6.7 Influence of Traffic Noise

The logged throughputs always contain noise. The noise can be from other applications on the same device. For example, the downloading jobs of other applications will cause network congestion, which may change the traffic pattern.
of the AR application and make it difficult for the attacker to recognize deployed AR contents. On the other hand, the noise can also be from other downloading jobs generated within the same AR application. For example, the AR application needs to synchronize with the server and download relative contents. This kind of traffic can be wrongly recognized as AR content deployed by the attacker, which leads to the attacker being unable to construct the trajectory of the victim.

To evaluate the influence of downloading jobs of other applications, we let a volunteer walk along the path in Fig. 6.8 for 10 times while using our self-built AR application. At the same time, the volunteer downloads a large file via Google Play on the same device. Experiment results show that all locations can be detected, which implies that the downloading jobs of other applications will not destroy the traffic pattern of AR application and thus do not influence the location detection performance of our attack model.

In order to evaluate the location detection performance under the influence of extra traffic generated by the AR application, we let the server send extra data to our AR application based on the network throughput distribution of Ingress and Pokemon. In this experiment, the threshold \( \tau \) is set to the minimal size of fake AR contents in order to remove the influence of extra network traffic. Since super increasing sequence determines the size of AR content deployed at each location, the smallest size of AR contents at all locations should be as small as possible. We set the smallest size to different values in order to evaluate what is the smallest size of AR contents required to ensure good location detection accuracy. Experimental results show that the location detection accuracy rises with the increase in size of fake AR contents. When the size of the smallest fake AR content is 20 \( KB \), ARSpy can provide location detection accuracy of at least 92%, which proves that our attack model is feasible even if the AR application frequently exchanges extra data with the server.
6.6.8 Influence of Different Speeds

In subsections 6.6.1 and 6.6.2, we only evaluate the performance of our attack model in the scenario where the victim is walking. In this subsection, we evaluate the location detection performance of our attack model when the victim is jogging at a speed of about 2.5 m/s. Here we do not consider the scenario where the victim is running at high speed since more network bandwidth and computation resources are required in this scenario, and few AR applications are designed for running. In this experiment, we ask a subject to run two applications we build while jogging along the path on the campus. Experiment results are shown in Fig. 6.14. It is clear that our attack model can achieve almost 100% location detection accuracy for all 8 locations. By comparing the ground truth in both walking and jogging scenarios, we find that the trajectory in jogging scenario can be approximately produced by decimating the trajectory in a walking scenario by 2. As long as the searching radius (12 m in our system) can tolerate the displacement of the victim between two GPS measurements (about 5 m while jogging), our scheme can still track the victim even if the victim is moving at high speed.

6.6.9 Battery Consumption

Besides accuracy and robustness, battery consumption is another important issue we need to consider when performing an attack. If an attack model requires a significant portion of available CPU time, the significant battery drain can be quickly noticed by the victim. Current security solutions can detect a variety of attacks by sensing abnormal battery behavior and energy patterns [152]. In our attack model, the network traffic monitoring is the key way to perform the attack and may cause battery drain. In order to evaluate the battery consumption of our network traffic monitoring, we used Batterystats and Battery Historian [153] to collect battery data. Battery Historian converted the report from Batterystats into an HTML visualization in the
browser and provided the battery data in a process level. During the experiment, we ran the application for about 75 minutes while all the other applications on the target smartphone remained closed and the screen kept on. Experimental results show that our network traffic monitoring application consumed about 0.03% of the total energy, while the battery consumption of GOOGLE_SERVICE was 0.05%. The results show that our attack model only introduces insignificant battery drain that cannot be detected by the victim and the battery behavior-based security solutions.

6.7 Mitigations

The cause of privacy leaks in location-based AR application is that, for the same path, the network throughput changes over the time is in a unique and identifiable way. Segmenting the returned data may reduce the granularity of the leak but does not prevent the attacker from revealing the location. A straightforward solution would be to store all AR contents locally (like Pokemon Go) to totally eliminate the potential information leak since the AR applications do not download contents from the server in real time. However, this solution is not suitable for large-scale AR systems which contain tons of ever-changing 3D AR models. Another solution would be padding each packet to achieve constant-size encoding to eliminate the leak, at the risk of a very inefficient encoding scheme since it would require transmitting more
redundant traffic than the actual content size. Similarly, implementing a tight rate control mechanism would result in an inefficient transmission protocol.

In order to limit the capability of the attacker, we propose 3 possible mitigation methods. First, SDK providers or developers can deploy and maintain an active cache with variable size to store the AR contents on the client side. The AR contents can be not only downloaded to the cache when the AR user reaches the location but also prefetched from the server based on the victim’s location and movement pattern. Once the AR contents are prefetched to the cache, they can be enabled to be displayed by sending a control signal instead of completely re-downloading it. Thus, the attacker has to know the detailed implementation of such variable cache and predict the victim’s movement in the same way as the server. Otherwise, the network throughput pattern can be destroyed, and the attacker cannot reconstruct the victim’s trajectory.

Second, the developers of AR application can put more limitations on AR users. For example, any AR user cannot deploy too many AR contents at a single location. Meanwhile, the size of each AR contents should not be too large. By doing this, the capability of the attacker is greatly limited since the property of super increasing sequence is hard to be satisfied. For example, the general deployment strategy cannot work since the maximal number of AR contents is limited.

Another method is to further limit the permission of network traffic monitoring on the victim’s devices, which means third-party applications cannot get the network traffic information. Existing mobile operating systems have noticed the potential threat of exposure of network traffic information. For example, Android protects network traffic information using “read phone status and identity” permission, but the user can still be deceived to install malicious applications since many popular applications also ask for this permission. Similarly, network content filters are not
permitted for regular applications in Apple store, but the attacker can disguise the malicious application as a normal application (e.g. Sift [154]) and deceive users to install the malicious application on the device. To address this issue, the network traffic data should be visible only to the operating systems, and the users should be alerted if the network traffic information is being monitored by any service.

6.8 Discussion

Scaling our approaches to various AR applications. Different AR applications may use different compressing algorithms, which results in different traffic patterns for the same AR content and influences the geolocation estimation. However, as long as the developers do not change the way to store and deliver the AR contents, the attacker can still know the relationship between the size of fake AR content and the traffic pattern after collecting enough data of a new AR application.

Influence of other AR contents from other users. In real scenarios, there may also be AR contents from other users around some geolocations, which alters the traffic pattern on the victim’s device. The attacker can address this issue by monitoring the size of all AR contents at each geolocation periodically. If some AR contents are already deployed by other users at a particular location, the attacker can change the size of fake AR contents accordingly so that the total size of fake and normal AR contents at each geolocation meets the requirement of either coarse-grained or fine-grained localization. The attacker can dynamically change the cycle time based on the size of interested region and cost. Even if other users frequently change the network traffic profile at a locations, the attackers can give up the current attack and restart attacking the victim when the victim reaches a better area. Although the attackers can lose much location information of the victim, but these limited information can still be aggregated with other data to infer more locations.
of the victim that are not detected by our system. For instance, [15] shows that the human mobility traces are highly unique and more than 95% of the individuals could be uniquely identified based on the top four locations. Therefore, the other locations of the victim in a day can be easily inferred by combining our detection results with other anonymous location dataset.

**Limitations and future work.** Our system involved a limited number of participants, and all users are university students. To better understand the performance of our system, more participants with more diverse backgrounds must be engaged. Also, the experiments were all conducted within 6 months. Considering that human behaviors and habits (e.g., speed of walking) may change, a long-term evaluation should be conducted. Besides, we only used WallaMe as an example to show the effectiveness of our system on real AR applications. In the future, we plan to evaluate our system for more AR applications and study how behaviors of different AR applications influence the performance of our attack model.

In our testbed, considering most location-based AR applications are designed for outdoor scenarios, we only tested our system in outdoor environments. In future work, we plan to implement our system for indoor AR applications that has indoor localization and computer vision techniques. Moreover, we will study using machine-learning techniques to improve the accuracy and robustness of location detection for real AR applications.

6.9 **Chapter Summary**

The booming of third-party SDKs allows the developer to create many interesting location-based AR applications. However, most users and application developers are unaware of the risk of potential location privacy leakage of their applications. Unlike smartphone where you can control when to turn on or off the sensors and applications,
the mobile AR device continuously receives inputs from the environment through multiple sensors and the network. In this chapter, we develop a novel user location tracking system – ARSpy, which could achieve accurate and involuntary tracking of the target by only monitoring the network throughput. Our real-world attack experiments on the Android platform show that our attack method achieves high localization accuracy and the attacker can recover the victim’s moving trajectory with high possibility. We have also proposed three defense mechanisms to mitigate such threats. Our study is expected to urge AR application developers to revise their geolocation transmission protocol and, more importantly, serve as a call for more attention from the application user and AR SDK designers to have the full knowledge of the potential risk brought by the location-based AR applications. This work in this chapter is published in the fifth paper of the publication list. Our work makes the following contributions:

- We propose strategies for generating and deploying fake AR contents in order to track the victim precisely. Also, we discuss the processing schemes of raw throughput data and the algorithm for reconstructing the victim’s trajectory.

- We implement our attack algorithms and build an automated user location tracking system. The real-world experiments on Android platform show that we could obtain the geolocation of any target with mean accuracy of at least 94.6% and perfectly reconstruct the victim’s trajectory with an accuracy of 77.5% in a small area. Moreover, our attack algorithms can infer at least top two locations with high accuracy of 86% based on a city-scale simulation.

- We discuss three potential mitigation methods to present this type of information leak in location-based AR applications and point out directions for a continuation of this work.
CHAPTER 7

CONCLUSION

In this dissertation, we conducted a in-depth study of the society and privacy issues in mobile CPS, including smartphones, AR devices, and IoT devices. These mobile CPSs have various sensors embedded to collect rich sensor data and also provide different types of side-channel information.

We first investigated the voice replay attacks on mobile CPS and proposed two voice liveness detection systems for smartphone and AR headsets, respectively. Then, we studied the face forgery attacks in real-time video chat scenarios and proposed a general and low-cost face liveness detection system. Finally, we designed a protection system for current PIN-based authentication systems by leveraging the information collected from IoT devices.

7.1 Summary of contributions

Mobile CPS is vulnerable to various of attacks. One major reason behind this is that current mobile CPS assume the collect information to be always trustworthy. Therefore, once attackers can generate and send malicious information to current mobile CPS, the security and privacy on these devices are not under protection any more. In this dissertation, we focused on three feasible attacks on mobile CPS and proposed corresponding defense systems.

Voice replay attacks aim to break voice-based systems by replaying modulated audio signals to the victim’s device. However, since attackers can easily get high-quality recording of the victim’s voice and replay it using loudspeakers, such attacks...
are hard to defend using traditional methods. To address this issue, we proposed a new voice liveness detection system for the smartphone platform. The key insight is that human voices are generated at multiple positions in the human vocal system. By collecting the throat voice using the second microphone on the smartphone, we can get the human voice in two channels, and these two voices have certain relationship. By measuring this relationship, our system can determine whether these voices are from a live person or replayed by a loudspeaker. Moreover, to avoid that attackers steal the victim’s throat voice from database, we further design an random noise-based scheme to differentiate whether the throat voice is new or stolen.

AR headsets have the potential to be widely used in the future, but they are also vulnerable to voice replay attacks. Considering current voice liveness detection systems cannot be implemented on AR headsets, we built a novel voice liveness system for AR headsets. The basic insight is that human voice can propagate through internal body, and the internal body voice has strong relationship with the mouth voice. Moreover, the internal body voice can be recorded by attaching a contact microphone around the user’s temple. We conducted extensive experiments and showed that our system can accurately defend against voice replay attackers even they try to generate internal body voice themselves.

Current IoT devices have great sensing capabilities and small sizes, which can be used to build more powerful defense systems for current mobile CPS. In this dissertation, we investigated how PIN input influences the received light at the PIN pad. We found that different PIN input behaviors generate different light signals, and the biological differences across people can also introduce variances to the light signals. Based on this insight, we proposed a defense system for current PIN-based authentication systems against PIN replay attackers. Experimental results showed that our system can effectively reject attackers who obtain the victim’s PIN number. Moreover, even attackers try to imitate the victim’s PIN input behavior, our system
can still defend against them with a high accuracy.

7.2 Future research

Numerous problems need to be addressed in the security and privacy area for mobile CPS. The long-term goal of my research is to solve practical problems and security issues for mobile cyber-physical systems by leveraging artificial intelligence and sensing techniques. Some potential research topics are as follows:

- **Trustworthy Computing and Communication in AR Devices.** Unlike existing smartphone devices, AR devices continuously sense the environment and get input from the user’s field of vision through video, audio, and other sensors. Moreover, the massive amount of data created by AR systems is coupled with the unique and stringent requirement on the underlying wireless network. Therefore, both cyber and physical attacks may lead to malfunction and subsequently disruption or failure of the AR systems. A security framework should be proposed to ensure acceptable network conditions and trustworthy input streams.

- **Robust and Cross-Domain Mobile Sensing Techniques.** Most mobile sensing techniques are built based on an assumption that the domain knowledge (e.g., signal patterns and theoretical models) does not change over time. However, the sensor data may be very sensitive to the change of domains. For instance, the channel state information of Wi-Fi signals is easy to be influenced by object displacement. To provide good performance, current mobile sensing techniques have to re-collect enough data in every new domain or environment, which introduces much training cost and makes them impossible to use in practice. Considering these facts, an interesting research topic is to propose cross-domain mobile sensing platforms that can be quickly launched in any new
domain with low or even zero training costs.

- **Secure Neural Network-based Classification Models.** Neural network-based models have been widely used in current mobile sensing systems for accurate and robust results. However, neural network-based models lack robustness against adversarial examples. These examples are generated from regular inputs by adding carefully selected perturbations. An extra protection layer should be proposed to protect current neural network-based models from adversarial examples.
PUBLICATIONS

Publications Included in This Dissertation


Other Publications

*Journal Publications*


**Conference Publications**


[8] D. Mukhopadhyay, M. Shirvanian, and N. Saxena, “All your voices are belong to


http://www.real-statistics.com/correlation/basic-concepts-correlation/


[120] J. A. de Guzman, K. Thilakarathna, and A. Seneviratne, “Security and


[137] L. Lu, E.-C. Chang, and M. C. Chan, “Website fingerprinting and identification


