12-2019

When keystroke meets password: Attacks and defenses

Ximing LIU
Singapore Management University, xmliu.2015@phdis.smu.edu.sg

WHEN KEYSTROKE MEETS PASSWORD: ATTACKS AND DEFENSES

LIU XIMING

SINGAPORE MANAGEMENT UNIVERSITY
2019

When Keystroke Meets Password: Attacks and Defenses

by

LIU Ximing

Submitted to School of Information Systems in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Computer Science

Dissertation Committee:

Yingjiu LI (Supervisor / Chair)
Associate Professor of Information Systems
Singapore Management University

Robert DENG Huijie (Co-supervisor)
Professor of Information Systems
Singapore Management University

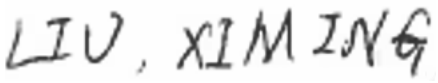
Debin GAO
Associate Professor of Information Systems
Singapore Management University

Tieyan LI
Head of AI Security
Huawei Singapore Research Center

Singapore Management University
2019

I hereby declare that this PhD dissertation is my original work
and it has been written by me in its entirety.
I have duly acknowledged all the sources of information
which have been used in this dissertation.

This PhD dissertation has also not been submitted for any degree
in any university previously.



LIU Ximing
06 December 2019

When Keystroke Meets Password: Attacks and Defenses

LIU Ximing

Abstract

Password is a prevalent means used for user authentication in pervasive computing environments since it is simple to be deployed and convenient to use. However, the use of password has intrinsic problems due to the involvement of keystroke. Keystroke behaviors may emit various side-channel information, including timing, acoustic, and visual information, which can be easily collected by an adversary and leveraged for the keystroke inference. On the other hand, those keystroke-related information can also be used to protect a user's credentials via two-factor authentication and biometrics authentication schemes. This dissertation focuses on investigating the PIN inference due to the side-channel information disclosure and exploring the design of a new two-factor authentication system.

The first work in this dissertation proposes a user-independent inter-keystroke timing attack on PINs. Our attack method is based on an inter-keystroke timing dictionary built from a human cognitive model whose parameters can be determined by a *small* amount of training data on any users. Our attacks can thus be potentially launched in a large scale in real-world settings. We investigate inter-keystroke timing attacks in different online attack settings and evaluate their performance on PINs at different strength levels. Our experimental results show that the proposed attack performs significantly better than random guessing attacks. We further demonstrate that our attacks pose a serious threat to real-world applications and propose various ways to mitigate the threat.

We then propose a more accurate and practical PIN attack based on ultrasound, named UltraPIN, in the second work. It can be launched from commodity smartphones. As a target user enters a PIN on a PIN-based user authentication system, an

attacker may use UltraPIN to infer the PIN from a short distance without a line of sight. In this process, UltraPIN leverages on smartphone speakers to issue human-inaudible ultrasound signals and uses smartphone microphones to keep recording acoustic signals. It applies a series of signal processing techniques to extract high-quality feature vectors from low-energy and high-noise signals. Taking the extracted feature vectors as input, UltraPIN applies a combination of machine learning models to classify finger movement patterns during PIN entry, and generates a ranked list of highly possible PINs as result. Rigorous experiments show that UltraPIN is highly effective in PIN inference and robust to different attacking settings.

Keystroke timing information and keystroke typing sounds can also be used to protect users' accounts. In the third work, we propose Typing-Proof, a usable, secure and low-cost two-factor authentication mechanism. Typing-Proof is similar to software token based 2FA in a sense that it uses password as the first factor and uses a registered phone to prove the second factor. During the second-factor authentication procedure, it requires a user to type any random code on a login computer and authenticates the user by comparing the keystroke timing sequence of the random code recorded by the login computer with the sounds of typing random code recorded by the user's registered phone. Typing-Proof achieves good performance in most settings and requires zero user-phone interaction in most cases. It is secure and immune to the existing attacks to recent 2FA mechanisms. In addition, Typing-Proof enables significant cost savings for both service providers and users.

This dissertation makes contributions to understanding the potential risk of side-channel information leaked by keystroke behaviors and designing a secure, usable and low-cost two-factor authentication systems. On the one hand, our proposed side-channel attacks make use of human cognitive model and ultrasound, which provides useful insights into the field of combining cognitive psychology and Doppler effect with human behavior related insecurity. On the other hand, our proposed two-factor authentication system eliminates the user-phone interaction in most cases and can effectively defend against the existing attacks to recent 2FA mechanisms.

Table of Contents

1 Introduction **1**

1.1 Leveraging Keystroke Timings for PIN Cracking 2

1.2 Leveraging Ultrasound for PIN Cracking 3

1.3 Leveraging Keystroke Timings for Authentication 4

1.4 Contributions and Organization 5

2 Literature Review **7**

2.1 Side-Channel Attacks 7

2.1.1 Audio-based Attacks 7

2.1.2 Video-based Attacks 9

2.1.3 Sensor-based Attacks 9

2.2 Two-Factor Authentication Mechanisms 10

3 User-Independent Inter-Keystroke Timing Attacks **14**

3.1 Introduction 14

3.2 Preliminaries 19

3.2.1 Keystroke Timing Collection 19

3.2.2 Human Cognitive Models 21

3.2.3 Adversary Model 25

3.3 Attack Methodology 28

3.3.1 Learning Phase 28

3.3.2 Attacking Phase 29

3.4 Experiments . . . 30
3.4.1 User Study . . . 31
3.4.2 PIN Strength Level . . . 32
3.4.3 Performance Evaluation . . . 34
3.5 Other Specific Attacks . . . 36
3.5.1 Target Attacks . . . 37
3.5.2 Multi-Entry Attacks . . . 39
3.5.3 Known Digits Attacks . . . 40
3.6 Discussions . . . 43
3.6.1 Comparison with HMM-based Attacks . . . 43
3.6.2 Attack Threats to Real-World Applications . . . 44
3.6.3 Mitigations . . . 45
3.6.4 Limitations . . . 48

4 UltraPIN: Inferring PIN Entries via Ultrasound 49
4.1 Introduction . . . 49
4.2 Preliminaries . . . 54
4.2.1 Smartphones . . . 54
4.2.2 Doppler Effect . . . 55
4.2.3 Attack Model . . . 57
4.3 UltraPIN Design . . . 59
4.3.1 Overview . . . 59
4.3.2 Audio Capturing . . . 60
4.3.3 Keypair Segmentation . . . 61
4.3.4 Feature Extraction . . . 62
4.3.5 Movement Modeling . . . 66
4.3.6 Learning Models . . . 67
4.3.7 Keypair Combination . . . 72
4.4 Performance Evaluation . . . 74

4.4.1 Experimental Setting 74
4.4.2 Overall Effectiveness 77
4.4.3 Impact of Keypad Layout, Size, and Angle 78
4.4.4 Impact of Smartphone Quantity and Position 81
4.4.5 Impact of Smartphone-Keypad Distance 83
4.4.6 Impact of Experimental Environment 83
4.4.7 Runtime Performance of UltraPIN on Commodity Devices . 84
4.5 Discussions 85
4.5.1 Limitations 85
4.5.2 A Countermeasure 86

5 Typing-Proof: Usable, Secure and Low-Cost Two-Factor Authentication Based on Keystroke Timings 87

5.1 Introduction 87
5.2 Assumptions and Goals 90
5.3 Typing-Proof 92
5.3.1 Enrollment and Login 93
5.3.2 Similarity Score 94
5.3.3 Usability Analysis 96
5.3.4 Cost Analysis 98
5.4 Prototype Implementation 98
5.5 Evaluation 99
5.5.1 Data Collection 100
5.5.2 Parameters Configuration 101
5.5.3 False Rejection Rate 104
5.5.4 False Acceptance Rate 105
5.6 Security Analysis 106
5.7 User Study 110
5.7.1 Procedure 110

5.7.2 Usability 111
5.8 Discussion 113

6 Dissertation Conclusion and Future Work 116

6.1 Summary of Contribution 116
6.2 Future Research Plan 117

List of Figures

3.1 The layout of the numeric pad used in our experiments. 25

3.2 Overview of our inter-keystroke timing attacks. 25

3.3 The proportion of human chosen PINs at each strength level. 35

3.4 The averaged frequency of each PIN at different strength levels. . . 35

3.5 The performance of *general attacks*. 37

3.6 The performance of *targeted attacks*. 38

3.7 The performance of *multi-entry attacks*. 40

3.8 The performance of *known digits attacks*. 41

3.9 A new keypad layout. 46

4.1 Examples of keypads for PIN entries 50

4.2 Sensing finger movement for PIN entry 55

4.3 Time domain of a beep sound signal and key press timestamps . . . 61

4.4 Frequency spectrum of an ultrasound signal and its partition 61

4.5 (a) Spectrogram of a keypair segment. (b) After signal energy enhancement. (c) After signal carrier removal. (d) After frequency contour extraction. 63

4.6 ATM keypad and POS keypad 67

4.7 Learning models in UltraPIN 69

4.8 In-lab environment 75

4.9 Success rates of PIN attacks on various keypads 78

4.10 Success rates of PIN attacks at various keypad angles 80
4.11 Success rates of PIN attacks with one smartphone 81
4.12 Success rates of PIN attacks with two smartphones 81
4.13 Success rates of PIN attacks with three smartphones 82
4.14 Success rates of PIN attacks at various smartphone-keypad distances 83
4.15 On-site environment 84
4.16 Success rates of PIN attacks in on-site experiments 84

5.1 The overview of Typing-Proof two-factor authentication login procedure. 92
5.2 Example of an audio signal which is recorded in a café. 95
5.3 An illustration of the three different environments tested in our experiment: One-person Office, Research Lab, Starbucks café. 100
5.4 False rejection rate and false acceptance rate as a function of threshold τ_{sim}. 103
5.5 The relationship between ERR and the minimum length of random code. 104
5.6 The relationship between FAR and the minimum length of random code when FRR is fixed. 104
5.7 Impacts of different environments, phone positions, keyboard models to FRR and FAR in different configurations. 105
5.8 Answer to the post-test questionnaire. 113

List of Tables

3.1 A segment in the inter-keystroke timing dictionary used in our experiments. 29
3.2 List of PINs used in our experiments. 31

4.1 Composition of 41 classes of keypairs and their labels. 68
4.2 List of PINs used in our experiments. 77

5.1 The number of the login attempts per volunteer for each combination of settings. 101
5.2 FRR and FAR when usability and security have different weights. . 102
5.3 The items of the System Usability Scale [23]. All items are answered with a 5-point Likert-scale from *Strongly Disagree* to *Strongly Agree*. 111
5.4 The items of the post-test questionnaire. All items are answered with a 5-point Likert-scale from *Strongly Disagree* to *Strongly Agree*. 111
5.5 Comparison of Typing-Proof against Sound-Proof [67] and SMS-based 2FA [50] using the framework of Bonneau et al. [21]. We use 'Y' to denote that the benefit is provided and 'S' to denote that the benefit is somewhat provided. 115

List of Publications

Conference Papers

- **X. Liu**, Y. Li, and R. H. Deng. Typing-Proof: Usable, Secure and Low-Cost Two-Factor Authentication Based on Keystroke Timings. In *2018 Annual Computer Security Applications Conference (ACSAC 2018)*.

Journal Papers

- **X. Liu**, Y. Li, R. H. Deng, B. Chang, S. Li. When Human Cognitive Modeling Meets PINs: User-independent Inter-keystroke Timing Attacks. *Computers & Security*, Volume 80, January 2019, Pages 90-107, 4(1), 2010, pp. 21-30.

Acknowledgments

I would like to express my sincere gratitude to my supervisors, Prof. Yingjiu Li and Prof. Robert H. Deng for their continuous support and encouragement during my PhD study and research. Besides my supervisors, I would like to thank the rest of my dissertation committee members, Prof. Debin Gao and Dr. Tieyan Li, for their patient guidance and insightful comments in completing my dissertation.

I also thank my friends Daoyuan Wu, Jiayun Xu, Daibin Wang, Bing Chang, Yunshi Lan, Jiaqi Hong, Xiongwei Wu, and my seniors Yan Li, Ke Xu for the research collaboration, their friendship and encouragement.

Finally, I would like to thank my parents, Jun Liu and Hong Mei, for their generous care and supporting me spiritually throughout my life.

Chapter 1

Introduction

Password is a word or a string of characters used for authenticating the user to prove the identity or authorization to access certain resources. Personal identification number (PIN) is a special form of the password, which only consists of numerical codes. With the advent of computers, username and password are commonly used during the login process to protect information security and property safety to the computer operating systems, mobile devices, automated teller machines (ATMs), point of sale (POS) terminals and electronic door locks. Even with the great developments of new alternative authentication technologies, such as face authentication, fingerprint authentication, voice authentication, password still dominates the authentication field due to its simplicity and convenience.

However, password-based user authentication systems are often exposed to side-channel attacks in which an attacker can observe the password entry process via one or more side channels to infer keystrokes pressed by the target victim. Such side channels can be timing-related information [119, 151, 71, 79], acoustic information [9, 18, 155, 131], visual information [150, 115], and sensor-based information [31, 32, 83]. Despite the risk against different side-channel attacks, many users do not secure their PIN entries in any way (such as shielding the PIN entry keypad or checking surrounding) according to a field study of real-world ATM use [30]. Most side-channel attacks can be used to infer passwords and private information

being typed, and given the low entropy of PINs, the risk can be much higher.

On the other hand, side-channel information can also protect users' accounts if properly used. In a two-factor authentication system, side-channel information, such as background sound [67], keystroke timings [78], can be leveraged to verify the proximity between a user's registered phone and computer, and then to prove that the user has the possession of the second factor (i.e., the registered phone). This approach can effectively eliminate user-phone interactions and lower the cost of service provider compared to existing 2FA solution.

This dissertation investigates how to make use of keystroke information for both attacks and defenses. We first propose an inter-keystroke timing attack to PIN entry leveraging human cognitive model, then propose an acoustic attack to PIN entry via ultrasound, and last propose a usable, secure, and low-cost two-factor authentication system based on keystroke timings. The details of these works are introduced as follows.

1.1 Leveraging Keystroke Timings for PIN Cracking

Timing attacks have been widely studied to reveal useful information about encryption keys of ciphers and other private information such as information leaked via user interactions with web browsers. Some researchers have also studied how to infer PINs and passwords using inter-keystroke timing information in side-channel attack settings. Existing methods on inter-keystroke timing attacks are mostly based on the Hidden Markov Models (HMM) which needs to be trained based on a large number of observations on a victim's typing behavior, thus limiting their applications in real-world scenarios.

To solve the problem brought by the above limitations, the first work in this dissertation proposes a practical and user-independent inter-keystroke timing attack based on a parameterized human cognitive model. The parameters of our model can be estimated from a small amount of training data about *any users'* inter-keystroke

timing information. The training data in our attacks may not be taken from the victim, but from *other users* such as the attacker himself or people he/she recruits. Once the human cognitive model is built, it can be applied to compromise *any victims* inputting any PIN on a particular keypad whose geometric measurement is known.

According to our experimental results, the success rate of our attacks is significantly higher than random guessing attacks, which poses a serious threat when applied to users in a large scale, even in online attack settings. We evaluate the performance of our attacks with different PIN strength levels under different attack scenarios. We further provide several solutions to mitigate our attacks.

1.2 Leveraging Ultrasound for PIN Cracking

Personal identification number (PIN) has been widely used for user authentication to protect user privacy and property. Existing PIN attacks, including shoulder surfing, hidden camera spying and social engineering based guessing, are difficult to launch in a large scale. Therefore, it is still considered to be secure enough while it is widely used on most ATMs, POS terminals, electronic door locks, and personal device logins. The second work in this dissertation proposes UltraPIN, a novel and practical attack to recover PIN with commodity smartphones by analyzing imperceptible acoustic signals issued by smartphones and reflected by finger movements during PIN entries.

Motivated by an observation that finger movements on a keypad can be captured by analyzing surrounding acoustic signals, UltraPIN leverages the speakers and microphones that are already embedded in commodity smartphones to play an inaudible ultrasound and record the reflected acoustic signal. In particular, it segments keypairs based on beep sounds, extracts frequency-shifted pattern based on centroid frequency, constructs learning models to classify different finger movements, and recovers the PIN sequence by a backward inference algorithm.

We conduct a series of experiments under different attacking scenarios to evaluate the performance and robustness of our model. The results show that UltraPIN can effectively recover 75% PIN sequences within three attempts in the default case. In addition, UltraPIN is robust with regard to different keypad layout, size and angle, smartphone quantity and position, smartphone-keypad distance, experimental environment.

1.3 Leveraging Keystroke Timings for Authentication

Two-factor authentication (2FA) systems provide another layer of protection to users' accounts beyond the password. Despite the security improvement introduced by 2FA, most users still prefer password-only authentication where 2FA is not mandatory. This is probably due to the extra burden that 2FA causes to users since it typically requires users to interact with hardware tokens or software tokens on their phones. Reducing extra burden in 2FA triggers increasing interests in recent years. A recent approach, *Sound-Proof*, requires zero interaction between user and phone but has been proved insecure against certain attacks. The third work in this dissertation designs a new two-factor authentication system called Typing-Proof to solve the above problems.

In Typing-Proof, the second factor is the proximity of a user's phone to the computer being used to log in to an authentication server. The proximity of a user's phone to the login computer is verified by comparing keystroke timing sequence recorded by the login computer with the keystroke sounds recorded by the user's registered phone. We provide one-button authentication as a backup solution to avoid the false rejection cases of the automatic matching. Our proposed 2FA system is secure against the existing attacks to recent 2FA mechanisms. In addition, it enables significant cost savings for both service provider and user compared to

existing approaches, including Sound-Proof, SMS-based 2FA and hardware token based 2FA.

We implement a prototype of Typing-Proof and empirical experiments demonstrate that Typing-Proof achieves relatively low false acceptance in various settings with recommended configurations. We design several experiments to examine that Typing-Proof can effectively defend against known typed-text attacks and co-located attacks. We further conduct a user study to show that Typing-Proof is more usable than Sound-Proof and SMS-based 2FA.

1.4 Contributions and Organization

To summarize, the following contributions have been made in this dissertation:

- For the first time in the field, we applied human cognitive modeling to design a timing attack on PINs. We proposed the use of a human cognitive model with a small number of parameters to replace the HMMs used in previous work, thus allowing no or less training data and training based on other users. We discovered that the effectiveness of our attacks is different for different types of PINs and study the inner structure of the whole PIN space. We considered four different variants of the attack which are applicable to different attacking scenarios and have different performances. We conducted experiments involving a user study with human participants to prove the effectiveness of all four attack variants.

- We uncovered a new vulnerability of numeric keypad by leveraging speakers and microphones of smartphones. This is the first work to leverage speakers and microphones to recover the victim's PIN entry on a numeric keypad without compromising victim's device, where the sounds of striking keys can hardly be captured. We exploited the Doppler effect to sense the finger movements by leveraging inaudible ultrasound. We conducted a series of experiments to examine the performance of UltraPIN and the results demonstrated that an attacker can

successfully recover a victim's PIN typed on a numeric keypad with a probability of 75% within three attempts when he/she place two smartphones at a distance of 75cm from the keypad. Our attack is robust to the change of keypad layout, size and angle, smartphone quantity and position, smartphone-keypad distance, experimental environment.

- We proposed Typing-Proof, a usable, secure and low-cost two-factor authentication system. It requires no user-phone interactions in most cases and one-button press in the backup case. Typing-Proof is practically secure and incurs significant lower costs compared to other solutions. We implemented a prototype of Typing-Proof for Android devices and used the prototype to evaluate the effectiveness of Typing-Proof in a number of different settings. We showed that Typing-Proof works in any environment is compatible with all major browsers, login computers, and smartphones, and does not require any additional plug-ins or external hardware to be used. We conducted a user study to compare the perceived usability of Typing-Proof, Sound-Proof, SMS-based 2FA. Participants ranked the usability of Typing-Proof higher than Sound-Proof and SMS-based 2FA.

The remainder of this dissertation is organized as follows: Chapter 2 is a literature review that examines closely related research on side-channel attacks for keystroke inference and two-factor authentication mechanisms. Chapter 3 proposes an inter-keystroke timing attacks on PINs. Chapter 4 proposes an accurate and practical attack via ultrasound. Chapter 5 provides a usable, secure and low-cost two-factor authentication system leveraging on keystroke timings and typing sounds. Finally, Chapter 6 summarizes the contributions of this dissertation and points the future direction.

Chapter 2

Literature Review

2.1 Side-Channel Attacks

With the computer and network incorporated our life, password leakage events happened all the time. Many side-channel attacks for keystroke inference have been proposed in the literature over a decade. We summarize three types of side-channel attacks, including audio-based attacks, video-based attacks, and sensor-based attacks.

2.1.1 Audio-based Attacks

Acoustic Signature Attacks. Asonov et al. [9] proposed the first keystroke inference attack based on acoustic signatures. Based on the observation that the sounds of keystrokes vary from key to key, a supervised learning algorithm was designed to classify keystroke entries according to their sounds. Zhuang et al. [155] revisited keyboard acoustic emanations, and showed that a more advanced unsupervised algorithm with cepstrum features achieved better performance in acoustic signature attacks. Later, a training-free method was proposed by Berger et al. [18] to make such attacks more practical. Beyond the observation that hitting different keys leads to different keystroke sounds, they discovered that the similarity of two keystroke acoustic signals has a negative relationship with the distance between the pressed

keys.

Those acoustic signature attacks share an essential assumption that an attacker can capture clear keystroke sounds. However, the keypads on ATMs, POS terminals, and electronic door locks are usually designed to be soundless or sound-light. An attacker can hardly capture the sounds of striking keys on such keyboards. Although many keypads produce a clear beep in response to a keypress action, such beep sounds cannot be used to recover the keystrokes because they are released via the same sound speaker on a keypad. No matter which keys are pressed, the acoustic signatures of the beep sounds are indistinguishable so that they can hardly be used to infer keystrokes.

Timing Difference of Arrival (TDoA) Attacks. Zhu et al. [131] proposed a context-free method and made use of multiple microphones to estimate keystrokes' physical positions based on the Time Difference of Arrival (TDoA) of keystroke sounds on standard physical keyboards. When a user presses a key on a keyboard, its keystroke sound arrives at a pair of microphones placed besides the keyboard at different time. Therefore, a distance range from the key to the microphones can be measured such that a candidate set of key positions can be estimated and narrowed using multiple pairs of microphones. Liu et al. [77] combined the TDoA feature with acoustic signature feature into a training-free and context-free method so as to improve its performance.

Similar to acoustic signature attacks, the TDoA attacks cannot be applied to infer PIN entries on ATMs, POS terminals or electronic door locks. Striking keys on those keypads are mostly soundless while beep sounds are produced clearly. Since the beep sounds are produced from the same location (i.e., a sound speaker), TDoA attacks can only estimate the position of the sound speaker rather than any keys.

Finger Tracking Attack. Zhou et al. [154] proposed an acoustic attack that cracks pattern lock by analyzing the acoustic signals reflected by a fingertip that draws a pattern on a touchscreen. The assumption is that an attacker has installed certain

malware on the victim's smartphone and acquired necessary permissions to access speakers, microphones, and motion sensors on the victim's smartphone. Then the attacker traces finger movements based on a triangle relationship among a finger, a smartphone speaker and a smartphone microphone where the ratio between the longest side and shortest side in the triangle is around 2:1. In addition, they require an attacker to access the speaker and microphone of the victim's smartphone.

2.1.2 Video-based Attacks

Early works [14, 13] exploited the reflections of screens on glasses, spoons, eyes of users to recover the users' inputs. These attacks require attackers to acquire videos directly capturing users' screens or screen reflections. Recent works showed that even when keyboards or screens are not visible from the videos, attackers can still infer users' inputs via analyzing users' fingers or hand movements using advanced computer vision algorithms [150, 149, 115]. Even users' hands movements are not visible from the videos, Sun et al. [126] analyzed the motion patterns of devices backsides caused by the users' keystrokes on different positions of the screens of the users' devices and classify them using Support Vector Machine. All these attacks require attackers to place cameras at proper angles near the users.

2.1.3 Sensor-based Attacks

A range of studies showed that embedded sensors on mobile devices or wearable devices can reveal sensitive information about users' keystroke behaviors. Various embedded sensors were investigated in this context, including accelerometers [83, 10, 80, 81, 96, 137], gyroscopes [31, 24, 89, 146], ambient-light sensors [120] and WiFi [3, 73]. All these attacks require attackers to hack into mobile devices or wearable devices for accessing sensor data or to place mobile devices near users' keyboards.

2.2 Two-Factor Authentication Mechanisms

On the other hand, researchers also dedicate to providing more secure but still usable user authentication schemes. Two-factor authentication is one of the most effective approaches which has been widely used in our daily life. We review two traditional 2FA mechanisms, including hardware token and SMS-based software token, as well as several recent 2FA proposals which incur less user-phone interactions.

Hardware token based 2FA is a widely deployed 2FA solution in practice (e.g., in financial industry). It requires users to carry and use hardware tokens for authentication. During an authentication session, a hardware token is used to generate an authentication code at fixed time intervals (usually 60 seconds) according to a built-in clock and a factory-encoded random key (known as "seed"). A user reads the authentication code from the hardware token and inputs it to a login computer after the user inputs the first factor.

Hardware token based 2FA requires users to interact with their hardware tokens, read and remember authentication codes temporarily before input them on login computers. It also requires a service provider to manufacture a number of hardware tokens and distribute them to all customers. The cost of tokens is considerably high (e.g., $60 per token [2]), which is usually bore by service providers. In addition, a hardware token usually has a limited lifetime of around 3 years, which implies that service providers should distribute new tokens to each customer every 3 years.

Due to the pervasive use of phones, SMS-based software token is becoming more popular in recent years. After a user inputs the first factor on a login computer which sends it to the corresponding server, the server sends a verification code to the user's registered phone via SMS. The user reads the verification code from the registered phone and inputs this code to the login computer to complete an authentication session. This solution does not require any additional hardware but it still requires the user to interact with his/her phone, temporarily remember a verification code, and manually inputs the code on the login computer. In this solution,

the service provider bears the cost for sending verification codes via SMS to users' phones.

Sound-Proof is a recent 2FA solution proposed to eliminate user-phone interactions and lower the cost [67]. After a user inputs the first factor, both login computer and registered phone begin to record background sounds simultaneously; then, the login computer sends the recorded audio data to the registered phone via server; a Sound-Proof application installed in the registered phone compares whether the two pieces of background sounds are similar, and determines if the login computer and the phone are located in the same environment, and thus decides whether the login attempt is legitimate or fraudulent.

Sound-Proof has a limitation that it rejects the login attempt if the average power of any recorded audio sample is below certain threshold in order to prevent an impersonation attack in the case that a victim's environment is quiet (e.g., while the victim is sleeping). This lowers its usability since it is common for a user to login to his/her accounts in a quiet place (e.g., home, office, and library). The sound introduced by user's typing would not make Sound-Proof work since the average power of keystroke sound is around 30dB as we measured while the threshold for sound recording is set to 40dB in Sound-Proof [67]. Sound-Proof suggests users make certain noise (by, e.g., clearing throat, knocking on the table) in quiet environments; however, it may be awkward for some users. It cannot work either if the login computer is not equipped with a built-in microphone since it cannot record the background sounds. We notice that most desktops are not equipped with microphones. In such cases, Sound-Proof demands additional hardware (i.e., external microphone) which may not be always convenient.

From a security point of view, Sound-Proof is vulnerable to certain practical attacks. Zhang et al. [153] proposed a *sound-danger attack* where an attacker may deliberately make a victim's registered phone to produce previously known sounds (e.g., making a phone call or VoIP call, sending an SMS, and triggering an app-based notification) remotely at the time of an attack. Therefore, the attacker can

make the same ringtone on his/her side at the same time to bypass the second-factor authentication since both ambient sounds of the victim and of the attacker are the same ringtone in such case. Another potential attack is *co-located attack* [67] where an attacker and a victim stay in the same environment (e.g., in the same café). The ambient sounds of the victim and of the attacker are obviously the same so that the attacker can bypass the second-factor authentication.

One-button authentication requires a user to install an application on user's smartphone and bind the application to the user's account. Whenever a login attempt occurs on a user's account, the user is notified via the application and prompted to approve or reject the request. For certain one-button authentication applications, users can approve login requests with notifications without even opening the applications. This solution makes two-factor authentication more user-friendly than SMS-based 2FA. It has been adopted by several enterprises, including Microsoft Authenticator [88], Blizzard Entertainment [20], Duo Security [36], LastPass [102], and Futurea [44].

However, most one-button authentication systems are not secure against *synchronized login attack*. If an attacker and a victim login to the victim's account at the same time, the victim cannot distinguish which login request sent to his/her registered phone is legitimate, and he/she may mis-approve the login request sent from the attacker. Although some one-button authentication applications display IP addresses of login computers along with authentication requests, it is still difficult for the users who have no knowledge about the IP addresses to distinguish which login request is legitimate. Furthermore, an attacker may forge an IP address if he/she knows the victim's IP address.

Short-range communications, such as Bluetooth, WiFi, or NFC, are also widely adopted to support two-factor authentication. An authentication service provider – SAASPASS [108] leverages on location-based iBeacon Bluetooth Low Energy (BLE) technology to authenticate users via Bluetooth communications between their registered phones and nearby login computers. Similarly, another 2FA pro-

posal, PhoneAuth [29], sets up unpaired Bluetooth communications between a login computer and user's phone via Bluetooth using a new challenge-response protocol. However, these solutions may not be always applicable since most browsers (e.g., Firefox, Internet Explorer, and Safari [86]) do not support Bluetooth APIs. In addition, these solutions are not secure if adversaries set up Bluetooth connections to victims' phones to bypass 2FA.

Instead of using Bluetooth, Shirvanian et al. [114] proposed using WiFi communications between login computer and user's phone for 2FA. However, this solution works only when both devices are connected to the same network.

As NFC is widely embedded into today's commodity smartphones, Facebook [38] introduced a physical NFC security key that allows users to login to their accounts on their smartphones via NFC. This solution makes hardware token based two-factor authentication process faster. Instead of reading an authentication code from a hardware token and inputting it to a login computer, a user just taps a NFC security key against his/her smartphone so as to complete an authentication session. However, this solution requires additional hardware and its cost is of similar concern as in the case of hardware token based 2FA.

Chapter 3

User-Independent Inter-Keystroke Timing Attacks

3.1 Introduction

This chapter investigates the inter-keystroke timing information disclosure threat against PIN-based authentication systems. Inter-keystroke timing attacks, which make use of the leaked keystroke timing information to infer a user's PIN, pose a serious threat to real-world applications, especially for online financial services whose authentication systems are based on PINs. Such attacks have triggered increasing interests in recent years due to the development of many practical approaches to obtaining users' keystroke timing information via different side channels, including CPU cache [57, 93, 99, 56, 76], shared event loops [133], I/O interrupts [33, 75, 151], and SSH [119]. Some approaches do not even require attackers to be physically close to victims or install malware on victims' devices, which significantly lower the barrier for launching inter-keystroke timing attacks in real-world scenarios.

Most of the existing inter-keystroke timing attacks on PINs or passwords are user-dependent. Since the seminal work published by Dawn Song et al. in 2001 [119], the Hidden Markov Model (HMM) has been exploited as a major tech-

nique to launching the inter-keystroke timing attacks [151, 71]. However, HMM is user-specific in a sense that it relies on the distribution of inter-keystroke times of a specific user typing each possible key pair (which represents a hidden state in HMM) so as to infer the user's PIN from the user's inter-keystroke timing information about a PIN entry. In other words, HMM requires that a sufficiently large amount of time intervals for each possible key pair that can be part of any PIN be typed by a specific user for model training so as to make the attacks to that specific user's PIN entry accurate and useful. It is usually difficult for an attacker to collect such large amount of inter-keystroke data about a victim before launching an effective attack. Even if it is possible, such attacks are not scalable. If an attacker intends to compromise a new victim, he/she needs to collect the new victim's inter-keystroke timing data about all possible key pairs and retrain his/her HMM for the new victim. In addition, the success rate of such attacks is too low to be practical in online attack settings since the number of guesses that is allowed to launch an online attack is usually restricted to small numbers (e.g., 3, 10, 100) in common practice.

In this chapter, we propose a user-independent approach to exploit inter-keystroke timing information for PIN inference, which makes inter-keystroke timing attacks much more scalable and practical. The model in our attacks is not user specific, which can be trained from a small amount of training data (e.g., a few key pairs instead of all possible key pairs) about *any* users (e.g., attackers or people recruited by attackers) instead of the target victims. In addition, our approach can be applied to attack *any* new victim without retraining the model. The success rate of our attacks is significantly higher than random guessing attacks, which poses a serious threat when applied to users in a large scale, even in online attack settings.

Our proposed approach leverages a human cognitive model to capture the common characteristics across *all* skilled users typing PINs. A skilled user can smoothly type a PIN without any typing error and without considerable recall time between consecutive keystrokes. The human cognitive model is derived from several PIN

typing behavioral phenomena which we summarize from the cognitive psychology literature. These PIN typing behavioral phenomena are universal to *all* skilled users. The parameters of our cognitive model can be estimated by a few key pairs from any user such as the attacker himself. Once the cognitive model is built, it can be used to attack any user inputting any PIN on a particular keypad whose geometric measurement is known.

At a high level, our attacks proceed as follows. First, an attacker builds a timing dictionary including all possible PINs and their corresponding timing sequences. The timing sequence of each PIN is derived from our cognitive model. Second, the attacker obtains the timing sequence of a victim's PIN entry via various side-channels (e.g., CPU cache, shared event loops, I/O interrupts, and SSH). Third, the attacker measures the cosine similarity between the observed inter-keystroke timing sequence and each entry in the timing dictionary and ranks all candidate PINs in the dictionary by their similarity values. Lastly, with a ranked list of candidate PINs, the attacker may launch online attacks using the PINs successively from the ranked list until he/she succeeds or the target account is locked (or the attacker aborts before the account is locked).

Besides the cognitive model that captures the common characteristics across all users typing PINs, another contributing factor to the user independence of our approach is the way an attacker measures the differences between a victim's PIN entry and each time sequence in the timing dictionary. We adopt cosine similarity since it is invariant to scaling of input vector. It can thus mitigate the negative impact of different typing speeds by different users.

We discover that the effectiveness of our attacks is different for different types of PINs. To examine the effectiveness of our attacks to different types of PINs, we study the inner structure of the whole PIN space and partition the PIN space into different strength levels. In particular, the 6-digit PIN space is partitioned into 5 PIN strength levels according to the directional density of the inter-keystroke timing sequences in the timing dictionary, where *level 1* is the weakest and *level 5* is the

strongest. Our attacks achieve much better performance on the PINs at the first four levels compared to the strongest level (i.e., *level 5*). For example, the attacks with 100 guesses on the PINs at *levels 1, 2, 3, 4* are 869, 221, 250 and 42 times more effective than on the PINs at *level 5*, respectively. The results suggest that users should choose their PINs at the strongest strength level for better security in the presence of inter-keystroke timing attacks.

We seek various ways to improve the success rate of our attacks. One question is whether we can achieve better performance using target victims' data for model training, which is commonly used in the existing inter-keystroke timing attacks. In this case, we train the cognitive model using the victim's own inter-keystroke timing data and launch our attacks to this victim's PIN entry. However, the results show that this way improves the success rate by about 4% only. Another question is whether an attacker can improve his/her success rate if he/she observes the victim's PIN entry for multiple times. In this case, the attacker can attack based on the average of the inter-keystroke timing sequences for multiple PIN entries from the same victim. It achieves around 2% performance improvement which is not significant either. Our study in these two cases shows that our approach is user-independent and it does not improve much using user-dependent data.

We further examine the scenario in which an attacker happens to know the values of certain digits of the target PIN before launching inter-keystroke timing attacks. It is reasonable to assume that an attacker may attain such knowledge about PIN digits due to the existence of many side-channel attacks (e.g., [32, 150, 115, 80, 135, 126]) and shoulder surfing attacks [127] to the PIN entry. Unsurprisingly, the success rate of our attacks is significantly improved due to the shrink of timing dictionary in our attacks. For example, when the attacker knows 2 digits, the success rate of attacking the PINs in *level 1* within 3 attempts is improved to 34.9% so that one out of every three target users can be successfully compromised. In this case, our attacks are practical in online attack settings when applied to a single user or a small number of users.

In general, the success rate of the proposed attacks may not be sufficiently high to pose imminent danger to an individual user's PIN. Our attacks are still practical because they are user-independent and can be applied to attack any number of users in a large scale. To show this, we study two cases in practical settings, where PINs are used as the only credential to protect users' accounts and where attackers can collect many users' inter-keystroke timing data for PIN entries using malicious JavaScripts.

To mitigate our attacks, we provide several solutions, including choosing longer PINs, choosing PINs at the strongest strength level, proposing a new keypad layout design, and implementing leakage resilient password systems (LRPSes). For the first countermeasure, the security strength of most existing PIN systems is determined by the success probability of random guessing attacks [147]. However, the security strength of PIN systems would be lowered significantly in the presence of our inter-keystroke timing attacks. We suggest users to choose 10-digit PINs to maintain the same security strength under our attacks as that of the 6-digit PINs under random guessing attacks. This solution does not require any change to the hardware of current PIN systems, though it requires users to remember longer PINs.

To relax the requirement on PIN length, our second suggestion is that users should choose PINs at the strongest strength level (i.e., *level 5* for 6-digits PINs[1]). Our study on 6-digit PINs shows that the success rate of attacking PINs at *level 5* is around 10 times higher than random guessing attacks. Therefore, to achieve the similar security strength of 6-digit PIN under random guessing attacks, we suggest users choose 7-digit PINs at the strongest strength level.

For the third countermeasure, if changes can be made to the keypad layout, we propose a novel keypad design secure against our proposed attack, which is also easy to use. Our new design nullifies all inter-keystroke timing attacks, which means

[1] *Level 5* includes 900,000 PINs which account for 90% of the total 6-digit PINs. It is thus relatively easy for a user to obtain a PIN at *level 5* if he/she simply chooses his/her PIN randomly.

the success rate of our attacks would be similar to that of random guessing attacks. Therefore, users can still use 6-digit PINs as before. For the last countermeasure, LRPSes have been well studied in the past two decades. A recent work [147] shows that in order to achieve the same security strength of current 6-digit PIN systems, LRPSes require hundreds of seconds to complete an authentication session [61, 7], which sacrifices their usability.

3.2 Preliminaries

In this section, we provide the basics about how to collect inter-keystroke timing information from users, how to model users' typing behavior and what our adversary model is.

3.2.1 Keystroke Timing Collection

To launch any inter-keystroke timing attacks, an attacker needs to collect inter-keystroke timing information about users' inputs. Several practical approaches have been proposed in recent years on how to collect inter-keystroke timing information through various leakage channels, including CPU cache [57, 93, 99, 56, 76], shared event loops [133], I/O interrupts [33, 75, 151], and SSH [119].

The first leakage channel through which attackers can collect inter-keystroke timing information is CPU cache [57, 93, 99, 56, 76]. Through CPU cache, an attacker can observe the effects of a user's keystroke operations and deduce the timestamp of each keystroke the user performs on a keyboard. One of these approaches [93] can be performed from browser sandboxes through remote websites using JavaScripts instead of installing malware on users' devices. Besides users' keystroke operations originated from hardware keyboard, other interactive operations, such as tap operations and swipe operations which are usually triggered on a touch screen, can also be monitored by an attacker [76]. Therefore, inter-keystroke timing attacks (including ours) can be applied to both devices with hardware key-

board and devices with soft keyboard. This keystroke timing collection approach requires malware installed on victim's device to access CPU cache but it does not need any permission.

The second leakage channel through which attackers can collect inter-keystroke timing information is shared event loops [133]. Through shared event loops in Google Chrome, an attacker can scan an event-delay trace using JavaScript and deduce the timestamp of each keystroke the user performs on a keyboard. This keystroke timing collection approach requires an attacker to trick victims to open a malicious website which has the permission of running JavaScript.

The third leakage channel through which attackers can collect inter-keystroke timing information is I/O interrupts [33, 75, 151]. An attacker may continuously acquire timestamps using JavaScript in a measuring process and monitor differences between subsequent timestamps [75]. Significant time differences will occur whenever the measuring process is interrupted by I/O operations (i.e., keystroke operations). The exact timestamp where the user presses a key is clearly visible and can be distinguished from other events. This keystroke timing collection approach also requires an attacker to trick victims to open a malicious website which has the permission of running JavaScript.

The last leakage channel through which attackers can collect inter-keystroke timing information is SSH [119]. Since every individual keystroke typed by a user is sent to a remote machine in a separate IP packet immediately after the key is pressed, precise inter-keystroke timings of the user's keystrokes can be learned from the arrival times of the packets. This keystroke timing collection approach requires an attacker to monitor the network and collect the arrival time of SSH packets which does not require any malware to be installed on victim's device or any permission from the victim.

The sampling rates of inter-keystroke timing information collected by different approaches are different (e.g., 40,000Hz for shared event loops and 100Hz for SSH). In our experiments, we use JavaScript to record the key code of each

keystroke event and the corresponding timestamp to get the ground truth. We observe that the timings of key-press events are distributed in clusters with a gap of 15 or 16 milliseconds; thus, the sampling rate in our experiments is no higher than $1000/15 \approx 66.7$Hz. Although our sampling rate is relatively low, our attacks still achieve satisfactory performance as shown in our experimental results (Section 3.4.3). The performance of our attacks may be improved further at higher sampling rates.

To determine the start and the end of victim's PIN entry, the attacker can monitor all the packets sent by the victim by a network sniffing tool on network packets such as Wireshark and records the timestamps of all packets whose destination IP is the targeted sensitive website (e.g., online banking website or Alipay) [73]. Since most of the important websites and applications are secured via HTTPS, it does not protect the meta data of the traffic such as destination server's IP address, which can be used to recognize the start of a time window for searching the victim's PIN entry using various approaches which have been mentioned earlier in this section. If the victim is entering PIN on an Android application, Cheng et al. [28] proposed a no-root approach to detect login activities as they share a common pattern that a login activity usually consists of two EditText fields for inputting a username, and a password and the second EditText field sets the attribute inputType to password-related by developers. In addition, malware installed in the victims' phone may make use of accessibility feature to monitor the timing of any event that is activated by the victim by the id of the view [105]. Since most developers use EditText fields with an id of 'password' or 'PIN' in the layout view, it is easy for the attacker to know the start time of a victim's PIN entry event.

3.2.2 Human Cognitive Models

History. Human cognitive models have been studied in the field of psychology for decades. They describe one or more specific human cognitive processes (e.g.,

memory, perception, attention, reasoning, and problem-solving) for the purpose of better understanding, predicting and simulating human behavior [4].

Typing PINs on a numeric keypad is one of the most important human-computer interactions in our daily life and it involves complicated interactions of concurrent perceptual, cognitive, and motoric processes [144]. To model typing behaviors and explore its underlying mechanisms, cognitive psychologists apply the knowledge of psychology, human-computer interaction and neuroscience. Card et al. [25] propose a keystroke-level model (KLM) to predict the time of a user accomplishing a given task without errors using a given interactive computer system. For typing task, KLM gives a rough estimate of the average inter-keystroke time, which is calculated by dividing the total time taken in a typing test by the total number of non-error keystrokes. Rumelhart and Norman [106] build a model of typing and provide detailed predictions about the movement of fingers and the relative response time for letters in different contexts. Furthermore, John [65] proposes a typing performance theory which is built within the framework of the Model Human Processor (MHP) [26] and can offer a more precise estimation. These models of cognitive processing have provided a wealth of information regarding how humans interact with keyboards.

Cognitive psychologists and HCI researchers have also developed several software tools for estimating human performance in terms of time needed by an average skilled user to finish a specific task, such as Cogulator [37], CogTool [129], SANLab-CM [97]. Such tools are normally used for modeling and simulating complicated processes involving both computer and human users, but this work focuses on determining the parameters of a specific model of the typing behavior, so we do not use such tools in our work. In the following sections, we build a new keystroke model combining models mentioned above with empirical analysis.

Typing Behavior Phenomena. Typing is a complex procedure involving cognitive activities as well as body movements, but we can still capture the common characteristics across all skilled users' typing behaviors. The typing procedure usually

involves two parts: (1) *cognition* of the task and (2) *motor* of the task. During the cognition process, the user conducts a memory retrieval process. Specifically in our scenario, the user recalls his/her PIN from the long-term memory, stores it into the working memory and mentally prepares for executing physical actions. During the motoric process, the user moves hand and fingers to the right key, presses the key, releases the key and prepares for the next keystroke. The total time between two keystrokes is the sum of the time for these two parts.

PIN entry behavior is one of the most common typing behaviors in our daily life. In order to explore PIN entry behavior, we generalize four typing behavior phenomena. They are based on the literature (e.g., [110]) which discusses several common phenomena about typing behaviors across all skilled users.

Phenomenon 1. The rate of typing is dependent on how familiar the user is with the typed string. According to a statistics report, 46 percent of the U.S. students use credit cards on a regular basis for everyday purchases [123]. And the average iPhone user tends to unlock his/her device 80 times in a day [101] while Android users tend to unlock their smartphones an average 110 times a day [139], so there is no doubt that people are proficient in typing their PINs.

Phenomenon 2. The variability of inter-keystroke time decreases with an increase in users' skill level. The distributions of inter-keystroke time for the same keystroke in the same context but across multiple repetitions are similar [109]. This phenomenon indicates that the typing pattern will stabilize after several practices.

Phenomenon 3. Inter-keystroke time of typing decreases following the power law of practice. Typing speed of a user will be significantly improved as the number of inputs increases. According to the learning curve of the single user in the study of Gentner [48], the improvement of inter-keystroke time follows exponential growth. If a skilled user can input PINs smoothly enough, the time of cognition process may be negligible. One reason is that muscle memory has been built after frequently typing and it may take little time for the cognitive processor to make decisions and schedule actions with the motor processor.

Phenomenon 4. The inter-keystroke time is dependent on the specific context, especially for the topography of the keyboard. The specific context here refers to the character before and after the target character. This topographical effect has been reported by Gentner [47, 48], Rumelhart and Norman [106], and Shaffer [113]. Intuitively, the latency between two keystrokes has a positive correlation to their distance on the keypad.

Based on Phenomena 1 and 2, the action of entering a PIN can be regarded as conducted by a skilled user whose typing pattern is stable and predictable. Based on Phenomena 2 and 3, we arrange a practice session before data collection in our experiments in order to collect skilled users' data and simulate people entering PINs in real life. For Phenomenon 4, we estimate the topographical effect by a function of the finger's moving distance and the size of target key using Fitts's law [40].

Inter-Keystroke Timing Modeling. We incorporate the above typing behavior phenomena to construct a linear model for predicting the inter-keystroke timings of any key pair.

For the topographical effect, our model uses Fitts's law [40] to make finer predictions. Fitts's law is a descriptive model of human movement which can predict the time required to move to a target area. It is used to model the act of physically touching an object with a finger or virtually pointing to an object. Striking the numeric keypad with one finger can be seen as this kind of action. It is a function of the ratio between the distance to the target (D) and target width (W):

$$T = a + b * I = a + b * \log_2(\frac{D}{W} + 1), \tag{3.1}$$

where D is the distance from the start point to the center of the target, W is the effective width of the target in the direction of the motion[2], $I = \log_2(D/W + 1)$ is called the *index of difficulty*, a and b are parameters varying from context to context.

[2]According to our observations in the experiments, the effective press area of each key is close to a circle centered on the center of the key and with a radius equal to the shorter side of the key (which is 0.5 inches). Therefore, we use 0.5 inches as the effective width for all keys including 0 and <Enter> keys.

Figure 3.1: The layout of the numeric pad used in our experiments.

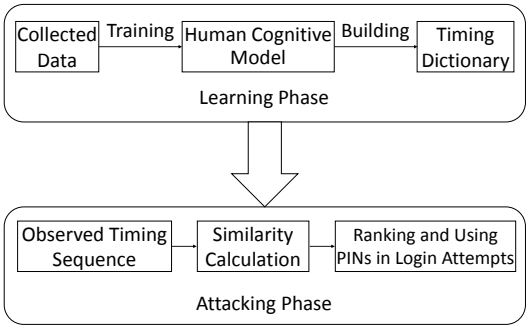


Figure 3.2: Overview of our inter-keystroke timing attacks.

We use the geometric center of each key to obtain the distance of each key pair. As for the repeated pressed key like '99', we set $I = 0$ so that $T_{motor} = a$. We estimate the values of a and b using inter-keystroke timing data of real human users. With these inter-keystroke timing data and the geometric measurement of victim's keyboard, an attacker can build his/her own inter-keystroke timing model.

3.2.3 Adversary Model

Basic Premises. It is usually difficult for a malware to directly record keystrokes due to the use of keylogger detection technologies [95, 125, 130, 94, 8]. The barrier for launching inter-keystroke timing attacks in real-world is much lower than directly recording keystrokes. Recent works (e.g., [57, 93, 99, 56, 76, 133, 33, 75, 151, 119]) have introduced many practical approaches to attaining user's keystroke timing information. While these works focused on how to capture keystroke timing information, our work focuses on how to make use of keystroke timing information to recover PINs. Therefore, our adversary model assumes that an attacker has already obtained the inter-keystroke timing information about a target user (victim) typing his/her PIN on a numeric keypad.

The inter-keystroke timing information about a PIN can be observed just once

or a number of times via several leakage channels such as CPU cache, shared event loops, I/O interrupts, SSH as introduced in Section 3.2.1. We notice that directly recording keystrokes requires certain permissions which are usually difficult to be gained (e.g., most software keyloggers require Windows hooks); in comparison, it is relatively easier for an attacker to obtain keystroke timing information. In particular, the shared event loop approach [133] and the I/O interrupts approach [75] require that victims' browsers support JavaScript, which is common for popular browsers in the default setting. In addition, the CPU cache approach [76] and the SSH approach [119] require no permission to obtain keystroke timing information.

It is also assumed that an attacker knows the layout of the keyboard (including the size of each key and the distance between each key pair) which the target victim uses in advance. This is a reasonable assumption since in most cases, the victim inputs his/her PINs on the number pads of ATMs, POS terminals, or standard keyboards. The layouts of these keypads are standardized or can be easily obtained in the public place. Figure 3.1 shows the layout of a DELL SK-8115 numeric keypad which is used in our experiments.

For the victim's PIN typing behavior, it is assumed that one finger is used to enter the whole PIN followed by an <Enter> key press to signal the end of a PIN entry process. It is also a reasonable assumption since according to our observation and the survey[3] we conducted during the experiments, a majority of users (63.2%) prefer using a single finger for PIN entry.

Online Attacks. In online attack settings, an attacker consecutively tries a number of candidate PINs to attack a PIN-protected account until the correct PIN is found or the account is locked (or the attacker aborts before the account is locked). The online attack that has been studied in most previous research on PIN systems [84] is random guessing attack in which an attacker inputs random PINs. In this work, we consider four other online attack settings by assuming that an attacker has different knowledge about a victim's typing behavior or the target PIN:

[3]Please refer to the Section 3.6.4 for the detailed statistical results.

(i) *General attacks*: An attacker collects a small amount of inter-keystroke timing data from the attacker himself or people he/she recruits for model training and obtains a single inter-keystroke timing sequence of a PIN entry made by a victim for PIN inference.

(ii) *Targeted attacks*: An attacker collects a small amount of inter-keystroke timing data about a victim typing known numerical sequences for model training and obtains a single inter-keystroke timing sequence of a PIN entry made by the victim for PIN inference.

(iii) *Multi-entry attacks*: An attacker collects a small amount of inter-keystroke timing data from the attacker himself or people he/she recruits for model training and captures several inter-keystroke timing sequences about a victim entering the same PIN. In this case, the attacker may combine all inter-keystroke timing sequences and obtain an averaged timing sequence for PIN inference.

(iv) *Known digits attacks*: An attacker knows certain digits of a target PIN before launching our *general attacks*. Such knowledge may be attained from various side-channel attacks [32, 150, 115, 80, 135, 126] or shoulder surfing attacks [127].

Limited number of login attempts. Most PIN systems enforce suspicious login detection and lockout [43], and thus the number of PINs an attacker may try in an online attack is limited. A successful online attack is defined as an attacker hitting the correct PIN within the number of allowed attempts. The number of login attempts is normally restricted to 3 for PINs with payment cards. When a payment card is used on a POS terminal or with a card reader, entering a PIN wrongly for 3 times may get the card locked. This limit is usually larger for mobile devices. For example, an Android device gets locked temporarily for 30 seconds after every 5 failed attempts, while an iOS device is restored to factory settings after 10 failed logins. Other cases limit online attackers to no more than 100 consecutive failed

attempts on a single account according to the digital authentication guidelines [55] and electronic authentication guidelines [54]. In our experiments, we demonstrate the success rates of our attacks with various limits on the number of consecutive login attempts.

Offline Attacks. In offline attack settings, it is assumed that an offline validation of guessed PINs can be performed. This is a less realistic scenario since users' PINs are usually stored in tamper-resistant hardware security modules on the server side as a common practice. Therefore, we focus on the online attacks in this work.

3.3 Attack Methodology

In this section, we describe the steps of our inter-keystroke timing attacks in detail. Figure 3.2 shows an overview of our inter-keystroke timing attacks, including a learning phase and an attacking phase. In the learning phase, an attacker trains a cognitive model based on certain collected data and builds a timing dictionary. In the attacking phase, the attacker (i) observes one or more entries from a victim, (ii) calculates the similarity between the timing sequence of the observed PIN entry and the calculated timing sequence of each entry in the timing dictionary, (iii) ranks all candidate PINs according to the similarity values, and (iv) attempts to login to the victim's account using the PINs in the ranked list starting from the top in an online attack.

3.3.1 Learning Phase

Data Collection. In the learning phase, an attacker needs to collect the inter-keystroke timing sequences for a small number of key pairs for model training. Since our cognitive model consists of two parameters, it requires that the training data consists of the inter-keystroke timing sequences for at least two key pairs (1,350 key pairs are used in our experiment). The training data used in the learning phase can be collected from the attacker himself or people he/she recruits. The simplest

Table 3.1: A segment in the inter-keystroke timing dictionary used in our experiments.

PINs	K_1-K_2	K_2-K_3	K_3-K_4	K_4-K_5	K_5-K_6	K_6-K_7
504316	232.9502	232.9502	237.2201	231.3787	237.2201	226.0874
504317	232.9502	232.9502	237.2201	231.3787	231.3787	268.5020
504318	232.9502	232.9502	237.2201	231.3787	237.2201	256.9941
504319	232.9502	232.9502	237.2201	231.3787	250.0087	247.2787
504320	232.9502	232.9502	237.2201	199.0121	203.7241	244.2814
504321	232.9502	232.9502	237.2201	199.0121	199.0121	254.0817
504322	232.9502	232.9502	237.2201	199.0121	135.9120	232.9502
504323	232.9502	232.9502	237.2201	199.0121	199.0121	203.7241
504324	232.9502	232.9502	237.2201	199.0121	214.2976	259.6575
504325	232.9502	232.9502	237.2201	199.0121	199.0121	243.2131

way to collect training data is to implement a keylogger which records the key code of every keystroke event and the corresponding timestamp to get the ground truth.

Cognitive Model Training. With the training data, the attacker can estimate the coefficients of the linear equation (Equation 3.1) in our cognitive model using the standard least squares method.

Timing Dictionary Building. Once the cognitive model is fixed, the attacker can compute the inter-keystroke timing sequences for *all* PINs and then generate a timing dictionary $D = \{(\text{PIN}_i, \overrightarrow{T_i})\}$ for $i = 1, 2, ..., 10^l$ where $\overrightarrow{T_i} = (\Delta T_{i1}, \Delta T_{i2}, ..., \Delta T_{il})$ and l is the PIN length. Here, ΔT_{ij} is computed according to the cognitive model for j-th key pair $(K_{ij}, K_{i(j+1)})$ in the i-th PIN $(K_{i1}, K_{i2}, ..., K_{il})$, where $j = 1, 2, ..., l$ and $K_{i(l+1)} =< Enter >$. Table 3.1 shows a segment in the inter-keystroke timing dictionary used in our experiments.

3.3.2 Attacking Phase

Data Collection. In the attacking phase, an attacker needs to obtain a single inter-keystroke timing sequence $\overrightarrow{T'}$ of a PIN entry made by a victim for PIN inference. Similar to each timing sequence in the timing dictionary, $\overrightarrow{T'}$ is an l-dimensional sequence, where l denotes the length of the target PIN.

Similarity Calculation. Once the attacker has an observed timing sequence $\overrightarrow{T'}$ of

the target PIN (from a victim) and a timing dictionary D, he/she can measure the similarity between $\overrightarrow{T}$ and each timing sequence in D.

There are many similarity metrics the attacker can use. We test three different metrics (cosine similarity, Euclidean distance and Pearson product-moment correlation coefficient) and discover that the cosine similarity gives the best results in most attacks. The cosine similarity is a measurement of the level of similarity between two vectors $\overrightarrow{A}$ and $\overrightarrow{B}$ that returns the cosine of the angle between them and is computed as follows:

$$cos = \frac{\overrightarrow{A} \cdot \overrightarrow{B}}{\left\|\overrightarrow{A}\right\| \cdot \left\|\overrightarrow{B}\right\|} = \frac{\sum_{i=1}^{l} a_i b_i}{\sqrt{\sum_{i=1}^{l} a_i^2} \cdot \sqrt{\sum_{i=1}^{l} b_i^2}}, \tag{3.2}$$

where a_i and b_i are the i-th elements of l-dimensional vectors $\overrightarrow{A}$ and $\overrightarrow{B}$, respectively. The time complexity for the similarity calculation is $O(n)$, where n is the number of all possible PINs. The cosine similarity is scale-free, i.e., the amplitudes of $\overrightarrow{A}$ and $\overrightarrow{B}$ have no impact to the result. This feature improves the robustness of our attacks against variation of typing speeds between victims and different users in the training data, which thus contributes to the user independence of our approach.

Ranking and using PINs in login attempts. The attacker then ranks all entries in the timing dictionary according to their similarity values so that those entries more similar to $\overrightarrow{T}$ appear closer to the top. Here, we use the Quicksort whose time complexity is $O(n*log(n))$ to rank all candidate PINs. Finally, the attacker attempts to login to the victim's account using the PINs starting from the top in the ranked list in an online attack.

3.4 Experiments

An IRB-approved user study is conducted to collect users' inter-keystroke timing data about PIN entries on a numeric keypad. The data collected from participants are kept confidential and anonymized. To examine the effectiveness of our attacks

Table 3.2: List of PINs used in our experiments.

Level 1	777777	777333	222233	633333	555553
	443333	088886	000553	055333	577773
Level 2	008853	166034	226633	515553	009666
	800053	705333	100086	222253	100553
Level 3	911182	590253	537473	086483	084953
	331086	410886	547733	537802	199993
Level 4	990872	098046	760973	301509	330117
	301246	095653	589107	530271	603294
Level 5	420381	191061	806205	079039	033645
	146928	501347	635210	684032	706759

to different types of PINs, we study the inner structure of the whole PIN space and partition the PIN space into different strength levels. In this section, we present the performance of our attacks in the general attack setting in which the training data and testing data are collected from different users.

3.4.1 User Study

Our user study involves 55 participants, including 24 males and 31 females with ages ranging from 19 to 34. All participants are students or members of staff at our university. Each participant is paid 10 dollars as a compensation for his/her time and effort. Since 6-digit PINs are commonly used in many PIN-based authentication systems, we use 6-digit PINs as examples of our attacks. Our user study consists of two sessions: training session and testing session. In both sessions, we use JavaScript to record the key code of each keystroke event and the corresponding timestamp to get the ground truth.

In the training session, 5 participants are asked to enter three 6-digit PINs (i.e., 146928, 501347, 635210) on a numeric keypad. The PINs they typed are randomly selected from the whole 6-digit PIN space. The participants are required to memorize one PIN intentionally, type the PIN for several times as exercises and type more times for data collection; then, they are required to forget the current PIN, and proceed in the experiment with the next PIN. In our experiments, we observe that

exercises for five times are sufficient for a participant to type a 6-digit PIN fluently. Then the participants type each PIN for 15 times continuously for training data collection. We ensure that each PIN entry is typed correctly. If a participant enters incorrect digits and uses the <Delete> or <Backspace> key to correct an input, he/she is required to retype the PIN.

In the testing session, we choose 50 PINs with 10 PINs randomly selected from each of five PIN strength levels as listed in Table 3.2. The other 50 participants (except the five in training to make our attacks user independent) are asked to enter PINs on the same numeric keypad. Each participant is assigned to type 25 PINs with 5 PINs chosen randomly from the 10 PINs in each PIN strength level. Similar to the training session, the participants type each PIN for 5 times as practice and type each PIN for 15 times for testing data collection. In total, 225 PIN entries are collected for training and 18,750 PIN entries for testing.

The raw data of each PIN entry we collected consists of the timestamps of (l+1) keystroke events for l-digit PINs, where the last keystroke is for pressing the <Enter> key. We define the inter-keystroke timing between keystrokes K_i and K_{i+1} as the difference between the two consecutive key-down times to cover both the time of finger movement between the two keys and the time for pressing the second key:

$$\Delta T_i = T^{\downarrow}_{K_{i+1}} - T^{\downarrow}_{K_i}. \tag{3.3}$$

Therefore, the inter-keystroke timing sequence of each PIN entry that is used in our experiment is represented by an l-dimensional sequence $\overrightarrow{T} = (\Delta T_1, \Delta T_2, ..., \Delta T_l)$.

3.4.2 PIN Strength Level

We study the inner structure of the whole PIN space to examine the effectiveness of our attacks to different PINs. We propose an approach to partition the whole PIN space into different PIN strength levels according to the directional density of the inter-keystroke timing sequences in the timing dictionary. Each inter-keystroke

timing sequence in the timing dictionary can be considered as an l-dimensional directional vector, where l is the PIN length. Intuitively, if a PIN vector locates in a dense region according to the cosine similarity measurement in the vector space, it is more difficult for an attacker to single it out, that is, infer the PIN. This implies that such a PIN is more secure against our attacks since our attacks rank candidate PINs according to the cosine similarity between each entry in the timing dictionary and the observed timing sequence of a target PIN as explained in Section 3.3.2. Based on this observation, we propose Algorithm 1 to measure the PIN strength of l-digit PINs.

Algorithm 1 : PIN Strength Measurement

Input: A trained timing dictionary $D = \{(\text{PIN}_i, \overrightarrow{T_i})\}$ for $i = 1, 2, ..., 10^l$ where $\overrightarrow{T_i} = (\Delta T_{i1}, \Delta T_{i2}, ..., \Delta T_{il})$.

Output: The strength measurement $\overrightarrow{S_i}$ for each PIN_i.

1: **for** each vector $\overrightarrow{T_i}$ in D **do**

2: calculate the cosine similarity between $\overrightarrow{T_i}$ and all other vectors in D and obtain a cosine similarity tuple $(cos_{i1}, cos_{i2}, .., cos_{i(i-1)}, cos_{i(i+1)}, .., cos_{i(10^l-1)})$ where $cos_{ij} = \frac{\overrightarrow{T_i} \cdot \overrightarrow{T_j}}{\|\overrightarrow{T_i}\| \cdot \|\overrightarrow{T_j}\|}$

3: rank all cosine similarities in descending order and obtain a new tuple $(cos'_{i1}, cos'_{i2}, ..., cos'_{i(10^l-1)})$ where $cos'_{i1} \geq cos'_{i2} \geq ... \geq cos'_{i(10^l-1)}$

4: $\overrightarrow{S_i} = (\overline{G}_1, \overline{G}_2, ..., \overline{G}_l)$ where $\overline{G}_j = \frac{1}{9*10^{j-1}} \sum_{10^{j-1} \leq n \leq 10^j - 1} cos'_n$ and $j = 1, 2, ..., l$.

5: **end for**

Algorithm 1 takes a trained timing dictionary D as input. For each timing vector $\overrightarrow{T_i}$ for PIN_i in D, the algorithm first calculates the cosine similarity between $\overrightarrow{T_i}$ and all other vectors in D. It then ranks all of the calculated cosine similarities in descending order and divide them into l groups where the j^{th} group consists of $(10^{j-1})^{th}$ to $(10^j - 1)^{th}$ cosine similarities. Finally, it calculates the average value $\overline{G}_j$ of cosine similarities for group j, where $j = 1, 2, ..., l$. The algorithm output an l-dimensional tuple $\overrightarrow{S_i} = (\overline{G}_1, \overline{G}_2, ..., \overline{G}_l)$ to represent the PIN strength for each PIN_i, where $i = 1, 2, ..., 10^l$. The overall time complexity of our PIN strength measurement algorithm is $O(l * 10^{2l})$ and its space complexity is $O(l * 10^l)$.

With the strength measurement $(\overline{G}_1, \overline{G}_2, ..., \overline{G}_l)$ for all PINs, we partition the

whole PIN space into (l-1) levels. First, an indirect stable sort with multiple keys is performed on all PINs. To be specific, it first ranks all PINs by key $\overline{G}_1$, if two PINs have the same value for key $\overline{G}_1$; then it ranks them by key $\overline{G}_2$; and so on. As a result, it ranks all 10^l PINs according to PIN strength in ascending order. The first 100 PINs after ranking are categorized into *level 1* which includes the weakest PINs. The 101^{th} to 1000^{th} PINs are categorized into *level 2*; the 1001^{th} to 10000^{th} PINs are categorized into *level 3*; and so on. In our experiments, we take 6-digit PINs as examples and divide all 6-digit PINs into 5 categories. *Level 1* to *level 5* consist of 100, 900, 9,000, 90,000, 900,000 PINs, respectively.

We further study the distribution of human-chosen 6-digit PINs according to our PIN space partition. The human-chosen 6-digit PINs are extracted from two leaked large-scale password databases (i.e., Rockyou and CSDN) [22, 136]. In total, we collect 2,353,101 leaked passwords and all of them are 6-digit PIN. Figure 3.3 shows the proportion of human-chosen PINs at each strength level and Figure 3.4 shows the averaged frequency of each PIN at different strength levels. The frequency is calculated by the ratio between the count of leaked passwords at each strength level and the number of PINs of the corresponding strength level. It is observed that although the PINs at the lowest security level (i.e., *level 1*) account for only 0.01% of the total, more than 2.7% of real users prefer to select PINs at this level and the averaged frequency of at this level is significantly higher than other strength levels. These results show that users tend to select weak PINs more often than strong PINs. It is thus meaningful to evaluate PIN attacks at different security levels.

3.4.3 Performance Evaluation

We evaluate the performance of our attacks in the general attack setting. First, we use the inter-keystroke timing data from the training session of our user study to train the cognitive model. The parameter a and b are 135.91 and 47.73, respectively.

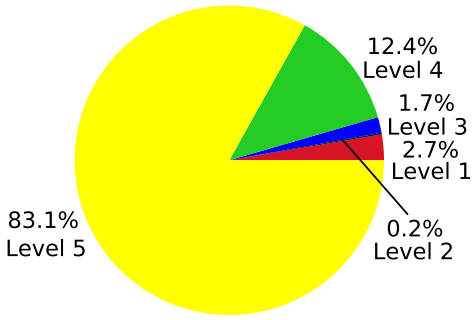


Figure 3.3: The proportion of human chosen PINs at each strength level.

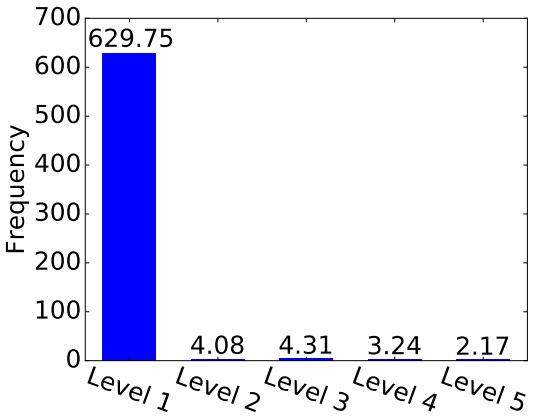


Figure 3.4: The averaged frequency of each PIN at different strength levels.

Based on this trained cognitive model, we estimate the timing sequence of all 10^6 6-digit PINs and generate a timing dictionary. According to our experiments, it takes 21.7s to generate a timing dictionary.

Then, we take each PIN entry typed by the participants from the testing session as an independent attacking case. In total, there are 18,750 individual cases for 50 PINs. Note that the training data and the testing data in our user study are collected from different groups of participants, which make our attacks user-independent. For each attacking case, we measure the cosine similarity between the observed timing sequence and each entry in the timing dictionary and rank all PINs according to their similarity values in the descending order. Given an observed timing sequence, it takes around 1s to get the ranking list of all candidate PINs in the general attack. If the correct PIN ranks x-th in the ranked list, an attacker needs to login to target victim's account for x times until success.

The performance of such *general attacks* is shown in Figure 3.5, where the x-axis denotes the position of a correct PIN in the ranked list and the y-axis denotes the success rate of hitting the correct PIN in an attack. The success rate of our attacks is calculated as the observed frequency that the correct PIN appears in the top x ranked PINs across all attacking cases. Note that the success rate of our attacks is 0 before any successful case is observed. Figure 3.5 also shows the success rate of random guessing attacks, assuming that the correct PIN has an equal probability to

appear at any position between 1 and 10^6. The success rate of random guessing attacks is $\binom{10^l-1}{x-1}/\binom{10^l}{x}$ for an l-digit PIN where x is the maximum number of allowed consecutive failures.

A general trend in Figure 3.5 is that it is more effective to attack PINs at lower strength levels. Beyond our expectation, the performance of PINs at *level 3* is better than *level 2* but the difference between them is not too significant. Maybe it is because that number of samples in each levels is small in our user study. This trend suggests that users should choose their PINs at the strongest strength level for better security in the presence of inter-keystroke timing attacks.

Another trend in Figure 3.5 is that the performance of *general attacks* is much better than random guessing attacks. In particular, if the number of allowed attempts is limited to 100, 10 and 3, our *general attacks* improve the success rate by 522, 2247 and 4004 times on average of all PIN strength levels over the random guessing attacks, respectively.

Our experimental results imply that the existing PIN-based authentication systems are vulnerable to our attacks, especially when they are launched at a large scale. When a victim types a PIN at *level 1*, an attacker can launch a successful attack within 10 attempts with a probability about 10%. It has been argued that if 10% of accounts in an authentication system are compromised, an attacker may access all resources of the system [41].

3.5 Other Specific Attacks

While the *general attacks* we discussed in the previous section are user-independent (i.e., the training data and the testing data are collected from different users), we examine other specific attacks to improve the success rate of *general attacks* with different assumptions on attackers' capabilities.

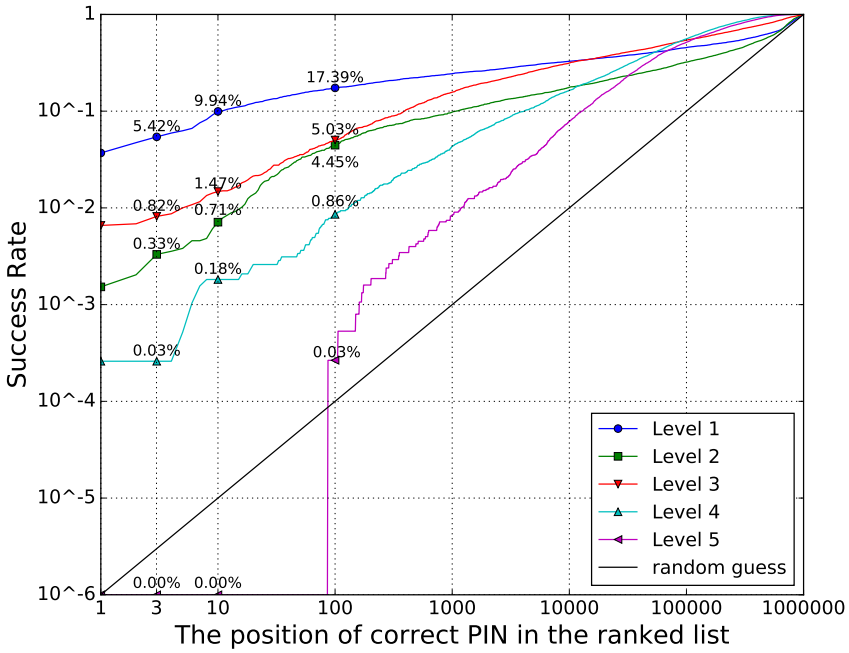


Figure 3.5: The performance of *general attacks*.

3.5.1 Target Attacks

We first examine whether the performance of our attacks can be improved using target victims' data for model training which is also used in HMM-based attacks in the literature [119, 151, 71]. Hence, we propose *targeted attacks* where an attacker obtains a small amount of inter-keystroke timing data about a victim typing known numerical sequences for model training. Although both *targeted attacks* and HMM-based attacks train their models based on a victim's own data, our approach requires much less training data. Our approach requires an attacker to know the inter-keystroke timing data about a few key pairs rather than all key pairs as required in HMM-based attacks. To collect such training data in practice, an attacker may trick a victim to install malware on his/her smartphone and collect inter-keystroke timing data when the victim dials phone numbers. Another possible way of collecting such data is to trick a victim to enter insensitive numerical sequences through phishing websites or phishing phone calls.

The procedure of the experiment of *targeted attacks* is similar to that of *general attacks* except that we use the inter-keystroke timing data from the testing session

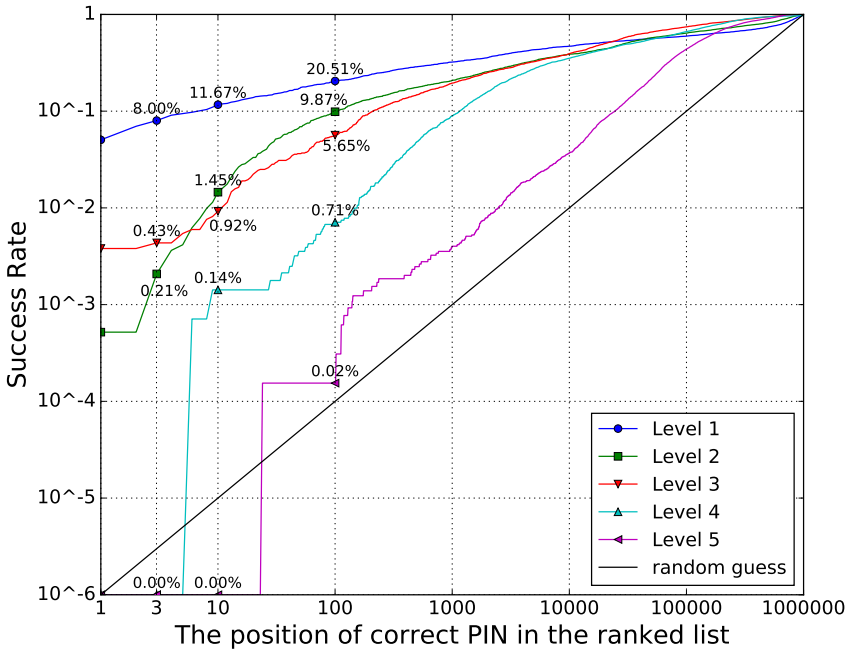


Figure 3.6: The performance of *targeted attacks*.

of our user study to train a cognitive model. In particular, to attack any one of the 25 PINs entered by a participant, we randomly choose 2 other PINs out of the 25 PINs entered by the same participant and use 30 collected inter-keystroke timing sequences for these 2 PINs for model training. In comparison, previous HMM-based attacks require that an attacker should obtain 30-50 inter-keystroke timing sequence for *each of 110 key pairs* (10×10 digit-to-digit key pairs and 10 digit-to-<Enter> key pairs) from a victim for model training. The same as the *general attacks*, we take each PIN entry typed by the participants from the testing session as an independent attacking case.

Figure 3.6 shows that *targeted attacks* have a similar trend as *general attacks* in terms of the effectiveness of attacking PINs at different PIN strength levels. Compared to *general attacks*, the success rate of *targeted attacks* is improved by about 4% on average for all levels. Considering that *targeted attacks* are user dependent, and they do not improve the success rate significantly over the *general attacks*, attackers may still prefer *general attacks* in practice.

3.5.2 Multi-Entry Attacks

We then examine whether an attacker can improve his/her success rate if he/she observes the victim's PIN entry for multiple times. Hence, we propose *multi-entry attacks* where an attacker captures the inter-keystroke timing sequences about a victim entering the same PIN for multiple times. With k inter-keystroke timing sequences of one PIN, an attacker can calculate an averaged timing sequence for PIN inference.

First, the attacker normalizes each observed PIN entry's inter-keystroke timing sequence so as to attain the same amplitude. The ratio of Sum_i to $\overline{\text{Sum}}$ is considered as the scaling value for the i-th inter-keystroke timing sequence $\overrightarrow{T_i} = (\Delta T_{i1}, \Delta T_{i2}, ..., \Delta T_{il})$, where $\text{Sum}_i = \sum_{1 \leq j \leq l} \Delta T_{ij}$, $\overline{\text{Sum}} = \frac{1}{k} \sum_{1 \leq i \leq k} \text{Sum}_i$, and l is the PIN length.

Then, the attacker calculates the i-th scaled inter-keystroke timing sequence $\overrightarrow{T'_i} = \overrightarrow{T_i} \times (\overline{\text{Sum}}/\text{Sum}_i)$. Given k scaled timing sequences $\overrightarrow{T'_1} = (\Delta T'_{11}, \Delta T'_{12}, ..., \Delta T'_{1l})$, ..., $\overrightarrow{T'_k} = (\Delta T'_{k1}, \Delta T'_{k2}, ..., \Delta T'_{kl})$, the attacker generates an averaged timing sequence $(\Delta\overline{T'_1}, \Delta\overline{T'_2}, ..., \Delta\overline{T'_l})$ where $\Delta\overline{T'_j} = \frac{1}{k} \sum_{1 \leq i \leq k} \Delta T'_{ij}$.

Similar to the *general attacks*, the attacker trains a cognitive model from other users' inputs and builds a timing dictionary. The attacker then calculates the similarity between the calculated averaged timing sequence and each entry in the timing dictionary and ranks all PINs according to their similarity values. Finally, the attacker attempts to login to the victim's account using the PINs starting from the top in the ranked list in an online attack.

In the experiment of *multi-entry attacks*, the same cognitive model and timing dictionary are used as in *general attacks*. We take the averaged inter-keystroke timing information of 10 PIN entries (i.e., $k = 10$) from each participant as an independent case. The procedure of the experiment of *multi-entry attacks* is similar to that of *general attacks* except that we take the averaged timing sequence as the observed timing sequence in each attacking case. Figure 3.7 shows that *multi-entry*

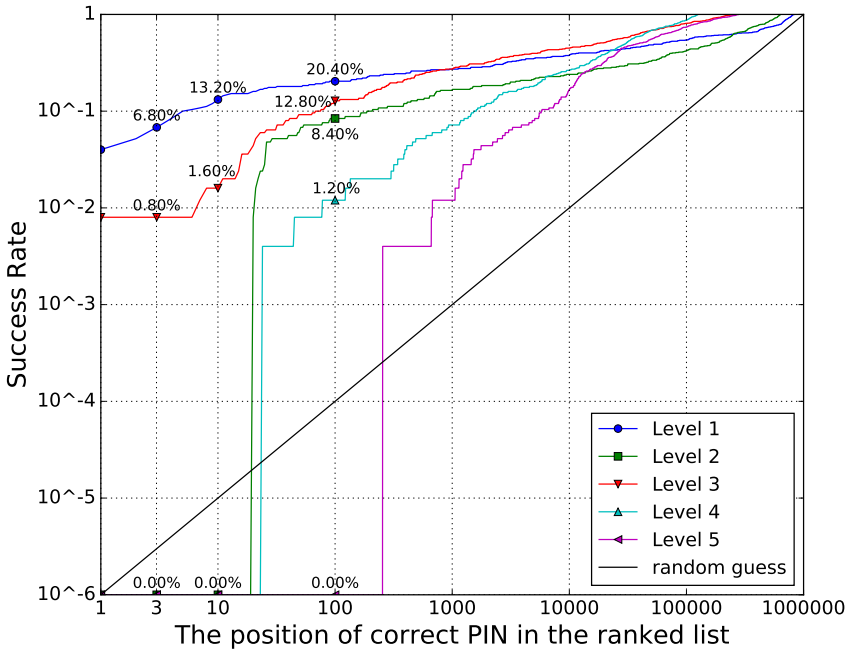


Figure 3.7: The performance of *multi-entry attacks*.

attacks have a similar trend as *general attacks* in terms of the effectiveness of attacking PINs at different PIN strength levels. Compared to *general attacks*, *multi-entry attacks* achieves better performance when x ranges from 100 to 10^6 but achieves worse performance to the PINs at *level 2, 4, 5* when x ranges from 1 to 100. One possible reason is that the number of samples in *multi-entry attacks* is much less than *general attacks* and the observation of finding the position of correct PIN in the top 100 is based on a large number of samples. In general, *multi-entry attacks* outperform *general attacks* with insignificant improvement (below 2% on average for all levels).

3.5.3 Known Digits Attacks

Considering that both *targeted attacks* and *multi-entry attacks* bring little improvement over *general attacks*, we further propose *known digits attacks* which improve the success rate significantly.

In this case, an attacker knows certain digits of a target PIN before launching inter-keystroke timing attacks (e.g., through other side channel attacks[32, 150, 115,

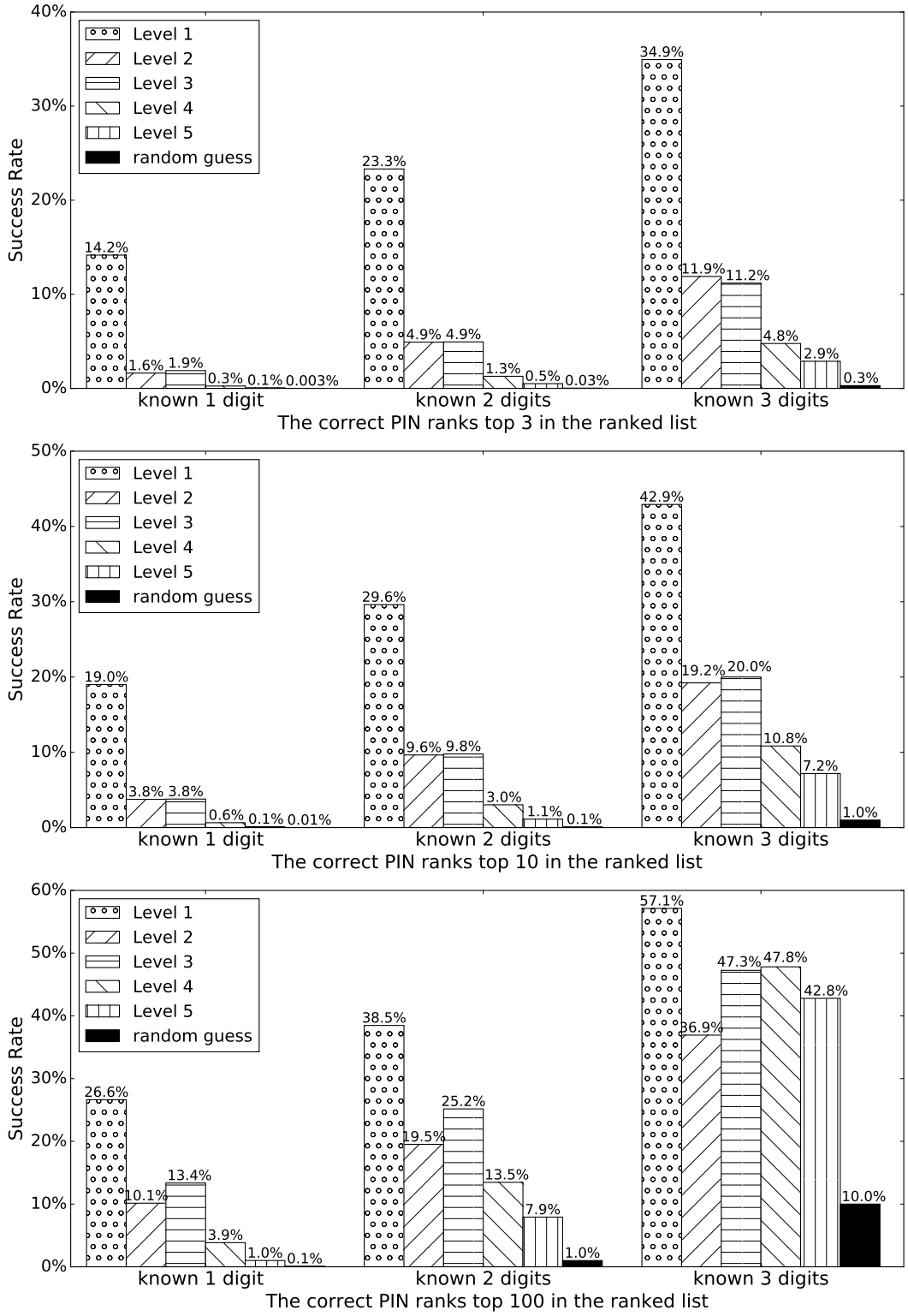


Figure 3.8: The performance of *known digits attacks*.

80, 135, 126] or shoulder surfing attacks [127]). Hence, he/she can reduce the size of his/her timing dictionary. For example, if the first two digits are known to the attacker which are '1' and '2', the reduced timing dictionary consists of 10^4

candidate PINs which range from '120000' to '129999'. The attacker measures the similarity between the observed timing sequence and each timing sequences in the reduced timing dictionary and ranks these 10^4 candidate PINs according to their similarity values. Finally, the attacker attempts to login to the victim's account using the PINs starting from the top in the ranked list in an online attack.

In the experiment of *known digits attacks*, we use the same cognitive model as in *general attacks* and generate reduced timing dictionaries. We evaluate the cases where an attacker obtains 1, 2 or 3 digit(s) of a target PIN. For each attacking case, one inter-keystroke timing sequence for each PIN entry is used. We enumerate all cases where the known value(s) are at any position(s) of the target PIN (i.e., 6 cases for known 1 digit, 15 cases for known 2 digits and 20 cases for known 3 digits for each PIN entry). For the similarity calculation, we measure the similarity between the observed timing sequence and each entry in the corresponding reduced timing dictionary. When an attacker knows any k digits of an l-digit PIN, the success rate of random guessing attacks is $\binom{10^{l-k}-1}{x-1} / \binom{10^{l-k}}{x}$ where x is the maximum number of allowed consecutive failures.

The results of *known digits attacks* are shown in Figure 3.8. It is clear that the success rate of *known digits attacks* is significantly higher than *general attacks*. For example, the success rates of inferring a target PIN at *level 1* within 3 attempts are 14.2%, 23.3%, and 34.9% when 1 digit, 2 digits, and 3 digits are known by the attacker, and they are 1.6, 3.3, 5.4 times higher than *general attacks*, respectively. In many cases, the success rate of guessing the correct PIN is above 10%. The results show that *known digits attacks* are more practical than *general attacks*. Even *known digits attacks* are applied to attack a single user or a small number of users, their success rates are not impractically low. These results also indicate the effectiveness of our inter-keystroke timing attacks to 3, 4 or 5-digits PINs and that the attacks pose a greater threat to shorter PINs as expected.

3.6 Discussions

In this section, we compare our attacks with HMM-based attacks to show the merits of our approach. Then, we demonstrate that our attacks pose a serious threat to real-world applications when applied at large scale. Next, we propose several feasible countermeasures to mitigate our attacks. Lastly, we discuss the limitations of our attacks.

3.6.1 Comparison with HMM-based Attacks

Most keystroke timing attacks in the literature follow a similar attacking framework based on a Hidden Markov Model (HMM) [119, 151, 71]. Compared to HMM-based attacks, our attacks have two merits.

The first merit is that our attacks are user-independent. The cognitive model in our attacks captures the common characteristics across all skilled users typing PINs so that it can be used to attack any users. In addition, the use of cosine similarity in our attacks enables an attacker to rank all candidate PINs similarly for inferring a target PIN even if different users may type the target PIN with different speeds. In comparison, the HMM-based attacks relies on the distribution of inter-keystroke timing for a specific user typing each possible key pair so as to calculate the probability of any possible underlying keystroke sequence given an observed inter-keystroke timing sequence. Because the distribution of inter-keystroke timing for different users typing any same key pair may not be similar, the HMM-based attacks are user-dependent. They require that an HMM be trained with the inter-keystroke timing data for all possible key pairs collected from a target user, and such a model is user dependent and has to be retrained from scratch if the target user changes.

The second merit is that the cognitive model used in our attacks can be trained based on inter-keystroke time intervals for a small number of key pairs (minimum two key pairs). To launch an HMM-based attack, however, an attacker needs to

collect a sufficiently large number of inter-keystroke time intervals for *each possible* key pair from a target user before launching the attack. For PIN inference, an attacker needs to capture 30-50 inter-keystroke time intervals for each of 110 key pairs (including 10×10 digit-to-digit key pairs and 10 digit-to-<Enter> key pairs) from a target user. It is usually difficult for an attacker to collect such large amount of data before launching an online attack in practical settings. Under the adversary model of our attacks, attackers cannot collect enough training data to support HMM-based attacks.

3.6.2 Attack Threats to Real-World Applications

In general, the success rate of the proposed attacks may not be sufficiently high to pose imminent danger to an individual user's PIN if the attacker does not have prior knowledge on any digits of the target PIN. However, our attacks are practical in online settings because the attacks are user-independent and thus can be applied to attack any number of users' PINs in a large scale. To show the threats of our attacks to real-world applications, we provide two examples where PINs are used as the only credential to protect users' accounts and where attackers can collect many users' inter-keystroke timing data for PIN entries using malicious JavaScripts.

One example is an Internet banking system of bank with pseudonym XYZ, which is the largest bank in a Southeast Asia country. It has more than three million Internet banking users. To login to an Internet banking account, a user needs to input a user ID and a 6 to 9-digit PIN as the credential (most users choose 6-digit PINs, which is the default case). Our tests show that users are not blocked within 50 login attempts. Although certain financial services (e.g. bank transactions) require a second-factor authentication, much sensitive information (e.g. account balances, usernames, addresses) can be leaked merely after PIN authentication. If ten percent of users' inter-keystroke timing data about PIN entries were collected, our online attacks can be applied to all these users' accounts with 50 tries per account, which

do not lead to any account being locked in practice. Consequently, On the average, around 4.16% of users' accounts would be compromised according to our experimental results. In other words, more than 12,000 users' accounts would be compromised due to our attacks.

The other example is ABCpay (pseudonym) which is the largest third-party mobile and online payment platform in Asia. It has more than 520 million users over the world. To make a payment through its service, a user needs to input his/her mobile phone number as user ID and a 6-digit PIN as password. It is not difficult for attackers to obtain many users' names, mobile phone numbers, and email addresses by crawling public web pages. The login attempts of each user's account in this platform is limited to 3. On the average, an attacker needs to launch online attacks to 83 users' accounts in order to compromise one account. In other words, if our attacks were applied to 1/1000 of users' accounts, then 6,000 users' accounts would be compromised on the average. Our attacks would cause serious damages in this case since attackers can transfer money from victims' accounts to other accounts.

Considering that many financial institutions have a large number of users and that malicious JavaScripts are easy to spread, our attacks pose a serious threat to real-world applications when applied in large scale.

3.6.3 Mitigations

Increasing PIN length. The security strength of the most existing PIN systems is chosen according to the success probability of random guessing attacks [147]. For example, the security strength for 6-digit PINs is considered to be 10^{-6}. However, our study reveals that the inter-keystroke timing attacks significantly lower the security strength of PIN systems. A simple approach to mitigating this threat is to increase the PIN length. Our calculation suggests that users should increase 6-digit PINs to 10-digit PINs whose security strength under the inter-keystroke timing attacks is higher than that of 6-digit PINs under the random guessing attacks on

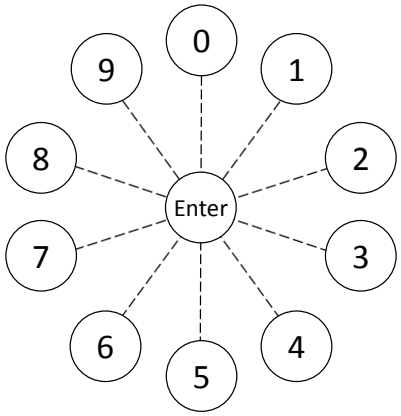


Figure 3.9: A new keypad layout.

the average. This mitigation does not require any change to the hardware of current PIN authentication systems, but at the expense of requiring users to memorize longer PINs.

PIN selection policy. As shown in Section 3.4.3, the performance of our attacks varies significantly when they apply to PINs at different strength levels. If a user selects a 6-digit PIN at *level 5* instead of *level 1* to *level 4*, the probability of a successful *general attack* within 100 attempts can be reduced by 870, 222, 251, and 43 times, respectively.

We thus suggest adopting a PIN selection policy where a user is required to choose a PIN at *level 5* when the user registers his/her account. If a user chooses a PIN at *level 1* to *level 4*, his/her registration would not succeed until the user changes his/her PIN to *level 5*. *Level 5* consists of $9 * 10^5$ PINs which account for 90% of all 6-digit PINs. It is thus relatively easy for a user to obtain a PIN at *level 5* if he/she simply chooses his/her PIN randomly.

Considering that the success rate of attacking 6-digit PINs at *level 5* is still around 10 times higher than random guessing attacks, to achieve a similar security strength of 6-digit PINs under the random guessing attacks, we suggest users choose 7-digit PINs at the strongest strength level. Note that when the PIN selection policy is adopted, it is unnecessary for users to choose 10-digit PINs which has been mentioned earlier.

A new keypad layout. As it is shown in the cognitive model, the inter-keystroke timing measurement for a user types a key pair on a keypad is mainly determined by the distance between the two keys of the key pair on the keypad. Based on this observation, we design a new keypad for PIN entry to mitigate inter-keystroke timing attacks. As shown in Figure 3.9, the keypad is in a circular shape. All 10 digits (0-9) keys are evenly distributed on a circle. An <Enter> key is located in the center of the keypad. When a user types his/her PIN, the user presses <Enter> key after pressing each digit. When submitting the PIN, the user may press the <Enter> key twice. During the PIN entry, the user always moves his/her finger through the same distance for entering any digit, leading to similar inter-keystroke timing sequence for entering any PINs.

Although a user may take double time for entering his/her PIN on this new keypad, the security strength of a PIN system is improved significantly against the inter-keystroke timing attacks. If this new keypad is adopted, the success rate of inter-keystroke timing attacks would be similar to that of random guessing attacks. We implement this keypad on a smartphone where the distances between any digit key and <Enter> key is 1 inch. It takes around 2.5 seconds for a user to enter a 6-digit PIN on the new keypad. In comparison, most existing leakage resilient password systems which have the same security strength as that of 6-digit PINs require hundreds of seconds for user authentication (e.g. [61, 7, 140, 74]).

Leakage resilient password systems (LRPSes). LRPSes [6, 61, 7, 140, 74, 15, 141, 148] are user authentication systems which do not disclose user credentials to observers. Such systems are by design secure against any side-channel attacks including our attacks. However, Yan et al. [147] point out that in order to be secure, LRPSes have remarkably low usability. A secure LRPS usually takes hundreds of seconds to complete an authentication session, which may not be practical in many applications [61, 7, 140, 74].

3.6.4 Limitations

Ecological validity. In our user study, we recruit students and young staff only from a single university. The performance of our attacks may vary among different populations. The ecological validity of our user study is limited, but the qualitative facts in our research are likely to remain true.

Typing styles. In our experiments, we require all participants to enter their PINs on a keypad using a single finger. We conducted a larger scale survey on user's typing habits through emails and social networks over three weeks. In total, we received 544 responses. The participants of the survey mainly came from Singapore, China and UK. They were not limited to the students or staffs in universities. According to the survey results, most participants reported that they tend to use a single finger when they enter PINs on numeric keypads in real life. In particular, 344 participants (63.2%) use one finger, 124 participants (22.8%) use two fingers, and 76 participants (14.0%) use more than two fingers for PIN entry. In order to attack users who use multiple fingers when typing PINs, our cognitive model should be extended to cover different typing styles.

Typing error. During the process of entering PINs in real-world scenarios, users may press a wrong digit and then use the <Delete> or <Backspace> key to cancel the wrong input. We exclude this situation because it rarely happens in PIN entries. We plan to address this issue and generalize our attacks for password inference in the future.

Chapter 4

UltraPIN: Inferring PIN Entries via Ultrasound

4.1 Introduction

This chapter proposes UltraPIN, a novel and practical attack to recover PIN with commodity smartphones by analyzing imperceptible acoustic signals issued by smartphones and reflected by finger movements during PIN entries. Personal identification numbers (PINs) are numerical codes used widely in our daily life. PIN-based user authentication originated with the introduction of automated teller machines (ATM) in 1967 [132] and it is still irreplaceable owing to the advantages of its low-cost, ease of deployment, and relatively good security. According to a recent report [103], about 3.24 million ATMs had been installed worldwide by 2018, and most of them use PINs to identify users.

As claimed by Ravi Sandhu in 2003, PIN-based ATM systems provide good-enough security in worldwide scale [111]. It is widely accepted that such systems are secure enough while providing user-friendly services. The current ATM frauds are nonzero, but they require highly capable attackers at a cost. In addition, current ATM frauds focus mainly on stealing card numbers rather than stealing PINs. According to a recent report [53], more than 46% data breaches lead to debit/credit

Figure 4.1: Examples of keypads for PIN entries

card numbers being compromised while only 6% data breaches result in compromised PINs. Current PIN stealing approaches, including shoulder surfing, hidden camera spying, and social engineering based guessing, are difficult to launch in a large scale. So far, it is mostly comfortable for users to use ATMs with tolerable risks.

However, PIN-based user authentication may not be as secure as it appears to be. In this chapter, we demonstrate a novel and practical attack, named UltraPIN, to infer a typed PIN effectively. It leverages on an acoustic side-channel to infer PIN entries on various keypads as shown in Figure 4.1. To launch an attack, an attacker may pretend to be a consumer lining up behind a target victim or stay nearby, holding smartphones in a distance as normal customers keep, where no line-of-sight is seen between the attacker and the victim's hand during PIN entry. The attacker starts an UltraPIN app on his/her smartphones before the victim enters a PIN. Then, UltraPIN continuously plays an *inaudible* ultrasound signal above 20 kHz and below 24 kHz via the attacker's smartphone speakers and keeps recording the received signal from smartphone microphones during PIN entry. When the victim finishes PIN entry, UltraPIN processes the recorded signal and displays a ranked list of candidate PINs on the attacker's smartphones. It is highly likely that the attacker can discover the victim's PIN by testing several candidate PINs from the ranked list on a target PIN-based user authentication system.

UltraPIN poses a serious threat to PIN based user authentication. On the one

hand, cloning of ATM cards or credit cards is not an uncommon attack in the real world [82]. Even with a cloned card, it still requires an attacker to know the correct PIN in order to withdraw money from ATM or make a credit card payment in certain countries. In such cases, UltraPIN serves as the missing piece to complete the attack puzzle. On the other hand, users may reuse their ATM PINs in other applications such as online banking and mobile payment [22]. UltraPIN would enable attacks to such applications without the need of cloning victims' cards.

UltraPIN can also be applied to compromise electronic door locks operated on PINs. Nowadays, more and more condos, hotels, and office buildings use such electronic door locks. In this scenario, staying too close to a target victim may raise the victim's suspicion. Instead, an attacker may deploy hidden wireless speakers and wireless microphones around the victim's door and stay 10 meters away from the victim, which should not raise the victim's alert. During the victim's PIN entry, the attacker uses UltraPIN on his/her smartphones to control the hidden speakers and microphones via Bluetooth, while the rest of the attack is the same as in the ATM case. Compared to hidden-camera spying where a high-resolution camera must be placed at proper position and angle so as to capture the victim's finger movements with a line of sight, UltraPIN is more practical since the hidden speakers and microphones can be placed at any positions near the door lock.

UltraPIN is designed based on the fact that a user's finger movements on a keypad during PIN entry may cause distinguishable frequency changes (i.e., Doppler shifts) of an acoustic signal and such changes reflect the user's finger movements. Our further studies demonstrate that different finger movement patterns can be classified according to the Doppler shifts; consequently, a ranked list of possible PINs can be inferred from the classification results using a commodity smartphone.

The Doppler shift effect has been widely exploited by Doppler radars to locate moving objects [142], for example, on fighter aircrafts and in meteorological observations [34]. Such specialized Doppler radars cannot function without emitting microwaves at Giga-Hz level [142]. However, commodity smartphones used by

UltraPIN cannot generate any acoustic signals above 24 kHz.

Several recent works [154, 138] exploit smartphones for tracking finger movements from a distance shorter than 10 cm. In comparison, UltraPIN requires the distance to be greater than 50 cm in order not to raise the victim's suspicion. The previous finger movement tracing techniques cannot be applied in UltraPIN for two reasons. The first reason is that the energy of Doppler shifts recorded by attacker's smartphones is much lower (around 1/115) in our attacking scenario in comparison to the Doppler shifts measured at a distance that is commonly used in the previous works. In addition, the acoustic signals recorded by attacker's smartphones involve much more noises due to signal reflections from surrounding static objects and moving objects in our attacking scenario, which makes it more difficult to discern finger movements and thus infer a PIN.

The second reason is that finger movements in the previous works are traced based on a triangle relationship among a target user's finger, a speaker and a microphone (both are very close to the finger), where the ratio of the longest side to the shortest side in the triangle is around 2:1. However, this ratio is from 10:1 to 30:1 in our attacking scenario, which would enlarge errors significantly in finger movement tracing. In our attacks, the distance between microphone and speaker (which are installed in the attacker's smartphone) is neglected as it is much shorter than the distance between the victim's finger and the attacker's smartphone. Consequently, the previous finger movement tracing techniques are not applicable.

A series of technical innovations are developed to make UltraPIN effective and robust. First, high-quality feature vectors are derived from low-energy and high-noise ultrasound signals that are received by UltraPIN for inferring a target PIN entry. This is achieved through (i) keypair segmentation, where a received signal is partitioned into successive keypair segments; (ii) signal energy enhancement, where the signal energy of each keypair segment is enhanced by signal differentiation; (iii) signal carrier removal, where the source signal and the signals reflected by surrounding static objects are effectively removed; and (iv) frequency contour

extraction, where the feature vector of each keypair segment is extracted from the Doppler shifts of typing finger movements after the Doppler shifts of other moving objects are filtered out.

Given feature vectors, UltraPIN generates a ranked list of possible PINs as PIN inference result, from which an attacker can test on a target PIN-based user authentication system. UltraPIN adopts a backward PIN inference strategy to achieve this. First, UltraPIN trains a convolutional neural network (CNN) for classing a number-to-Enter keypair from its feature vectors, and generating a ranked list for the last PIN digit. Next, UltraPIN classifies each number-to-number keypair in a reverse order, producing a ranked list for each PIN digit until the first digit. In this process, UltraPIN trains a support vector machine (SVM) to determine whether or not a keypair is a repeated keypair. It further applies a CNN model to classify the keypair among all non-repeated keypair classes. The last step of UltraPIN is to generate an overall ranked list for inferring the target PIN, where the PINs in the list are ranked in an anti-lexicographic order according to the ranked list for each PIN digit.

A series of experiments is conducted to evaluate the performance of UltraPIN. Experimental results show that the success rate of recovering a PIN within three attempts is about 75% in our recommended setting. This is a significant improvement over random guessing, which requires about 750,000 attempts to achieve a similar success rate. Experimental results also show that UltraPIN is user-independent and robust. UltraPIN can be trained with data collected from *any* users, such as attackers themselves or people hired by them; once trained, UltraPIN can be used to attack any victims for PIN inference with high success rates. The success rates of UltraPIN remain stable even if the settings for PIN inference are moderately different from the settings in which UltraPIN is trained.

The runtime performance of UltraPIN is evaluated on commodity devices. It takes hours for UltraPIN to be trained on a laptop with GPU in the training phase, and less than a second for running on a commodity smartphone in the attacking phase, with memory requirement below 207 MB on smartphone. Toward the end of

this chapter, we also discuss an effective countermeasure to thwart UltraPIN attacks.

4.2 Preliminaries

In this section, we introduce the capability of current commodity smartphones which are used by UltraPIN. We then introduce the concept of Doppler effect and clarify the attack model for the design of UltraPIN.

4.2.1 Smartphones

Most commodity smartphones in the market are equipped with at least one microphone and one speaker. Certain high-performance smartphones (e.g., iPhone XR, iPhone XS [63], and Huawei P20) enjoy more than one pairs of microphone and speaker to reduce background noises and support stereo sounds.

With powerful microphones and speakers, commodity smartphones may be used for keystroke inference. Recent studies [72, 39] have shown that smartphone speakers are capable of transmitting sound signals with a frequency up to 24 kHz. Moreover, smartphone microphones support audio recording at a sampling rate of 44.1 kHz or 48 kHz [45].

Smartphone has become a promising platform for machine learning and deep learning which UltraPIN exploits for PIN inference. Apple has implemented a neural engine on its iPhone X and newer models to accelerate artificial-intelligence software [116]. Android system has integrated TensorFlow Lite (TFLite) framework since 2017 [52]. It is thus not difficult to deploy UltraPIN on learning-capable smartphones. On the other hand, if an attacker's smartphones are not capable of running UltraPIN, the attacker can still use his/her smartphones to collect PIN entry information on-site, and send such information to a remote server or a personal computer on which UltraPIN's learning models are running for PIN inference.

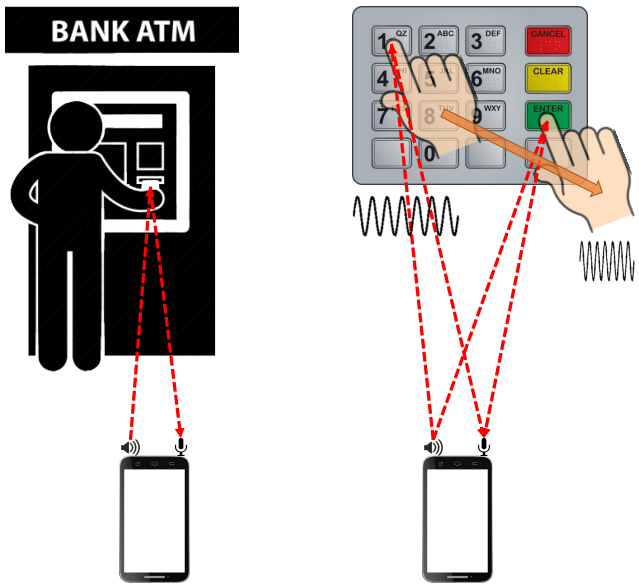


Figure 4.2: Sensing finger movement for PIN entry

4.2.2 Doppler Effect

Doppler effect is the change in frequency or wavelength of a wave (sound wave in our context) for an observer who is moving in reference to the wave source. A common example of Doppler effect is that the frequency of an ambulance's siren increases as the ambulance comes towards a listener and diminishes as it passes away. This phenomenon was first described by Austrian physicist Christian Doppler and has been widely exploited in astronomical measurements, radar, and modern navigation [27, 142].

UltraPIN leverages on the Doppler effect of ultrasound signals to sense finger movements during PIN entries. Figure 4.2 illustrates an attacking scenario in which a user is typing a PIN on an ATM while a nearby attacker uses a smartphone to sense the user's finger movements for PIN inference. During this process, the smartphone emits a human-inaudible ultrasound signal via its built-in speaker, and records the received signals with its microphone. The Doppler effect in the received signals is then analyzed by UltraPIN for PIN inference.

In particular, a finger typing a PIN can be considered as a virtual transmitter producing reflected sound waves, where the distance between speaker and microphone

on the same smartphone can be ignored in comparison to the much larger distance between the smartphone and the target user. If the typing finger moves towards the smartphone, the reflected frequency should be higher than the emitted frequency. Conversely, if the finger moves away from the smartphone, the reflected frequency should be lower than the emitted frequency.

If a typing finger moves at speed v, it generates a reflected sound wave at a shifted frequency f_r:

$$f_r = f_t \left(\frac{1 + v \cdot \cos(\alpha)/v_s}{1 - v \cdot \cos(\alpha)/v_s} \right) = f_t \left(\frac{v_s + v \cdot \cos(\alpha)}{v_s - v \cdot \cos(\alpha)} \right), \tag{4.1}$$

where f_t is the frequency of the transmitted sound wave, v_s is the speed of sound in the medium, and α is the angle between the finger's forward velocity and the line of sight that is between the finger and the smartphone's microphone. The Doppler shift f_d can be computed as:

$$f_d = f_r - f_t = 2v \cdot \cos(\alpha) \left(\frac{f_t}{v_s - v \cdot \cos(\alpha)} \right), \tag{4.2}$$

Since the finger movement velocity is much smaller than the speed of sound (i.e., $v \cdot \cos(\alpha) \ll v_s$), we have $(v_s - v \cdot \cos(\alpha)) \rightarrow v_s$, and thus:

$$f_d \approx 2v \cdot \cos(\alpha) \frac{f_t}{v_s} \tag{4.3}$$

Equation (4.3) indicates that a faster radial velocity (i.e., larger $v \cdot \cos(\alpha)$) leads to a larger frequency shift. The variation of Doppler shifts depends on the direction of the typing finger. The Doppler shift reaches its maximum when the typing finger moves directly toward or away from the smartphone, and it diminishes with increasing angle between the direction of finger motion and the direction of ultrasound wave, until no Doppler shift is observed when the two directions are perpendicular. The magnitude of Doppler shift can thus be used to detect the directions of finger movements during PIN entry.

UltraPIN chooses to emit ultrasound signals in a range of 20 kHz to 24 kHz. Such ultrasound signals can be emitted from smartphone speakers and recorded with smartphone microphones, but they are inaudible to human ears. The human range of hearing is commonly given as from 20 Hz to 20 kHz. Since the speed of sound is $346.6\ m/s$ in dry air at 26°C, the speed resolution is in a range of $\frac{346.6}{24000} = 1.44\ cm/s$ to $\frac{346.6}{20000} = 1.73\ cm/s$. It is significantly smaller than the averaged speed of finger movement during PIN entry, which is in a range of $10\ cm/s$ to $20\ cm/s$ [128, 79]. In theory, the resolution of UltraPIN is sufficient for sensing typing finger movements from Doppler effect. The challenge is to ensure that the sensing of typing finger movements is accurate enough for PIN inference with low-energy of received signals in the presence of high noises in practice.

4.2.3 Attack Model

We consider a PIN inference attack happening on the site where a user enters his/her PIN on a numeric keypad. The numeric keypad can be either hardware keypad mounted on ATM, point of sale (POS) terminal, and electronic door lock, or software keypad displayed on smartphone, smart tablet, and any other devices equipped with a touchscreen. The experiments of this work focus on ATM keypads and POS keypads, which are shown in Figure 4.6.

An attacker's objective is to infer the target user's PIN after PIN entry using smartphones without raising the target user's awareness. An attacker is referred to either a single person holding one or more smartphones, or a small group of persons each holding a smartphone. It is possible more than one smartphones are used in a PIN inference attack. In case multiple smartphones are used, UltraPIN controls all of them emitting inaudible ultrasound signals at different frequencies in the range of 20 kHz to 24 kHz so as to avoid interference between their signals. It is assumed that an attacker knows the layout of the keypad used for PIN entry. This is a reasonable assumption since the layout of keypad is either standardized or can

be easily obtained in a public place.

We consider two typical attack scenarios, including ATM/POS attack and electronic door lock attack. In ATM/POS attack, an attacker stays close to a victim in a distance around 50 cm to 150 cm without raising the victim's alert. The attacker starts an UltraPIN app on his/her smartphones before the victim enters a PIN on an ATM or POS keypad. UltraPIN continuously plays ultrasound signals via the attacker's smartphone speakers and keeps recording the received signals from smartphone microphones during PIN entry. After PIN entry, UltraPIN processes the recorded signals and displays a ranked list of candidate PINs as PIN inference result.

An electronic door lock attack targets at someone entering a PIN for opening a PIN-operated door lock. If an attacker stays too close to the person as in the ATM/POS case, the attacker's behavior may raise an alert. In such case, we suggest that an attacker should place one or more hidden wireless speakers and microphones (e.g., AirPods [5], HUAWEI FreeBuds [62]) at any places close to the door (e.g., within 50 cm-150 cm) and stay 10 meters away from the victim in order not to raise the victim's attention. Instead of using smartphone's built-in speakers and microphones, UltraPIN remotely controls the wireless speakers and microphones via Bluetooth, while the rest of the attack remains the same as in the first scenario.

As for a victim's PIN typing behavior, it is assumed that a single finger is used to enter the whole PIN followed by an "Enter" keypress to signal the end of a PIN entry process. According to our survey [79], a majority of users (about 63.2%) prefer using a single finger for PIN entry.

4.3 UltraPIN Design

4.3.1 Overview

UltraPIN consists of six major components, including *Audio Capturing*, *Keypair Segmentation*, *Feature Extraction*, *Movement Modeling*, *Learning Models*, and *Keypair Combination*. While an attacker may use multiple smartphones in PIN inference attack, we focus on one smartphone in explaining the components unless otherwise clarified since they are mostly the same on different smartphones. While UltraPIN can be easily extended to attack PINs of various sizes, we focus on 6-digit PINs, entering of which requires pressing six number-keys and one "Enter" key in this work.

In *Audio Capturing*, UltraPIN triggers an attacker's smartphone to emit an inaudible ultrasound signal continuously through its speaker with a frequency taken from the range of 20 kHz to 24 kHz during PIN entry. Meanwhile, UltraPIN activates the smartphone's microphone to keep recording acoustic signal in this process.

In *Keypair Segmentation*, UltraPIN parses the recorded signal once the PIN entry process is complete, and segments the recorded signal into a sequence of six signal segments, each of which corresponds to entering a keypair (i.e., moving a typing finger from the previously pressed key to the current key) during PIN entry.

In *Feature Extraction*, UltraPIN extracts a feature vector from each segment of signal according to its Doppler effect so as to represent the typing finger movement of pressing a keypair during PIN entry. Because the recorded signal is usually weak and involves high noises, UltraPIN applies a series of signal processing techniques to improve the quality of feature extraction.

In *Movement Modeling*, UltraPIN groups all possible keypairs, including 100 number-to-number keypairs and 10 number-to-Enter keypairs, into 41 classes. The keypairs in each class share the same finger movement direction and displacement during PIN entry, which means that typing them during PIN entries causes similar

feature vectors to be extracted.

In *Learning Models*, UltraPIN retrieves a feature vector as input for classifying each keypair segment. In the case multiple smartphones are used for the PIN inference attack, all feature vectors that are retrieved from these phones are used for classifying each keypair. To maximize PIN inference accuracy, UltraPIN takes a backward PIN inference strategy to classify the last keypair (i.e., number-to-Enter keypair) first, and the other keypairs (i.e., number-to-number keypairs) in a reverse order using two CNN models and one SVM model.

Finally, in *Keypair Combination*, UltraPIN combines candidate keypairs in the reverse order and generates a ranked list of possible PINs. An attacker may test the PINs one by one from the list on a target PIN-based user authentication system (e.g., ATM, POS, and electronic door lock) within a limit of tries.

4.3.2 Audio Capturing

Audio Capturing leverages on either built-in speaker and microphone on a smartphone or wireless speaker and microphone connected to the smartphone. During a PIN entry process, the speaker emits a continuous acoustic wave $A\sin(2\pi ft)$ with amplitude A and frequency f. Frequency f can be any value taken from 20 kHz to 24 kHz, which is beyond the hiring range of healthy young people (i.e., 20 Hz - 20 kHz [104, 100, 118]), and within a smartphone's capability (see Section 4.2.1). In our experiments, a PIN inference attack may employ at most three smartphones for which the frequencies are set at 20.5 kHz, 21.5 kHz, and 22.5 kHz, respectively.

While the acoustic tone is played, the microphone keeps recording received signal. The sampling rate of signal recording should be no lower than 48 kHz according to Nyquist–Shannon sampling theorem. In our experiments, the sampling rate is set at 48 kHz, which is sufficient for launching effective attacks.

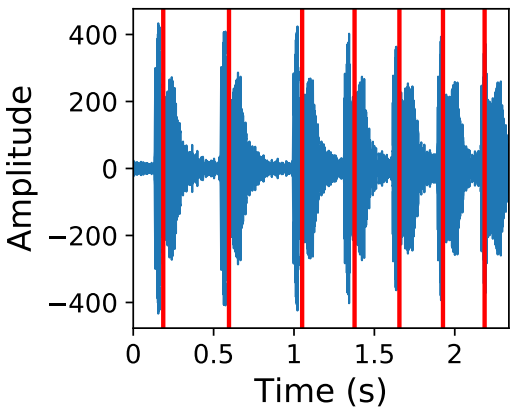


Figure 4.3: Time domain of a beep sound signal and key press timestamps

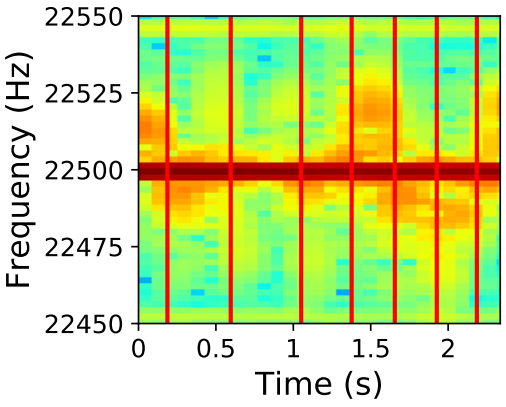


Figure 4.4: Frequency spectrum of an ultrasound signal and its partition

4.3.3 Keypair Segmentation

Keypair Segmentation segments the recorded acoustic signal into six successive segments, with each segment representing a keypair during PIN entry. Most, if not all, PIN entry keypads produce a clear beep sound when a user presses a key. Note that the beep sound is usually produced from a sound speaker connected to the keypad instead of from the pressed key itself. This is to avoid keypad acoustic emanation which leads to acoustic signature attacks [9]. While providing feedbacks to users' keypress actions, such beep sounds serve as keypress landmarks for keypair segmentation.

Keypair Segmentation identifies the timestamp of each keypress from the recorded signal by locating the peak point of corresponding beep sound. To achieve this goal, it first applies two band-pass filters to the recorded signal, one in the ultrasound frequency range $f \pm 50$ Hz for producing an ultrasound signal that encodes finger movement patterns, and the other in the hearing range $f' \pm 200$ Hz for producing a beep-sound signal, where f' is the average frequency of beep sounds, which is 500 Hz in our experiments. It then transforms the beep-sound signal into energy levels $e(t)$ using windowed Discrete Fourier Transform (DFT), where $e(t)$ is the sum of all DFT coefficients of the transformed signal at time t [78]. An accumulated energy $E(t)$ is calculated using a sliding window of 10 samples, or equivalently 10/48

ms under 48 kHz sampling rate:

$$E(t) = \sum_{j=t}^{t+10/48\ ms} e(j) \tag{4.4}$$

The timestamp of a keypress t_p is identified as the timing of a peak point from the accumulated energy:

$$\begin{aligned} t_p = &\arg\max_t E(t) \\ &\text{s.t. } E(t) \geq E(j) \text{ for } t - 10\ ms \leq j \leq t + 90\ ms, \\ &\text{and } E(t) \geq E_\theta \text{ where } E_\theta = \max E(t)/10 \end{aligned} \tag{4.5}$$

The search range is set to 100 ms because the period of keypress is typically about 100 ms [77, 131]. Threshold E_θ is empirically set to be one-tenth of the maximum accumulated energy value. After all key press timestamps are obtained, they are used to partition the corresponding ultrasound signal into six segments, each of which corresponds to entering a keypair during PIN entry. To include keypress time, each segment is extended by 100 ms on both sides. Hereafter, we call each segment a *keypair segment*.

Figure 4.3 shows a beep-sound signal in time domain, where the vertical lines represent identified key press timestamps. Figure 4.4 shows the frequency spectrum of the corresponding ultrasound signal and its partition according to the identified key press timestamps.

4.3.4 Feature Extraction

In *Feature Extraction*, UltraPIN generates a feature vector for each keypair segment, which should reflect the finger movement pattern for typing a keypair during PIN entry.

Since each keypair segment is of low-energy and high-noise, the major challenge in feature extraction is how to enhance signal energy and reduce signal noises.

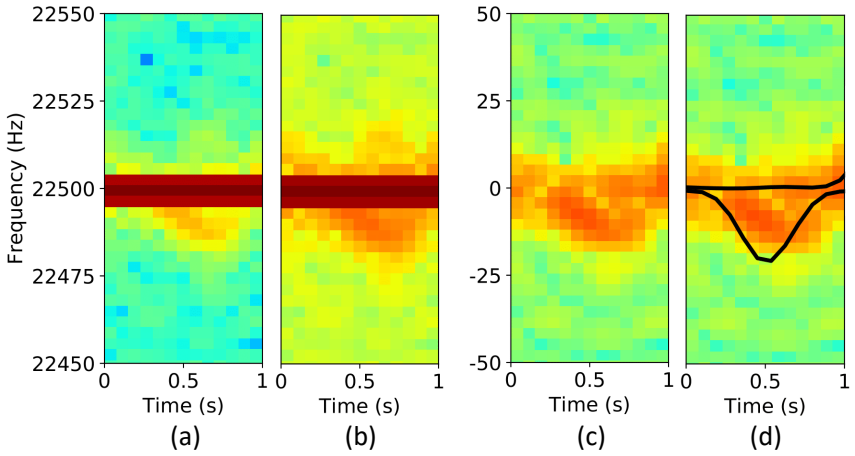


Figure 4.5: (a) Spectrogram of a keypair segment. (b) After signal energy enhancement. (c) After signal carrier removal. (d) After frequency contour extraction.

According to our pilot study, the signal energy of Doppler shifts for each keypair segment in our attacking scenario, where the smartphone-keypad distance is 75 cm, is about one one-hundred-and-fifteenth of the energy of the Doppler shifts measured at smartphone-keypad distance 7.5 cm that is commonly used in the previous finger tracking attack [154]. The signal noises in our attacking scenario originate from nearby acoustic signal sources, acoustic signal reflections by nearby static objects and moving objects, and background noises. To address this challenge, *Feature Extraction* is decomposed into three steps following keypair segment modeling, including signal energy enhancement, signal carrier removal, and frequency contour extraction.

Keypair Segment Modeling. Assuming that there is only one ultrasound signal source (i.e., $A\sin(2\pi ft)$) with frequency f in the ultrasound frequency range $f \pm 50$ Hz, a keypair segment $s(t)$ can be modeled as the sum of all received signals, including the signal received directly from the source, and signals reflected by static objects and moving objects near the source. In general, $s(t)$ can be modeled as follows

$$s(t) = \sum_i a_i \sin\left(2\pi ft + 2\pi f_d t + \theta_i\right) \tag{4.6}$$

In this model, a_i denotes the amplitude of the signal reflected by the i^{th} object; f_d is

the Doppler shift caused by the i^{th} object, which is zero if the object is static, or the signal is received directly from the source; θ_i denotes the relative phase difference between the reflected signal and the source signal. Plugging in Equation (4.3) and setting the starting time as zero for the keypair segment, one has

$$s(t) = \sum_i a_i \sin\left(2\pi f t + 2\pi f \cdot \frac{2}{v_s}\int_0^t v_i(\tau)d\tau + \theta_i\right) \tag{4.7}$$

where $v_i(\tau)$ denotes the radial velocity of the i^{th} object (along the line of sight between smartphone microphone and the object), v_s is the speed of sound in the air, and the term $2\pi f \cdot \frac{2}{v_s}\int_0^t v_i(\tau)d\tau$ indicates Doppler shift.

Signal Energy Enhancement. The signal energy of a keypair segment is enhanced by differentiating it with respect to time

$$\frac{d}{dt}s(t) = \sum_i \left(2\pi a_i f + \frac{4\pi a_i f v_i(t)}{v_s}\right) \cdot \cos\left(2\pi f t + 2\pi f \cdot \frac{2}{v_s}\int_0^t v_i(\tau)d\tau + \theta_i\right) \tag{4.8}$$

The energy (which is directly proportional to the square of the amplitude) of each component signal in the differentiated signal $\frac{d}{dt}s(t)$ is much higher than its counterpart in the original keypair segment. In addition, the energy of each component signal reflected by a moving object is additionally enhanced if its radial velocity is non-zero. The differentiated signal is shown in Figure 4.5(b), while the original keypair segment is shown in Figure 4.5(a). In these figures, the energy of each component signal is color coded: the darker the color, the higher the energy. The brown narrow bands in the center of these figures represent the source signal in combination with the signals reflected by static objects. The yellow signals in Figure 4.5(a) represent the signals reflected by moving objects, including the typing finger during PIN entry. The color of such signals turns red in Figure 4.5(b), indicating that the energy of signal reflecting typing finger movement is significantly enhanced in this process.

Signal Carrier Removal. Next, signal carrier components are removed from the

differentiated signal, where signal carrier components are defined to be the source signal in combination with the signals reflected by static objects, which correspond to the brown narrow band in Figure 4.5(b). To achieve this goal, we multiply the differentiated signal with signal $\sin(2\pi ft)$, yielding $\sin(2\pi ft) \cdot \frac{d}{dt}s(t)$. Applying Equation (4.8), and the product-to-sum formula in trigonometric identities, one has:

$$\begin{aligned}
&\sin(2\pi ft) \cdot \frac{d}{dt}s(t) \\
=&\sum_i \left(2\pi a_i f + \frac{4\pi a_i f v_i(t)}{v_s}\right) \cdot \sin(2\pi ft) \cdot \cos\left(2\pi ft + 2\pi f \cdot \frac{2}{v_s}\int_0^t v_i(\tau)d\tau + \theta_i\right) \\
=&\sum_i \left(\pi a_i f + \frac{2\pi a_i f v_i(t)}{v_s}\right) \cdot \Big[\sin\left(4\pi ft + 2\pi f \cdot \frac{2}{v_s}\int_0^t v_i(\tau)d\tau + \theta_i\right) - \\
&\sin\left(2\pi f \cdot \frac{2}{v_s}\int_0^t v_i(\tau)d\tau + \theta_i\right)\Big]
\end{aligned} \tag{4.9}$$

The above equation shows that the transformed signal is composed of two types of component signals, one with frequency around zero, and the other with frequency around $2f$ (both with term $2\pi f \cdot \frac{2}{v_s}\int_0^t v_i(\tau)d\tau$ for Doppler shift).

Then, UltraPIN applies a band-pass filter in the frequency range ± 50 Hz so as to filter out all component signals of frequency around $2f$, and keep all component signals of frequency around zero. As a result, all signal carrier components are removed. The filtered signal can be written as:

$$-\sum_i \left(\pi a_i f + \frac{2\pi a_i f v_i(t)}{v_s}\right) \cdot \sin\left(2\pi f \cdot \frac{2}{v_s}\int_0^t v_i(\tau)d\tau + \theta_i\right) \tag{4.10}$$

Figure 4.5(c) shows the spectrogram of the filtered signal. Comparing to Figure 4.5(b), the brown narrow band representing all signal carrier components is removed, while the Doppler shifts caused by all surrounding moving objects, including typing finger movement, remain unchanged.

Frequency Contour Extraction. Given a filtered signal, the last step of *Feature Extraction* is to filter out the Doppler shifts from all moving objects except from the typing finger. In most cases, it is reasonable to assume that the typing finger

should have the highest speed among all moving objects surrounding the attacker's smartphone, which leads to the largest frequency shift at each time point in the filtered signal. A frequency contour is thus retrieved from the filtered signal to represent the Doppler shifts from the typing finger only.

Given a signal within a frequency range, a frequency contour of the signal is defined to be a time series of centroid frequencies [152, 98], where each centroid frequency at time t is calculated as the weighted mean of all frequencies present in the signal at time t. The weight of a frequency at time t in the signal is defined to be its amplitude in the signal transformed by Short-Time Fourier Transform (STFT) [112].

Two frequency contours are retrieved from the filtered signal, one within the frequency range from 0 Hz to 50 Hz, and the other within the frequency range from -50 Hz to 0 Hz. Figure 4.5(d) shows the frequency contours that are retrieved from the filtered signal given in Figure 4.5(c).

The feature vector of the corresponding keypair segment, which is the output of *Feature Extraction*, is the concatenation of the two frequency contours retrieved from the filtered signal. In our experiments, it takes on average 3367.92 ms for entering a 6-digit PIN and 561.32 ms for entering each keypair. The feature vector of each keypair segment is a 40-dimensional real-valued vector as 20 samples are evenly taken from each frequency contour.

4.3.5 Movement Modeling

Consider two typical keypads as shown in Figure 4.6. All keypairs, including 100 number-to-number keypairs and 10 number-to-Enter keypairs, are grouped into 41 classes according to finger movement patterns during PIN entry. The keypairs in each class share the same finger movement direction and displacement, indicating that typing these keypairs during PIN entries causes similar feature vectors to be extracted. The 41 classes include 31 number-to-number classes, and 10 number-

Figure 4.6: ATM keypad and POS keypad

to-Enter classes, where each number-to-Enter class consists of one keypair only. Among the 31 number-to-number classes, it is worth noting that one class consists of 10 repeated keypairs, including $0 \rightarrow 0$, $1 \rightarrow 1 \ldots$, and $9 \rightarrow 9$.

For the ease of reference, a label is assigned to each class of keypairs. In particular, label l_0 is assigned to the class of repeated keypairs; each of the other number-to-number classes is associated with label l_i, where $i \in [1, 30]$; in addition, each of the number-to-Enter classes is assigned with label l_{30+i}, where $i \in [1, 10]$. Please refer to the Table 4.1 for the composition of the 41 classes and their labels.

4.3.6 Learning Models

The process of entering a 6-digit PIN consists of typing six number-keys, K_1, $K_2 \ldots$, K_6 followed by an "Enter" key K_7. This process can be decomposed into a sequence of 5 number-to-number keypairs $K_1 \rightarrow K_2$, $K_2 \rightarrow K_3 \ldots$, $K_5 \rightarrow K_6$ followed by a number-to-Enter keypair $K_6 \rightarrow K_7$.

UltraPIN retrieves a feature vector for each keypair from the acoustic signal recorded by a smartphone during a PIN entry. In the case multiple smartphones are used for the PIN inference attack, all feature vectors that are retrieved on these phones are used for classifying each keypair.

To maximize PIN inference accuracy, UltraPIN takes a backward PIN inference strategy to identify the last keypair (i.e., a number-to-Enter keypair) first through a convolutional neural network (CNN) model, and the other keypairs (i.e., number-to-

Table 4.1: Composition of 41 classes of keypairs and their labels.

Label	Keypairs
l_0	$0 \to 0, 1 \to 1, 2 \to 2, 3 \to 3, 4 \to 4, 5 \to 5, 6 \to 6, 7 \to 7, 8 \to 8, 9 \to 9$
l_1	$1 \to 4, 2 \to 5, 3 \to 6, 4 \to 7, 5 \to 8, 6 \to 9, 8 \to 0$
l_2	$4 \to 1, 5 \to 2, 6 \to 3, 7 \to 4, 8 \to 5, 9 \to 6, 0 \to 8$
l_3	$1 \to 2, 2 \to 3, 4 \to 5, 5 \to 6, 7 \to 8, 8 \to 9$
l_4	$2 \to 1, 3 \to 2, 5 \to 4, 6 \to 5, 8 \to 7, 9 \to 8$
l_5	$1 \to 5, 2 \to 6, 4 \to 8, 5 \to 9, 7 \to 0$
l_6	$5 \to 1, 6 \to 2, 6 \to 4, 9 \to 5, 0 \to 7$
l_7	$3 \to 5, 2 \to 4, 6 \to 8, 5 \to 7, 9 \to 0$
l_8	$5 \to 3, 4 \to 2, 8 \to 6, 7 \to 5, 0 \to 9$
l_9	$1 \to 7, 2 \to 8, 3 \to 9, 5 \to 0$
l_{10}	$7 \to 1, 8 \to 2, 9 \to 3, 0 \to 5$
l_{11}	$1 \to 8, 2 \to 9, 4 \to 0$
l_{12}	$8 \to 1, 9 \to 2, 0 \to 4$
l_{13}	$3 \to 8, 7 \to 2, 6 \to 0$
l_{14}	$8 \to 3, 2 \to 7, 0 \to 6$
l_{15}	$1 \to 3, 4 \to 6, 7 \to 9$
l_{16}	$3 \to 1, 6 \to 4, 9 \to 7$
l_{17}	$1 \to 6, 4 \to 9$
l_{18}	$6 \to 1, 9 \to 4$
l_{19}	$3 \to 4, 7 \to 6$
l_{20}	$4 \to 3, 6 \to 7$
l_{21}	$2 \to 0$
l_{22}	$0 \to 2$
l_{23}	$1 \to 9$
l_{24}	$9 \to 1$
l_{25}	$3 \to 7$
l_{26}	$7 \to 3$
l_{27}	$1 \to 0$
l_{28}	$0 \to 1$
l_{29}	$3 \to 0$
l_{30}	$0 \to 3$
l_{31}	$0 \to Enter$
l_{32}	$1 \to Enter$
l_{33}	$2 \to Enter$
l_{34}	$3 \to Enter$
l_{35}	$4 \to Enter$
l_{36}	$5 \to Enter$
l_{37}	$6 \to Enter$
l_{38}	$7 \to Enter$
l_{39}	$8 \to Enter$
l_{40}	$9 \to Enter$

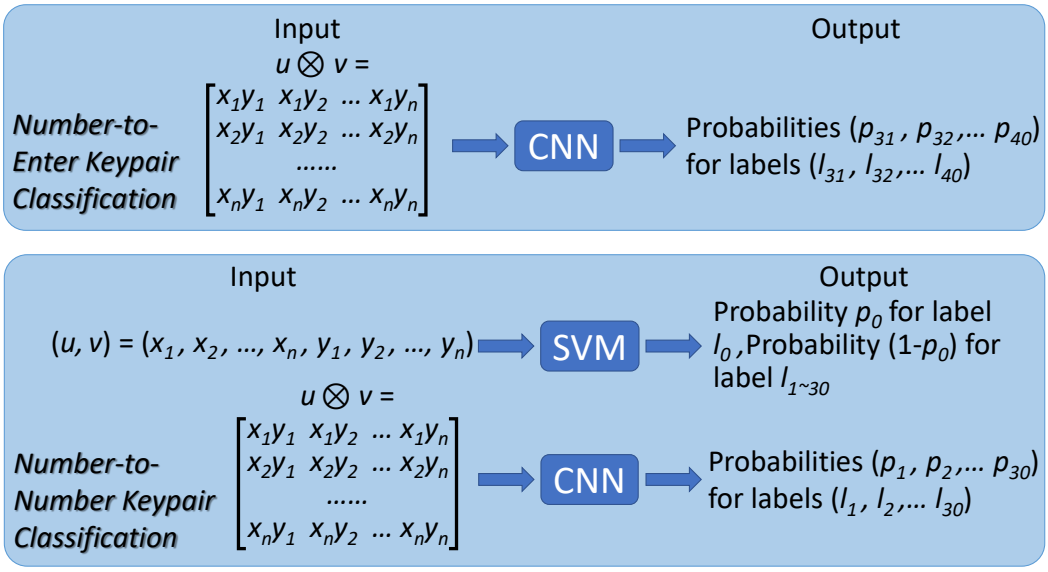


Figure 4.7: Learning models in UltraPIN

number keypairs) in a reverse order through a support vector machine (SVM) and a CNN. Without loss of generality, we assume that two smartphones are used for PIN inference in the rest of this section. Let $u = (x_1, x_2, \ldots x_n)$ denote the feature vector that is retrieved from a keypair segment on one smartphone, and $v = (y_1, y_2, \ldots y_n)$ denote the feature vector that is retrieved from the same keypair segment on the other smartphone ($n = 40$ in our experiments). The learning models in UltraPIN are illustrated in Figure 4.7, and elaborated in the following.

All these learning models are trained in a training phase with training samples, and evaluated in an attacking phase with testing samples. In our experiments, all training samples are generated by two users typing all kinds of keypairs on a numerical keypad, where each keypair is entered by each user repetitively for about $m/2$ times (in our experiments, $m = 200$). In the attacking phase, the performances of the learning models are evaluated with 10 target users entering 6-digit PINs on a numerical keypad with the same layout (of the same size or different size), and these 10 users are *different* from the two users who are involved in the training phase.

Number-To-Enter Keypairs. Given two feature vectors $u = (x_1, x_2, \ldots x_n)$ and $v = (y_1, y_2, \ldots y_n)$ corresponding to a number-to-Enter keypair, UltraPIN trains

a CNN for classifying it among all 10 number-to-Enter keypairs, whose labels are $l_{31}, l_{32}, \ldots l_{40}$. UltraPIN performs this classification first because it is relatively easy to distinguish among all number-to-Enter keypairs. Recall that each number-to-Enter keypair belongs to a single-member class, which means its finger movement pattern is unique; hopefully, it can be classified accurately from its feature vectors which encode the Doppler shifts of a typing finger's movement.

UltraPIN exploits CNN because it works well with data that has a spatial relationship and thus suits for processing the feature vectors that are retrieved from frequency contours. The CNN model we used consists of 8 layers, including one input layer, two convolutional layers, two pooling layers, two dense layers, and one output layer.

The training of CNN is performed with $10 \cdot m$ samples and associated labels, where each of ten labels $l_{31}, l_{32}, \ldots l_{40}$ is associated with m samples. For each entered keypair, UltraPIN derives two n-dimensional feature vectors, u and v, and uses their outer product $u \otimes v$ to train the CNN model to identify the keypair's label. Compared to simple concatenation of these vectors, their outer product encodes their correlations in detail [60, 59, 19], leading to significant improvement in classification accuracy (which is about 23.91% in our experiments).

After the CNN model is trained, it is used in the attacking phase to classify the last keypair segment of a PIN entry. The PIN entry may be performed by any target user different from the training users. For each number-to-Enter keypair entered in the attacking phase, UltraPIN also retrieves two n-dimensional feature vectors, u and v, and takes their outer product $u \otimes v$ as an input to the CNN model, which outputs probabilities $(p_{31}, p_{32}, \ldots p_{40})$ for labels $(l_{31}, l_{32}, \ldots l_{40})$.

Number-To-Number Keypairs. Given two feature vectors $u = (x_1, x_2, \ldots x_n)$ and $v = (y_1, y_2, \ldots y_n)$ corresponding to a number-to-number keypair, UltraPIN first trains a SVM for classifying it between repeated keypair, whose label is l_0, and non-repeated keypair, whose label is $l_{1\sim30}$ representing all labels $\{l_1, l_2, \ldots l_{30}\}$. UltraPIN then trains a CNN model for further classifying it among all 30 non-repeated

keypairs, whose labels are $l_1, l_2, \ldots l_{30}$.

SVM is used to classify each number-to-number keypair into two classes first because it is memory efficient and suitable for binary classifications [69]. It is highly accurate for SVM to classify between repeated keypairs and non-repeated keypairs because their finger movement patterns are clearly different, and so are the Doppler shifts of typing finger's movements which are encoded in the retrieved feature vectors.

The training of SVM is performed with $100m$ samples with associated labels, where $10m$ samples are associated with label l_0; $90m$ samples are associated with label $l_{1\sim30}$; and each sample is the concatenation (u, v) of u, v derived from a keypair entry. The SVM model is trained to classify each keypair's label.

After the SVM model is trained, it is used in the attacking phase to classify the first five keypair segments of a PIN entry. For each such keypair, UltraPIN retrieves two n-dimensional feature vectors, u and v, and takes (u, v) as an input to the SVM model, which outputs probabilities $(p_0, 1 - p_0)$ for labels $(l_0, l_{1\sim30})$.

After each number-to-number keypair is classified into two classes by SVM, it is further classified by a CNN model into 30 classes of non-repeated keypairs. The training of this CNN model is performed with $90m$ samples and 30 associated labels, where $k_i \cdot m$ samples are associated with each label l_i for $i = 1, 2, \ldots 30$; k_i is the number keypairs in the class of keypairs with label l_i (see the Table 4.1 for detail); and each sample is the outer product $u \otimes v$ of u, v derived from a keypair entry. The CNN model is trained to classify each keypair's label.

After the CNN model is trained, it is used in the attacking phase to classify the first five keypair segments of a PIN entry. For each such keypair entered, UltraPIN retrieves two n-dimensional feature vectors, u and v, and takes their outer product $u \otimes v$ as an input to the CNN model, which outputs probabilities $(p_1, p_2, \ldots p_{30})$ for labels $(l_1, l_2, \ldots l_{30})$.

4.3.7 Keypair Combination

In the attacking phase, a target user enters a PIN by typing $K_1 \rightarrow K_2 \rightarrow \ldots K_7$ on a numerical keypad, where $K_1, K_2, \ldots K_6$ are number-keys, and K_7 is the "Enter" key. After applying learning models to classify each keypair segment of the PIN entry, UltraPIN outputs a ranked list of 729 PINs in the default setting, where the size of the ranked list can be adjusted in practice.

In the process of generating the ranked list, UltraPIN starts from classifying the number-to-Enter keypair $K_6 \rightarrow K_7$. The CNN model trained for classifying number-to-Enter keypairs outputs labels $(l_{31}, l_{32}, \ldots l_{40})$ with probabilities $(p_{31}, p_{32}, \ldots p_{40})$, where each of these labels represents a different number-to-Enter keypair, and thus different number-key K_6. UltraPIN sorts these labels according to their probabilities from highest to lowest. Let the sorted labels be denoted as $(l_{s_{31}}, l_{s_{32}}, \ldots l_{s_{40}})$, where $s_{31}, s_{32}, \ldots s_{40} \in \{31, 32, \ldots 40\}$. Let $key(l_{s_i})$ denote the number-key K_6 corresponding to label l_{s_i}, where $i \in \{31, 32, \ldots, 40\}$. In the default setting, UltraPIN takes a list L_6 of three most likely number-keys for K_6:

$$L_6 = \langle key(l_{s_{31}}), key(l_{s_{32}}), key(l_{s_{33}}) \rangle$$

Next, UltraPIN classifies each of the number-to-number keypairs in the reverse order. For each keypair $K_i \rightarrow K_{i+1}$, where $i = 5, 4, \ldots 1$, UltraPIN first classifies it using the trained SVM model, outputting probability p_0 for repeated keypairs of label l_0, and probability $1 - p_0$ for non-repeated ones of label $l_{1\sim31}$. UltraPIN further classifies the keypair using the CNN model trained for classifying non-repeated keypairs, which outputs labels $(l_1, l_2, \ldots l_{30})$ with probabilities $(p_1, p_2, \ldots p_{30})$. Next, UltraPIN sorts all labels $p_0, p_1, \ldots p_{30}$ in two cases:

- Case 1: If $p_0 > 0.5$, put l_0 first, followed by sorting $l_1, l_2, \ldots l_{30}$ according to their probabilities from highest to lowest.

- Case 2: If $p_0 \leq 0.5$, sort $l_1, l_2, \ldots l_{30}$ first according to their probabilities from

highest to lowest, followed by l_0.

The reason UltraPIN sorts all number-to-number keypairs in this way is that the output of SVM is highly accurate for identifying repeated keypairs from non-repeated keypairs, while CNN is used to further classify among different non-repeated keypairs.

For each candidate $K_{i+1} \in L_{i+1}$, where $i = 5, 4, \ldots 1$, there are exactly 10 number-to-number keypairs $K_i \rightarrow K_{i+1}$ (associated with different labels) ending with K_{i+1}. Let these 10 keypairs be denoted as $(l_{s_1}, l_{s_2}, \ldots l_{s_{10}})$ in the sorted list of labels $p_0, p_1, \ldots p_{30}$, where $s_1, s_2, \ldots s_{10} \in \{0, 1, \ldots 30\}$. In the default setting, UltraPIN takes a list L_i of three most likely number-keys for K_i:

$$L_i = \langle key(l_{s_1}), key(l_{s_2}), key(l_{s_3}) \rangle$$

where $i = 5, 4, \ldots 1$.

Finally, UltraPIN generates a ranked list $L = \langle K_1 \rightarrow K_2 \rightarrow \ldots K_6 | K_i \in L_i, i = 1, 2, \ldots 6 \rangle$ for inferring the target PIN, where the PINs in the list are ranked in an anti-lexicographic order; that is, they are ranked according to K_6 in L_6 first, then K_5 in L_5, and so on. This anti-lexicographic order is consistent with the backward PIN inference approach taken by UltraPIN.

In the default setting, the size of each candidate list L_i is set to three for $i = 1, 2, \ldots 6$, yielding 729 PINs in the ranked list L. In our experiments, it is observed that the probabilities of top three labels are usually significantly higher than other labels for inferring each keypair segment. It is thus reasonable to keep the top three candidates in each L_i. On the other hand, most PIN-based user authentication systems set a limit for the number of attempts which each user may make for entering a correct PIN without being locked. Such limits are no greater than 100 in most cases. Therefore, it may not be too meaningful for UltraPIN to generate a lengthy list. Nonetheless, the size of each L_i is easily adjustable for producing the ranked list L of different sizes.

4.4 Performance Evaluation

A series of experiments is conducted to evaluate the effectiveness and robustness of UltraPIN. In this section, we first clarify experimental setting, including keypads, smartphones, environments, and data collection. We then measure the overall effectiveness of UltraPIN in a default setting. To evaluate the robustness of UltraPIN, we report the impacts of keypad layout, keypad size, keypad angle, smartphone quantity and position, smartphone-keypad distance, and experimental environment. Lastly, we present the runtime performance of UltraPIN on commodity devices.

4.4.1 Experimental Setting

Keypads. UltraPIN is evaluated on both ATM keypads and POS keypads as shown in Figure 4.6. These two types of keypads have different layouts: while the Enter key of ATM keypad is located at the right side of the keypad, it appears below all number-keys on POS terminal. The keypads used in our experiments include a $120\ mm \times 90\ mm$ ***ATM keypad*** and a $80\ mm \times 75\ mm$ ***POS keypad***. To examine the robustness of UltraPIN, a *downsized ATM keypad* ($96\ mm \times 72\ mm$) and an *enlarged POS keypad* ($120\ mm \times 112.5\ mm$) are also used in our experiments. Note that extraordinarily large or small keypads are rarely used for PIN entries in practice since extraordinarily large keypads make it easy for shoulder surfing and video-based attacks [150, 145, 115], while extraordinarily small keypads are difficult to use. We thus choose PIN keypads of typical sizes in our experiments.

In order to obtain ground-truth data for evaluating UltraPIN, we implement a keypad simulator on a tablet Surface Pro 4. The simulator can display various keypad layouts on a touch screen; whenever a key is pressed, it records the pressed key as well as the timestamp of the keystroke as ground-truth in our experiments. The keypad simulator produces a 500 Hz beep sound at the time of each keystroke.

Smartphones. Three smartphones are used in each experiment, including a HUAWEI Mate10 device and two HUAWEI P20 devices. All smartphones run on

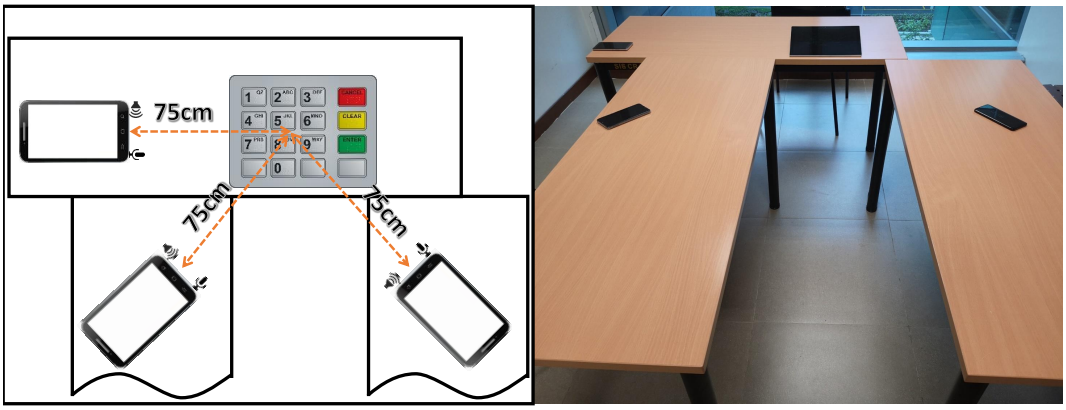


Figure 4.8: In-lab environment

Android 9.0. The microphone and speaker are located at the bottom of each smartphone. UltraPIN is developed as an application installed on these smartphones with the functions of ultrasound emission, sound recording, communications, and so on. In practice, an attacker may activate UltraPIN on one smartphone before a target user enters a PIN, while the other smartphones, if they are used for PIN inference, are activated automatically by the first smartphone via Bluetooth. In our experiments, we automate the activation by connecting UltraPIN on each smartphone to the keypad simulator on the tablet via Wi-Fi. Whenever a user enters a PIN, the keypad simulator sends an indicator to UltraPIN on all three smartphones simultaneously. Receiving the indicator, UltraPIN activates the three smartphones to emit an inaudible ultrasound signal through their speakers at frequency 20.5 kHz, 21.5 kHz, and 22.5 kHz, respectively. Meanwhile, they keep recording acoustic signals with their microphones at sampling rate 48 kHz.

Environments. UltraPIN is evaluated in both in-lab environment and on-site environment. The in-lab environment is a group study room in a university as shown in Figure 4.8. The on-site environment is a public place near a real ATM and a café shop as shown in Figure 4.15, which is exposed to various kinds of noises, such as people's walking, talking, and background music. In our experiments, the tablet is placed flat on a table with its screen/keypad-simulator upside; a user stands facing the tablet, entering a PIN on its keypad simulator; a HUAWEI Mate10 smartphone

is placed flat to the left of the tablet on the same table; two HUAWEI P20 smartphones are placed to the left-rear and right-rear of the tablet (at 7:30 o'clock position and 4:30 o'clock position), respectively, on two other tables of the same height as the first table. The distance between the microphone installed at the bottom of the smartphone (or wireless microphone controlled by the smartphone) and the center of the keypad simulator displayed on the tablet is set to 75 cm in the default setting, and it is changed to 50 cm, 100 cm, and 150 cm in other settings. This distance is referred to as *smartphone-keypad distance* hereafter.

Data Collection. Twelve volunteers are recruited, where two serve as "attackers" training UltraPIN, while the other 10 serve as "victims" in the attacking phase. All of them enter PINs using their right index fingers during data collection.

In particular, each of the two "attackers" is instructed to enter all possible keypairs, including 100 number-to-number keypairs and 10 number-to-Enter keypairs, on both ATM keypad and POS keypad at smartphone-keypad distance 75 cm in the in-lab environment. Each keypair is entered by each participant on each keypad for 100 times. In total, 22000 acoustic signals of keypair segments are collected from the two "attackers" on each smartphone for each keypad in the training phase of UltraPIN.

In the attacking phase, each of the 10 "victims" is instructed to enter 25 6-digit PINs on the ATM keypad at smartphone-keypad distance 75 cm in the in-lab environment. Each PIN is entered by each participant for 10 times. These 25 PINs are shown in Table 4.2, which are randomly selected from the whole PIN space. In total, 2500 acoustic signals of PINs are collected from the 10 "victims" on each smartphone in the default setting. In other settings, 500 acoustic signals of PINs are collected from two "victims" on each smartphone. The success rate of UltraPIN is evaluated across all 25 PINs according to the acoustic signals collected.

Table 4.2: List of PINs used in our experiments.

760973	222233	555553	990872	009666
088886	684032	501347	410886	055333
008853	033645	530271	330117	095653
705333	222253	420381	806205	226633
911182	084953	537473	777777	537802

4.4.2 Overall Effectiveness

In our experiments, no significant difference is observed for the effectiveness of UltraPIN in inferring different PINs and across different users. Therefore, we report the overall effectiveness of UltraPIN for inferring 25 PINs entered by 10 different users in the default setting, where two smartphones are used in both training phase and attacking phase at smartphone-keypad distance 75 cm for PIN entries on the same ATM keypad.

The effectiveness of UltraPIN in the default setting is evaluated by averaging its success rates with top-k candidates over 25 PINs and three combinations of two smartphones (three smartphones are used in each experiment). The success rate with top-k candidates for inferring each PIN is defined as the ratio of the sum of top-k success hits to the total number of attacking cases (100 attacking cases for inferring each PIN as it is entered by 10 users $\times$ 10 times per user). In each attacking case, the top-k success hit is set to one if the top k candidates in the ranked list returned by UltraPIN include the correct PIN entered, and it is set to zero otherwise. For ease of presentation (especially in figures), we round all success rates to two decimal places.

The success rates of UltraPIN with top-3, top-10, top-25, top-50, top-100 candidates in the default setting are 75%, 80%, 81%, 83%, 88%, respectively. We note that the limit of PIN entry attempts is usually set to three times on ATM in most places around the world [64]. The limit may vary on POS terminals, electronic door locks, and other PIN-based user authentication systems. The success rates of UltraPIN indicate that it is highly likely that a PIN attack would be successful; in

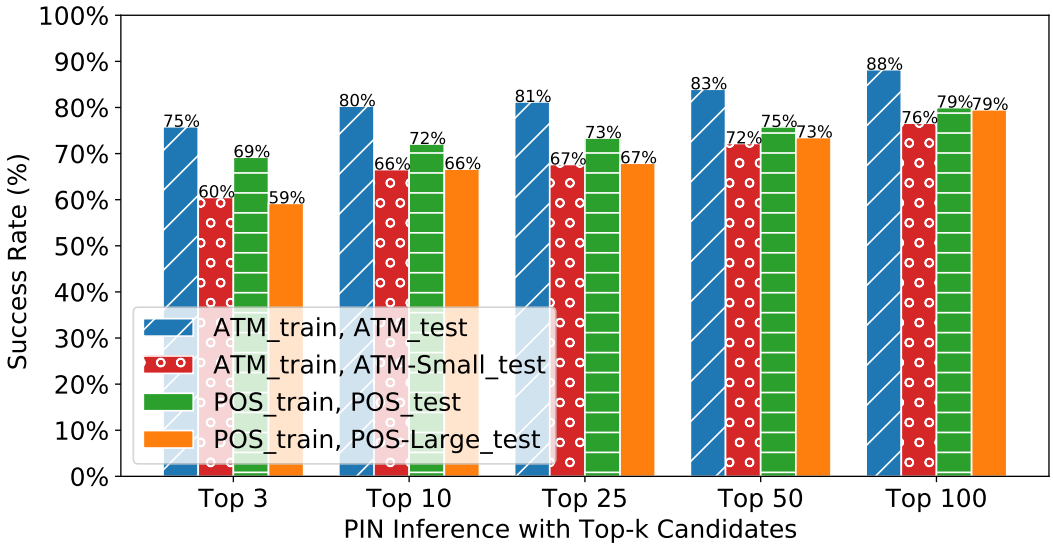


Figure 4.9: Success rates of PIN attacks on various keypads

comparison, random guessing would require more than 750,000 attempts to achieve similar success rates, which is either beyond the limit of PIN entry attempts or takes too much time to complete on any PIN-based user authentication system.

A common trend that can be observed across all of our experiments is that the success rate of UltraPIN with top-k candidates increases as k increases. However, the slop of this trend is not too steep, indicating that UltraPIN is particularly suitable for small k, and thus practical.

4.4.3 Impact of Keypad Layout, Size, and Angle

The keypads of various PIN-based user authentication systems may have different layouts, sizes, or be installed at different angles. The impact of keypad layout[1] to UltraPIN is measured with two different layouts (see Figure 4.6), including ATM keypad ($120\ mm \times 90\ mm$) and POS keypad ($80\ mm \times 75\ mm$). Our evaluation results are shown in Figure 4.9, where legend "ATM_train, ATM_test" means that UltraPIN is trained and tested on ATM keypad; and "POS_train, POS_test" means that UltraPIN is trained and tested on POS keypad. While the success rates of UltraPIN on POS keypad are lower than on ATM keypad, their differences are within

[1] A keypad's layout is defined as the specific arrangement of the keys on the keypad, as well as their relative positioning and sizes.

10%, indicating that UltraPIN is still highly effective. The weaker performance on POS keypad is probably due to its smaller size, and thus weaker Doppler shifts caused by typing finger movements comparing to ATM keypad.

Knowing the keypad layout of a target system before launching a PIN inference attack, an attacker can always train UltraPIN with a similar keypad layout. Nonetheless, many PIN keypads share similar layouts but in different sizes. It may not be convenient for an attacker to train UltraPIN repetitively for similar keypads in various sizes. Figure 4.9 shows that UltraPIN is still effective to infer PIN entries on resized keypads without retraining, where legend "ATM_train, ATM_Small_test" means that UltraPIN is trained with ATM keypad, and tested with downsized ATM keypad ($96\ mm \times 72\ mm$); and "POS_train, POS_Large_test" means that UltraPIN is trained with POS keypad, and tested with enlarged POS keypad ($120\ mm \times 112.5\ mm$). The success rates drop by about 10% – 15% in the ATM case, and within 10% in the POS case. It seems better for an attacker to train UltraPIN on a large keypad and test it on a small keypad instead of vice versa.

In practice, a PIN keypad may be installed at various angles between keypad surface and ground surface. While ATM keypads and keypads of electronic door locks are usually installed at 0° and 90°, respectively, POS terminals may be installed at various angles between 0° and 90°. In our default setting, the keypad is flat at angle 0°, and all smartphones are coplanar with the keypad's surface. Recall that UltraPIN infers PINs according to the Doppler shifts received by each microphone, which are determined by a typing finger's radial velocity along the line of sight between the typing finger and the microphone (see Equation 4.3). UltraPIN's performance is not affected by any change in keypad angle as long as all microphones (either smartphone microphones or wireless microphones controlled by smartphones) are coplanar with the keypad's surface, and their relative positioning remains unchanged.

Nonetheless, we need to consider the case in which a recording microphone is not coplanar with the keypad's surface in the attacking phase. Figure 4.10 shows the performance of UltraPIN, which is trained on POS keypad at 0° keypad angle,

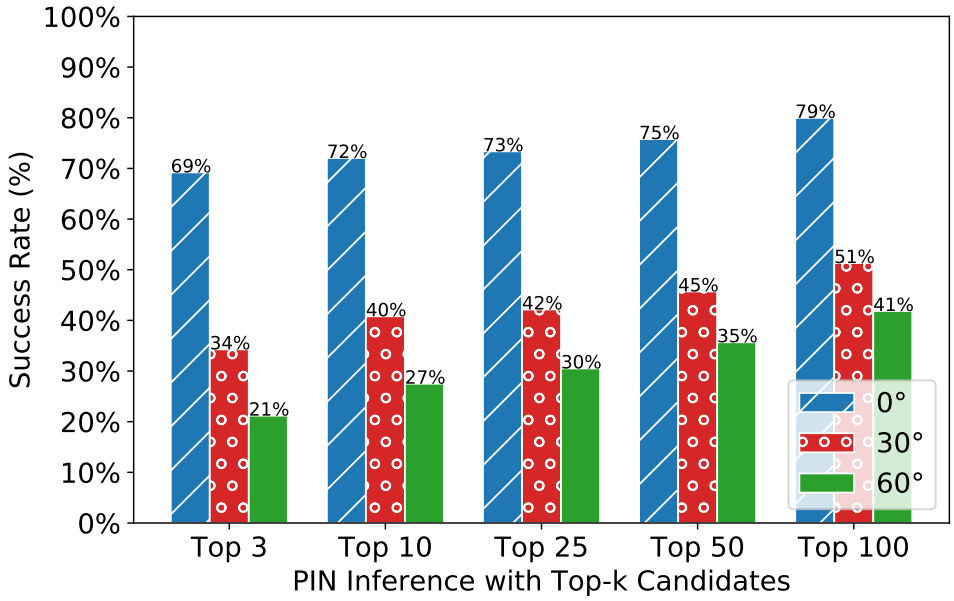


Figure 4.10: Success rates of PIN attacks at various keypad angles

for attacking PIN entries on POS keypad at different keypad angles, including $0°$, $30°$, and $60°$. Even if the success rates of UltraPIN drop considerably at $30°$ and $60°$ compared to $0°$, they are still within the same order of magnitude.

We note that POS terminals are seldom installed at $90°$ as keypads of electronic door locks. In this latter case, it may raise an alert to a target user entering a PIN code while an attacker stands nearby launching a PIN inference attack with smartphones. To avoid this alert, an attacker may attach hidden wireless speakers and wireless microphones (e.g., AirPods [5], HUAWEI FreeBuds [62]) to the victim's door surface (it is equivalent to the $0°$ setting since the speakers and microphones are coplanar with the door keypad) or close to the door surface for better performance. The attacker may stay 10 meters away from the victim, using UltraPIN on his/her smartphones to control the wireless speakers and microphones via Bluetooth for launching a PIN inference attack. Since the layouts of electronic lock keypad are not as standardized as ATM and POS keypads, we do not evaluate UltraPIN on numerous electronic door keypads. While the principle of UltraPIN attacks remains effective, UltraPIN needs to be adapted to various layouts of electronic door locks in practice.

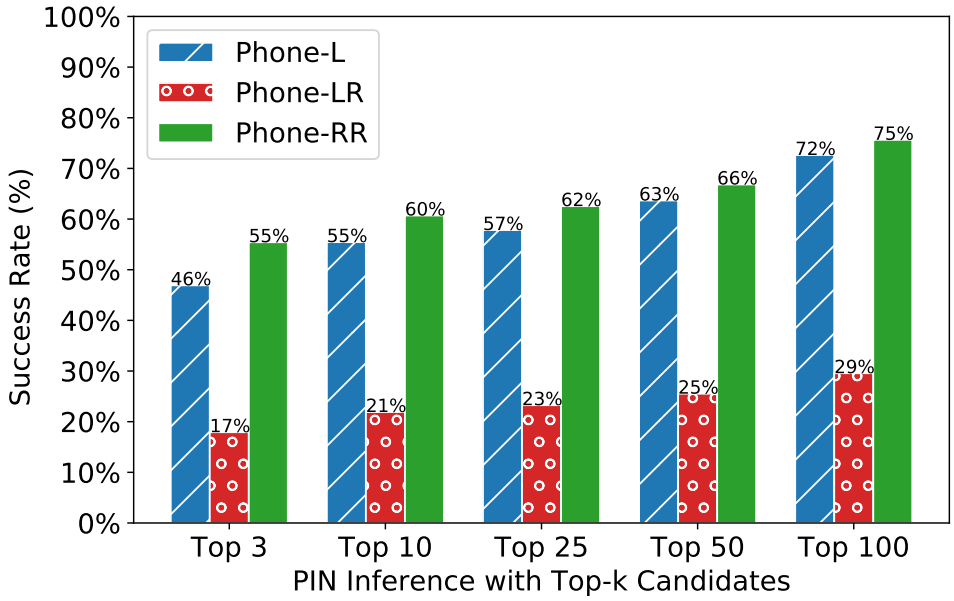


Figure 4.11: Success rates of PIN attacks with one smartphone

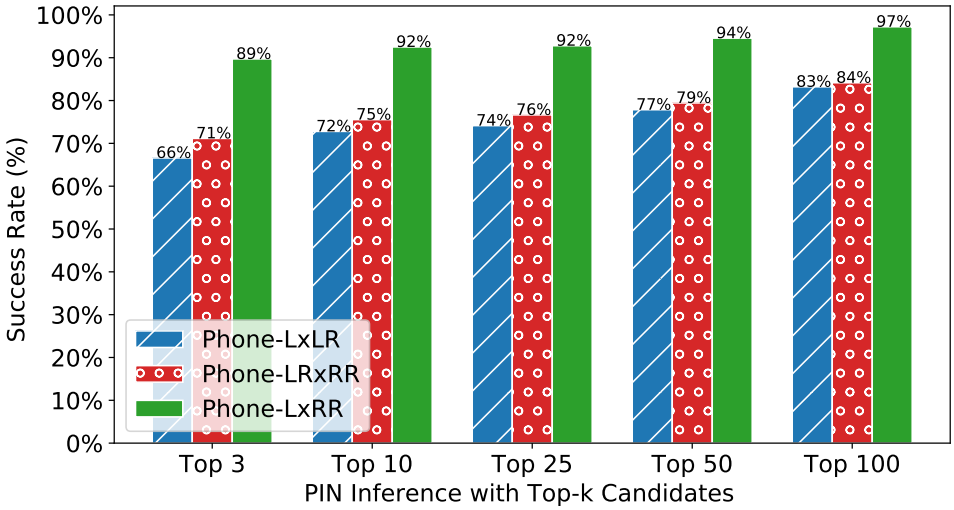


Figure 4.12: Success rates of PIN attacks with two smartphones

4.4.4 Impact of Smartphone Quantity and Position

UltraPIN with a single smartphone is able to sense the radial velocity of a finger' movement in one direction (along the line of sight between the finger and the smartphone). Intuitively, if an attacker uses more smartphones in both training and attacking phases, UltraPIN may achieve higher accuracy. To measure the impact of smartphone quantity, three smartphones are placed at a distance of 75 cm near the keypad as illustrated in Figure 4.8. For ease of reference, let "Phone-L" denote

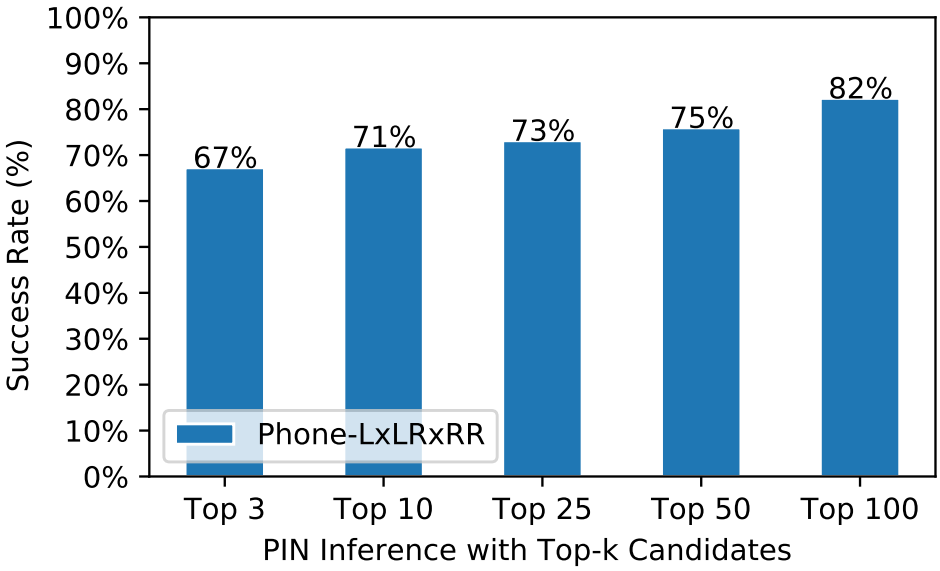


Figure 4.13: Success rates of PIN attacks with three smartphones

the smartphone on the left side, "Phone-LR" denote the smartphone on the left-rear side, and "Phone-RR" denote the smartphone on the right-rear side. UltraPIN is trained and tested with various smartphones and their combinations, including single phones (denoted by Phone-L, Phone-LR, and Phone-RR), two-phone combinations (denoted by Phone-LxLR, Phone-LRxRR, and Phone-LxRR), and three phones (denoted by Phone-LxLRxRR).

Figures 4.11, 4.12, and 4.13 demonstrate the success rates of UltraPIN with different smartphone combinations. The highest success rates are achieved with two smartphones, with one placed on the left side and the other on the right-rear side. Beyond our expectation, the success rates of UltraPIN with three smartphones, Phone-LxLRxRR, are similar to or worse than the success rates in the two-smartphone cases. It would be sufficient to use at most two smartphones to launch a PIN inference attack.

It is also observed that the success rates of UltraPIN vary if smartphones are placed at different positions. A single phone placed on the left-rear side (Phone-LR) performs considerably lower than the other two positions (left and right-rear) probably because it is most difficult to sense a victim's right hand's typing (in our experiments, all users use their right hands for entering PINs) from the left-rear side

of the victim's body.

4.4.5 Impact of Smartphone-Keypad Distance

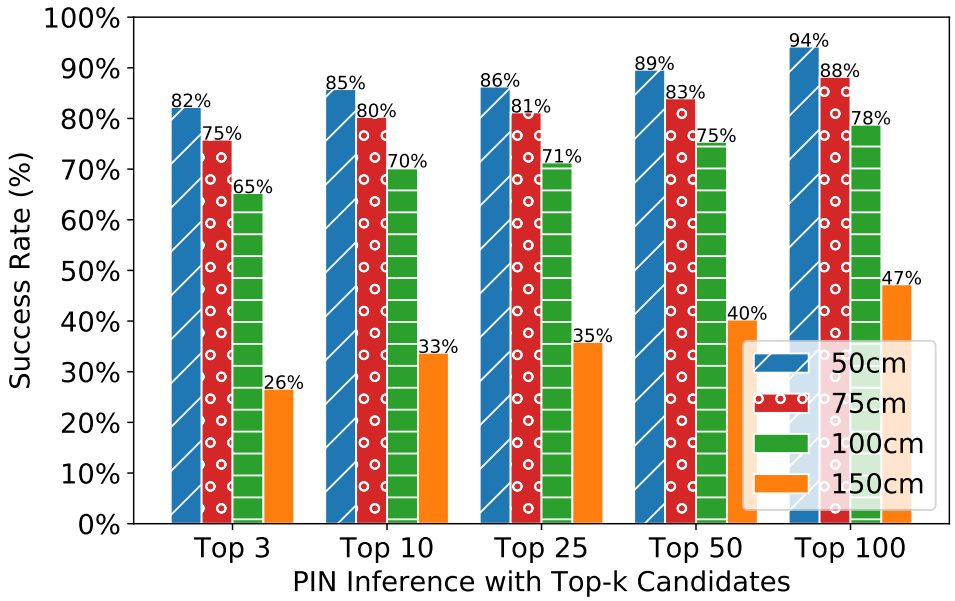


Figure 4.14: Success rates of PIN attacks at various smartphone-keypad distances

While UltraPIN is trained at smartphone-keypad distance 75 cm in our default setting, the success rates of it in the attacking phase are evaluated at different smartphone-keypad distances, including 50 cm, 75 cm, 100 cm, and 150 cm. Figure 4.14 shows that UltraPIN performs better at 50 cm than 75 cm even it is trained at 75 cm. With increasing smartphone-keypad distance, UltraPIN's performance drops, but not too drastically.

4.4.6 Impact of Experimental Environment

The performance of UltraPIN is also evaluated in the on-site environment (see Figure 4.15) and is compared to its performance in the in-lab environment. In both cases, UltraPIN is trained in the in-lab environment in the default setting (at 75 cm smartphone-keypad distance). Considering that a one-meter waiting line may be drawn on the ground in front of an ATM in certain areas, an attacker is required to hold smartphones one meter away from the victim's keypad in the attacking phase.

As it is shown in Figure 4.16, the on-site environment introduces only about 5% performance loss due to environmental noises (e.g., people walking, talking, and background music), indicating that most of such noises are effectively filtered out in the feature extraction phase of UltraPIN.

Figure 4.15: On-site environment

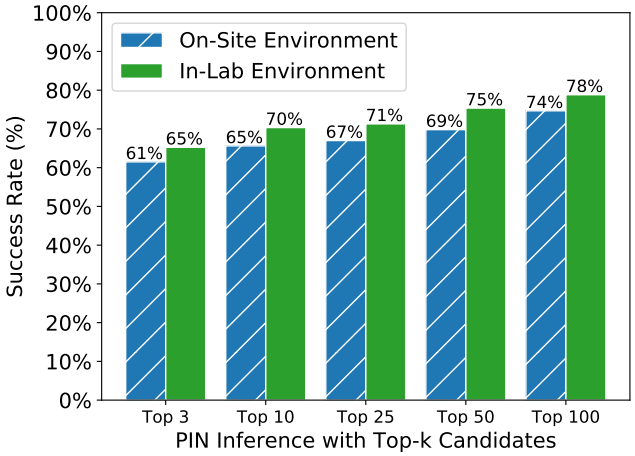


Figure 4.16: Success rates of PIN attacks in on-site experiments

4.4.7 Runtime Performance of UltraPIN on Commodity Devices

The runtime performance of UltraPIN is dominated by its learning models, including a SVM model and two CNN models. These learning models are trained on a commodity personal computer (PC) in the training phase, and run on a commodity smartphone in the attacking phase.

The training of three learning models is performed on a Lenovo Y7000P laptop with Intel i7-8750H CPU and NVIDIA GeForce GTX 1060 GPU. All models are written in Python; machine learning library scikit-learn is used for training the SVM model, while TensorFlow deep learning framework is used for training the two CNN models. In the default setting, it takes 2.5 hours to train the SVM model, 3 hours to train the CNN model for number-to-Enter keypair classification, and 5.25 hours to train the CNN model for number-to-number keypair classification. The runtime performance in the training phase is measured on average among six experiments on two different keypads and with three combinations of two smartphones.

After training, the parameters of all models are saved to local files and compiled into mobile applications for PIN inference attacks. These applications are written in Java, using scikit-learn porter [91] for SVM classification and TensorFlow Lite [52] for CNN model classification. Running on a HUAWEI P20 smartphone with Kirin 970 CPU and 4 GB RAM, it takes $16\ ms$ for SVM classification, $3.6\ ms$ for CNN number-to-Enter keypair classification, and $10.8\ ms$ for CNN number-to-number keypair classification. Overall, it takes about $137.6\ ms$ on average for UltraPIN to infer each PIN entry (including classifying five number-to-number keypairs and one number-to-Enter keypair) in the attacking phase. The memory requirement for running UltraPIN on the smartphone is below 207 MB.

4.5 Discussions

4.5.1 Limitations

Twelve volunteers, including 11 students and one faculty member, were recruited from a university for entering PINs in our experiments. Among different populations, UltraPIN's performance may vary. Although the ecological validity of our experiment is limited, we believe that the qualitative facts in our research remain true.

In our experiments, all participants were instructed to enter PINs using a single finger. According to our survey conducted in 2018 among 544 users across three countries [79], 344 users (63.2%) prefer using a single finger for PIN entry. If some users enter PINs with multiple fingers, it is more difficult for UltraPIN to capture their finger movement patterns unless UltraPIN is trained with similar typing styles.

Wrong typing is not considered in our experiments. If a user types wrongly, most ATM, POS, and electronic door locks provide a "Cancel," "Clear," or "$\times$" button for uses to terminate the current PIN entry and start a new one. UltraPIN can be easily extended to cover this case: its *Keypair Segmentation* component may

simply output the last six keypair segments.

UltraPIN relies on beep sounds that are produced during a PIN entry to segment acoustic signals into keypair segments. If such peep sounds are not produced or cannot be captured in rare cases, the accuracy of UltraPIN may decline. It remains interesting to train a deep learning model to learn how to segment acoustic signals into keypair segments without beep sounds. We leave this as a future work direction.

4.5.2 A Countermeasure

An effective defense to UltraPIN attacks is to randomize the layout of the keypad for each PIN entry such that UltraPIN cannot map each key-position to a PIN-digit for PIN inference. This requires that soft keypad be implemented on touchscreen, and touchscreen be adopted by PIN-based user authentication system. While it is possible to replace the existing hard keypads with soft, randomized keypads at a large scale, the cost factor is not negligible. Another concern is user experience. According to a usability evaluation [107], the average completion time for typing each PIN digit on a randomized keypad is 35.04% longer than typing on a conventional keypad.

Chapter 5

Typing-Proof: Usable, Secure and Low-Cost Two-Factor Authentication Based on Keystroke Timings

5.1 Introduction

This chapter designs a usable, secure and low-cost two-factor authentication mechanism which eliminates user-phone interaction in most cases and immune to the existing attacks to recent 2FA mechanisms. Two-factor authentication (2FA) systems are pervasively used for protecting login attempts and online transactions. They require users to provide two separate pieces of credentials for user authentication. The first factor (credential) is typically a knowledge factor, where passwords or PINs serve as something that only legitimate users should know. The second factor (credential) is typically a possession factor, where hardware tokens (e.g., ID cards, USB tokens, and wireless tags) or software tokens (e.g., smart-phones or smart-watches) serve as something that only legitimate users should possess.

Hardware token based 2FA introduces extra burden to users since they typically require a user to carry and interact with a hardware token. A one-time code displayed on the hardware token should be submitted by the user to a server for user

authentication. Besides its usability issue, a service provider must manufacture a number of hardware tokens and distribute them to all customers, which is expensive (e.g., $60 per token [2]) if the customer base is large. A hardware token usually has a lifetime around 3 years; therefore a service provider needs to distribute new tokens to each customer every 3 years, which also adds to the costs.

In recent years, due to the pervasive use of phones, SMS-based 2FA becomes more popular. After a user's first factor is verified by a service provider, a verification code is sent to the user via SMS. The user needs to use this code to prove the possession of the second authentication factor. This solution relaxes the requirement on additional hardware but still requires users to interact with their phones so as to read and input verification codes during authentication processes. In addition, a service provider bears a significant cost for sending verification codes via SMS to users' phones to complete all authentication sessions in daily operations.

To eliminate the user-phone interactions, Karapanos et al. proposed Sound-Proof [67] recently which enables a server to verify a user's second factor by matching two pieces of ambient sounds recorded respectively by the user's phone and by the browser in a login computer during a short period of time (5 seconds in [67]) right after the server verifies the user's first factor (i.e., username and password submitted to the server via the browser in the login computer) for each authentication session. However, it has usability limitations such that it is not designed to be used in quiet environments and it cannot work when the login computer has not been equipped with a microphone or the browser in the login computer does not support audio recording. In addition, this solution is vulnerable to certain practical attacks, including sound-danger attack [153] and co-located attack [67].

In this chapter, we propose Typing-Proof, a usable, secure and low-cost two-factor authentication system. In Typing-Proof, the second factor is the proximity of a user's phone to the computer being used to log in to an authentication server. A user needs to place his/her registered phone near the login computer. After the user passes the first-factor authentication using a browser in the login computer,

he/she is required to type a random code by the user's choice (i.e., a sequence of any keys) on the computer's keyboard. During the typing, the browser in the login computer records the keystroke timing sequence (i.e., a sequence of all keystrokes' timestamps) by JavaScript and the user's phone records the keystroke sound. After finishing the typing, the keystroke timing sequence is sent to the user's phone via the server. Then the user's phone compares the keystroke timing sequence with the recorded keystroke sound, and approves the second factor if they "match", meaning that the registered phone is near the computer. If this second-factor authentication fails, Typing-Proof provides a backup solution where the user's phone displays the random code through an application. The user checks whether it matches the code typed and displayed on the browser, and presses an "Approve" or "Deny" button accordingly.

Typing-Proof is user-friendly. It requires no user-phone interactions in most cases, and one-button press in the backup case. Typing-Proof works in any environment, even in a noisy place. It can be easily deployed since it is compatible with all major browsers, login computers, and smartphones, and does not require any additional plug-ins or external hardware to be used.

Typing-Proof is practically secure. In particular, Typing-Proof is more secure than Sound-Proof since it is immune to sound-danger attack and co-located attack. In sound-danger attack, an attacker deliberately makes a victim's registered phone to produce particular sounds. However, it is difficult to simulate keystroke sound on a victim's side remotely and such simulation can be easily blocked in Typing-Proof. In co-located attack, an attacker logins to a victim's account using 2FA in the same environment with the victim. It is still difficult for the victim's phone to capture the attacker's keystroke sound in Typing-Proof, except that the distance between the attacker's login computer and the victim's phone is sufficiently short (e.g., within 100cm), which may raise the victim's awareness.

Typing-Proof incurs significant lower costs compared to other solutions, including Sound-Proof, hardware token based 2FA, and SMS-based 2FA. In particular,

Typing-Proof lowers the charges of data transfer compared to Sound-Proof. Only a keystroke timing sequence and a random code (around 250 bytes) need to be sent from a user's login computer to the user's phone in Typing-Proof during the second-factor authentication; this is smaller in size than the audio signal transmitted in Sound-Proof. For hardware token based 2FA and SMS-based 2FA, they do not involve any data transfer cost, but they cost much more on manufacturing hardware tokens and sending short messages, respectively.

We have implemented a prototype of Typing-Proof for Android devices. Compared to password-only authentication mechanisms, Typing-Proof takes around 4.3 seconds longer for completing user authentication on average. This additional time is not only substantially shorter than the time overhead of 2FA mechanisms based on verification codes (roughly 10.4 seconds longer than the password-only solution) but also shorter than Sound-Proof mechanisms (around 5 seconds longer than the password-only solution). A user study we conducted shows that users prefer Typing-Proof over both SMS-based 2FA [50] and Sound-Proof [67].

5.2 Assumptions and Goals

System Model. Our two-factor authentication system requires two devices – a computer and a phone on the user side, and an authentication server on the server side (hereinafter referred as the "server"). The computer is used to login to the user's account through a web browser application (hereinafter referred as the "login computer"). The phone is installed with a "Typing-Proof" application which is bound to the user's account (hereinafter referred as the "registered phone"). Note that the login computer and the registered phone can be the same physical device when a user logs in from the browser on his/her registered phone.

During an authentication procedure, a user points his browser to the server's webpage and enters his/her username and password. The server verifies the user's credential and challenges the user to prove the second authentication factor.

Threat Model. We assume that an adversary has obtained a victim's username and password. This assumption is reasonable since password database suffers from various cyber-attacks. Many companies, including Dropbox [68], LinkedIn [58], Yahoo [117], are targets of password database leakage recently. An adversary is successful in impersonating a victim user if the adversary is able to convince the server that he/she also holds the second authentication factor of the victim.

We further assume that the adversary cannot compromise the victim's registered phone. In other words, the victim's registered phone is trusted by the server. This assumption is also shared by other two-factor authentication systems based on software tokens.

We do not consider Man-In-The-Middle attack. Client-web authentication cannot fully prevent such attacks even if web applications employ HTTPS communications [66]. We leave out active phishing attack where attackers trick users to visit phishing websites and relay stolen credentials to legitimate websites in real-time. Such attacks can be defended using anti-phishing technologies [46, 121].

Design Goals of 2FA Mechanism.

- *Usability.* A 2FA mechanism should be easy to learn and efficient to use. In most cases, users should not be asked to interact with their phones. In particular, a registered phone (i.e., software token) should work well even when the phone is locked or the authentication application in it runs in its background. In addition, the second authentication factor should require no memory demand for users.

- *Security.* A 2FA mechanism should be secure under a general threat model shared by other 2FA mechanisms. It should be resilient to guessing. The second factor should be independent with the first factor. This implies that the leak of any single factor should not affect the security of the other factor.

- *Low-cost.* A 2FA mechanism should not consume too much computing resources, especially for the authentication applications installed on registered phones. The total costs using 2FA, including the costs at the server's end (e.g., SMS fee and

data transfer cost) and at the user's end (e.g., data transfer cost), should be low or negligible.

5.3 Typing-Proof

In this section, we introduce Typing-Proof in detail. Our solution uses password as the first factor and the proximity of a user's registered phone to a login computer as the second factor. The proximity of the two devices is determined by comparing the keystroke timing sequence recorded by the login computer for the user's typing of a random code on the computer with the keystroke sound recorded by the user's registered phone which is placed closed to the login computer. We analyze that our approach is usable, secure and low-cost.

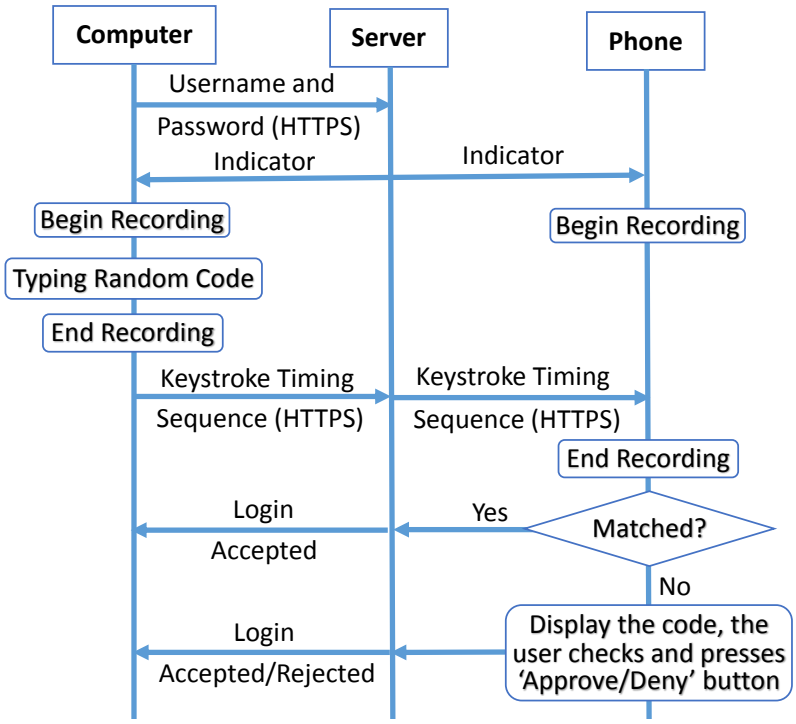


Figure 5.1: The overview of Typing-Proof two-factor authentication login procedure.

5.3.1 Enrollment and Login

Similar to other 2FA mechanisms based on software tokens, Typing-Proof requires a Typing-Proof application to be installed on a user's smartphone as a software token and be bound to his/her account on the server. This is a one-time operation, which can be carried out using similar existing techniques to enroll software tokens (e.g., [50, 67]).

Figure 5.1 shows an overview of the login procedure, in which a user places his/her registered phone near a login computer and uses the login computer to login to a server. In the login procedure, a user points the browser to the URL of the server on a login computer and enters his/her username and password. If both username and password are correct, the server sends two separate indicators to the login computer and the user's registered phone, respectively for activating the second authentication process. Upon receiving an indicator, the browser pops up an input box for the user to type a random code by the user's choice. At almost the same time, the registered phone receives another indicator and starts recording audio through its embedded microphone. During the user's typing, the browser records a timing sequence of the user's keystrokes using JavaScript. After finishing typing, the browser stops recording and sends the random code as well as the keystroke timing sequence to the registered phone through the server. When receiving the random code and the keystroke timing sequence, the registered phone stops recording and compares the keystroke timing sequence with the recorded audio signal for the second-factor authentication. In particular, it calculates a similarity score between the two. If and only if the similarity score is above a threshold τ_{sim}, the Typing-Proof application in the registered phone concludes that it is close to the login computer and informs the server that this login attempt is legitimate. Note that this second-factor authentication process is automatically carried out without any user-phone interactions.

When a user logins to his/her account using Typing-Proof in an abnormal environment, such as the keyboard is soundless, or the environment is too noisy, the

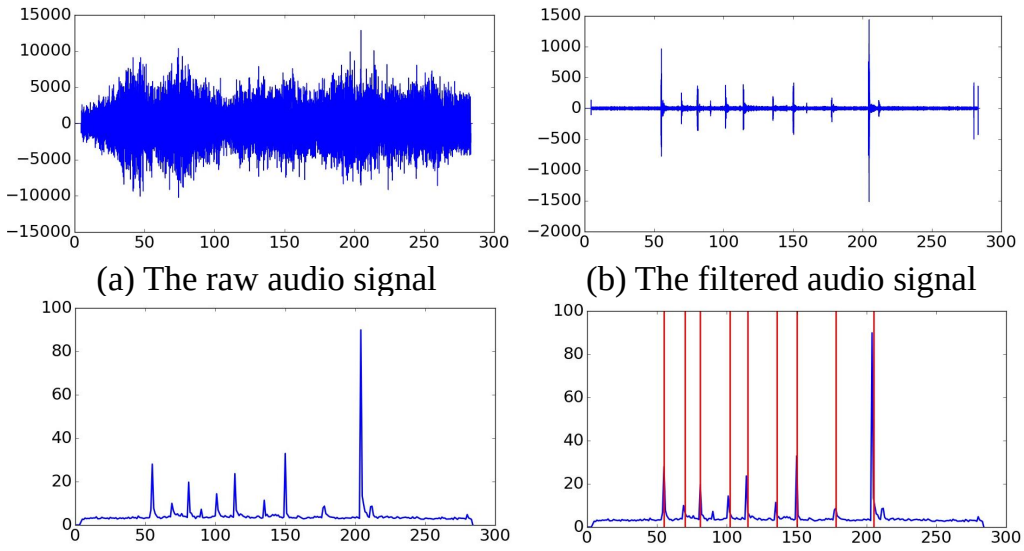
automatic second-factor authentication may fail. Typing-Proof provides a backup solution where the registered phone displays the random code and 'Approve/Deny' buttons. The user presses an 'Approve' and 'Deny' button to manually accept or reject the login attempt after checking whether the random code displayed on the registered phone is the same as the one typed and displayed on the login computer. Our approach is specific to our proposed scheme and securer than the existing one-button authentication solutions described in Section 2.2 since it is immune to the synchronized login attack: the user can easily identify his/her login request by checking the random code displayed on the registered phone.

For the security reason that an attacker may infer the password from keystroke timing information [119, 151], the keystroke timing sequence is recorded from typing a random code instead of from password entry. In addition, all browser-server and phone-server communications over the Internet are transmitted via HTTPS and the server does not need to store any keystroke information (i.e., keystroke timing sequence, random code, and keystroke audio sample).

5.3.2 Similarity Score

The Typing-Proof application on a registered phone computes a similarity score between a keystroke timing sequence and a piece of audio signal in three main steps including noise reduction, energy level extraction, and cross-correlation.

Noise Reduction. The environment where a user conducts his/her authentication may have various kinds of noise, such as other users' typing on their computers, people's talking, and background music. A user's keystroke sound using Typing-Proof may be covered by such noise. We observe that the keystroke sound mainly lies in the frequencies higher than 15,000Hz. Therefore, we utilize a high pass filter to remove the noise below. Figure 5.2(a) and Figure 5.2(b) show a raw audio sample recorded in a Starbucks café and the corresponding filtered audio sample. We have evaluated this step over a number of samples, and it turns out that keystroke signals



(a) The raw audio signal

(b) The filtered audio signal

(c) The energy level signal

(d) The energy level signal with corresponding keystroke timing vertical

Figure 5.2: Example of an audio signal which is recorded in a café.

can be correctly 'sanitized' in most cases.

Energy Level Extraction. Similar to previous study [9, 18, 155, 131], we observe that the acoustic signal of one keystroke usually involves three peaks: touch peak, hit peak, and release peak. According to our experiments, when a registered phone is placed more than 50cm away from the keyboard of a login computer or the environment is too noisy, the touch peak and the release peak may become inconspicuous while the hit peak remains clear. We thus use the hit peak to serve as a landmark of a keystroke. To highlight the hit peak, we transform the signal sequence into energy levels using time windows. Particularly, we calculate the energy levels of a keystroke sound using windowed discrete Fourier transform (DFT) and take the sum of all FFT coefficients as its energy. Figure 5.2(c) shows the energy-level signal corresponding to the filtered audio signal that is shown in Figure 5.2(b).

Cross-correlation. We choose cross-correlation as our similarity metrics. Cross-correlation is a standard measure of similarity between two time series. We use x to denote the energy-level signal converted from the audio signal recorded by a registered phone and use k to denote the keystroke timing sequence recorded by the login computer. First, we transform the keystroke timing sequence k into a pulse

sequence y:

$$y[t] = \begin{cases} 0 & \text{if } t \text{ is not a element in } k \\ 1 & \text{if } t \text{ is a element in } k \end{cases} \tag{5.1}$$

where t ranges from 0 to the length of the energy-level signal x. Given two time series x and y, we then let:

$$CC_{x,y}(l) = \sum_{i=0}^{n-1} x[i] \cdot y[i-l] \tag{5.2}$$

This is a sliding dot-product of the two time series, where y is shifted by l samples over x. To accommodate different amplitudes of the two signals, the cross correlation is normalized as:

$$CC'_{x,y}(l) = \frac{CC_{x,y}(l)}{\sqrt{CC_{x,x}(0) \cdot CC_{y,y}(0)}} \tag{5.3}$$

where $CC_{x,x}(0)$ and $CC_{y,y}(0)$ is the auto-correlation. The cross-correlation is maximized at the offset l where the two time series are most similar. We define $max_l(CC'_{x,y})$ to be the similarity score between the keystroke timing sequence and the audio signal where l is bounded between 0 and t_{max}. Figure 5.2(d) shows a plot of the keystroke timing sequence and the audio signal where the two time series are matched best. The red vertical lines in the Figure 5.2(d) denote the timestamps of all keystrokes.

5.3.3 Usability Analysis

Typing-Proof requires users to place their phones near the login computer but it does not require users to interact with their phones in most cases. Users need not take any action to launch the Typing-Proof application on their registered phones before they conduct authentications. Even the Typing-Proof application is running in the background, or the registered phone is locked, the Typing-Proof application can still be activated to record keystroke sound in the second-factor authentication

process as long as the registered phone is connected to a network.

The usability of Typing-Proof is slightly lower than the user authentication with password only. In most cases, Typing-Proof requires no user-phone interactions for 2FA. It takes 4.3 seconds on average without user-phone interactions, and it takes additional 7.1 seconds on average for using the backup solution in case it is triggered.

The usability of Typing-Proof is significantly higher than Sound-Proof. First, Sound-Proof is not designed to work in a quiet environment while Typing-Proof works well in various environments. Second, Sound-Proof requires a login computer to equip with a microphone for audio recording while Typing-Proof does not require any additional hardware. Third, many browsers (e.g., Internet Explorer and Safari [87]) do not support audio recording. In comparison, Typing-Proof can work with all major browsers since they all support keydown event API. On the other hand, we acknowledge that Sound-Proof is convenient to use since it does not require random typing.

The usability of Typing-Proof is also significantly higher than hardware token based 2FA and SMS-based 2FA. Typing-Proof does not require users to remember anything. In the backup solution, users need to check whether the random code displayed on the registered phone is the same as the one typed and displayed on the login computer. According to a quantitative usability analysis framework [147], the cognitive workload of this comparison can be calculated by $0.4077 \cdot \lceil x/4 \rceil = 0.101925 \cdot x$ (seconds), where x is the length of random code. The cognitive workload is about 1.02 seconds for $x = 10$. However, for hardware token based 2FA and SMS-based 2FA, a user needs to memorize the verification code (in most case, the length of the code is 6) temporarily and then inputs it into the browser on a login computer. The memory demand in these solutions can be calculated by $\lceil 6/4 \rceil / 29.6\% = 6.76$ (seconds) when the length of a verification code is 6 [147]. Therefore, Typing-Proof takes a shorter time for cognitive operations than hardware token based 2FA and SMS-based 2FA.

5.3.4 Cost Analysis

The costs for Typing-Proof stem from data transfer. During the second-factor authentication of Typing-Proof, a random code and its corresponding keystroke timing sequence (around 250 bytes) are sent from a login computer to a registered phone via server. In comparison, Sound-Proof transmits a piece of audio signal (around 250k bytes) whose data size is about 1000 times larger than that in Typing-Proof. Thus, Typing-Proof costs significant less than Sound-Proof for data transfer. For hardware token based 2FA and SMS-based 2FA, they do not involve any data transfer cost (e.g., $0.09 per GB [11]), but they cost more on manufacturing hardware tokens (e.g., $60 per token[1] [2]) and sending short messages (e.g., $0.00645 per SMS [12]), respectively.

5.4 Prototype Implementation

Web Server Settings. Authentication server is implemented using CherryPy web framework. SQLite database is used to store username and password information. For experimental evaluation, we store each keystroke timing sequence and the corresponding random code into a text document and store the corresponding audio data into a 16-bit byte array in the debug version for data collection. In the released version of Typing-Proof, no keystroke timing information or audio is stored on server side. HTTPS is supported for communications between browsers/login computers and server, and between server and registered phones.

Web Client Settings. All major browsers support our prototype without any browser code modifications or plug-ins. In our experiment, we test our prototype on Google Chrome (version 55.0.2883.87), Internet Explorer 11 (version 11.0.9600.18860) and Microsoft Edge (version 41.16299.15.0). The client website is written entirely in HTML and JavaScript. We use jQuery keydown()

[1]The price of hardware token may drop significantly if a large number of hardware tokens are purchased.

Method [134] to record the timestamp of each key press event. We use Ajax to send and retrieve data from the server to the client asynchronously.

Mobile Client Settings. We develop an Android application and test it on a Google Nexus 5x, a Google Nexus 6 (both running on Android version 6.0.1) and a Huawei P10 (running on Android 8.0.1) smartphones. We use the Google Firebase Cloud Messaging (FCM) service [49] to send indicators to Android devices.

Time Synchronization. Typing-Proof requires that registered phones and corresponding login computers are loosely synchronized. For this reason, login computers and registered phones run a simple time-synchronization protocol (Network Time Protocol [143]) with the server. In a server-client scenario (where the client can be either login computer or registered phone), the client initiates a time-request exchange with the server so that the client is able to calculate the link delay and its local offset, and adjust its local clock to match the clock at the server's computer. The protocol can synchronize all participating devices and mitigate the effects of variable network latency. According to our experimental results, the NTP protocol usually maintains clock difference within tens of milliseconds, which is good enough for Typing-Proof.

5.5 Evaluation

In this section, we conduct an experiment to examine the effectiveness of Typing-Proof. We use our prototype to collect a large number of keystroke timing sequences and their corresponding audio samples. Following the similarity score calculation algorithm described in Section 5.3.2, we find the threshold of the similarity score that leads to the best results in terms of false rejection rate (FRR) and false acceptance rate (FAR).The performance evaluations of Typing-Proof are conducted in different settings.

Figure 5.3: An illustration of the three different environments tested in our experiment: One-person Office, Research Lab, Starbucks café.

5.5.1 Data Collection

Two volunteers recruited from our university logged in their accounts using Google Chrome, Internet Explorer, and Microsoft Edge[2] over 2 weeks. At each login, the volunteers are required to type at least 5 characters for the second-factor authentication. A login computer records keystroke timing sequence and a registered phone records audio through its microphone. This pair of data samples was stored for post-processing. The login attempts were conducted in various settings:

Environment: Different environment settings were used in our experiments, including a one-person office which is quiet, a research lab where many users sitting surrounding the user type on their own computers at the same time, and a Starbucks café with people's talking and background music. Figure 5.3 provides an illustration of the three environments tested in our experiment. Note that the cubicle in the research lab environment was surrounded by other 5-8 cubicles with a distance of around 1.5 meter between two cubicles next to each other.

Phone Position: In our experiment, the volunteers place their registered phones at various distances from the corresponding login computers, including a short distance (i.e., 20cm), a medium distance (i.e., 50cm), and a long distance (i.e., 100cm). Here, the phone-keyboard distance is measured from the center of a registered phone to the center of a login computer's keyboard.

[2]We used Google Chrome, Internet Explorer, and Microsoft Edge since they are currently most popular browsers [92]. We also test Typing-Proof with other browsers during our user study and experience similar performance.

Table 5.1: The number of the login attempts per volunteer for each combination of settings.

	One-person Office	Research Lab	Café
Desktop Keyboard	50S, 50M, 50L[1]	50S, 50M, 50L[1]	
Laptop Keyboard	50S, 50M, 50L[1]	50S, 50M, 50L[1]	50S[2]
Software Keyboard	50	50	50

[1] 50S, 50M, and 50L refer to collecting 50 login attempts per volunteer in the setting of short, medium, and long phone-keyboard distance, respectively.

[2] The table in the Starbucks is small so that we only consider the cases where users place registered phones 20cm away from keyboards.

Keyboard Model: We tested three different keyboard models for volunteers to use Typing-Proof on login computers, including a standard QWERTY keyboard (Acer PR1101U), a laptop keyboard on Mac Book Pro 13", and a software keyboard on Google Nexus 5x. In particular, the software keyboard setting refers to the scenario where a login computer and a registered phone are the same device (i.e., the registered phone). In our experiment, we used Google Keyboard-English (US) for the input, with the keypress vibration turned on and the keypress sound turned off. Therefore, for the software keyboard on a smartphone, we record the sound of keypress vibration instead of directly recording the sound of touching on the screen which is too slight to be recorded.

We collected 50 login attempts per volunteer for each combination of settings, totaling 1600 login attempts (1600 keystroke timing sequences and 1600 audio samples). Table 5.1 shows the structure of our dataset.

5.5.2 Parameters Configuration

The collected data is used to discover the best threshold τ_{sim} for comparing keystroke timing sequences with keystroke sound. The best threshold is selected according to FRR and FAR, where FRR measures the proportion of legitimate logins which are falsely rejected by the server, and FAR measures the proportion of fraudulent logins which are falsely accepted by the server. Note that in Typing-

Table 5.2: FRR and FAR when usability and security have different weights.

	FRR	FAR	Threshold
$\alpha = 0.1, \beta = 0.9$	0.07625	0.00603	0.394
$\alpha = 0.2, \beta = 0.8$	0.02188	0.01336	0.370
$\alpha = 0.3, \beta = 0.7$	0.00125	0.01961	0.358
$\alpha = 0.4, \beta = 0.6$	0.00125	0.01961	0.358
$\alpha = 0.5, \beta = 0.5$	0.00125	0.01961	0.358
$\alpha = 0.6, \beta = 0.4$	0.00000	0.02091	0.356
$\alpha = 0.7, \beta = 0.3$	0.00000	0.02091	0.356
$\alpha = 0.8, \beta = 0.2$	0.00000	0.02091	0.356
$\alpha = 0.9, \beta = 0.1$	0.00000	0.02091	0.356

Proof, the use of backup solution ensures that FRR is negligible assuming that users do not make any mistakes using the backup solution[3]. Therefore, we use "FRR" to denote how frequent the backup solution is activated in the following evaluations. We set t_{max} to 200ms since this is the highest clock difference experienced while testing our synchronization protocol (see Section 5.4). Using Typing-Proof, a volunteer/user is authenticated if and only if the similarity score is greater than the threshold τ_{sim} and $l < t_{max}$, where l is the offset where the two time series are most similar.

To compute FAR, we use the following strategy. For each audio sample recorded from one of the volunteers (acting as the victim), we use all the keystroke timing sequences recorded from the other volunteer as the attacker's samples. We then switch the roles of the two volunteers and repeat the above process. Since the length of the victim's audio sample and the duration of the attacker's keystroke timing sequence are mostly different, we cut the longer sample/sequence according to the shorter one for similarity comparison. The total number of comparisons is 800*800*2=1,280,000 in our experiment.

Figure 5.4 plots FRR curve and FAR curve as a function of threshold τ_{sim} when

[3]No mistake was observed in our experiments for users to compare two random codes displayed on a browser and on a registered phone, using the backup solution. The cognitive workload of comparing two random codes in Typing-Proof is significantly lower than remembering of a code displayed on a hardware token or phone and typing it on a browser [147] as it is required by hardware token based 2FA and SMS-based 2FA.

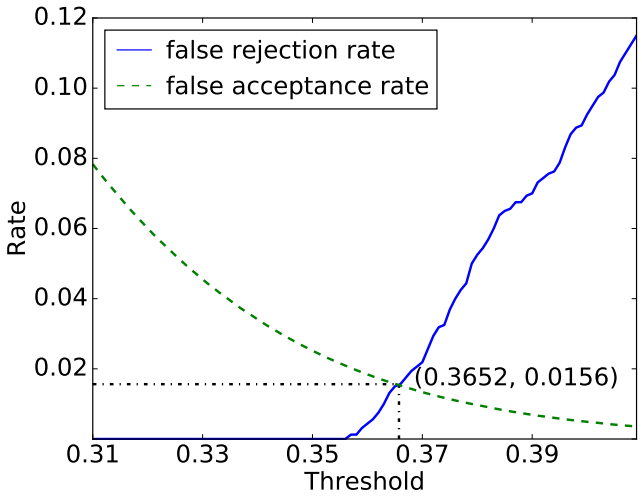


Figure 5.4: False rejection rate and false acceptance rate as a function of threshold τ_{sim}.

the backup solution of Typing-Proof is not used. The threshold τ_{sim} for similarity score can be determined based on the Equal Error Rate (ERR). ERR is the rate at which FRR and FAR are equal. The value of ERR is derived from the crossing point of FRR and FAR, which is 0.015625. We have $\tau_{sim} = 0.365235$ at this point.

The threshold for similarity score can also be computed when usability and security are weighted differently by the service provider. In particular, we compute the threshold that minimizes $f = \alpha \cdot FRR + \beta \cdot FAR$, for $\alpha \in [0.1, ..., 0.9]$ and $\beta = 1 - \alpha$. Table 5.2 provides FRR and FAR when usability and security have different weights. In Typing-Proof, we value security higher than usability since we have the backup solution which reduces FRR to almost zero. The FAR can be reduced to 0.006 if we set $\alpha = 0.1, \beta = 0.9$.

We observe that FAR is highly correlated with the length of random code. If the length of random code is longer, the keystroke patterns of different users are more diverse so that an attacker has a lower probability to bypass the second-factor authentication. Figure 5.5 shows the relationship between ERR and the minimum length l_{min} of random code. Figure 5.6 further shows the relationship between FAR and l_{min} for fixed FRRs. The FAR can be reduced significantly if users are required to type longer random codes for the second-factor authentication. For example, if it

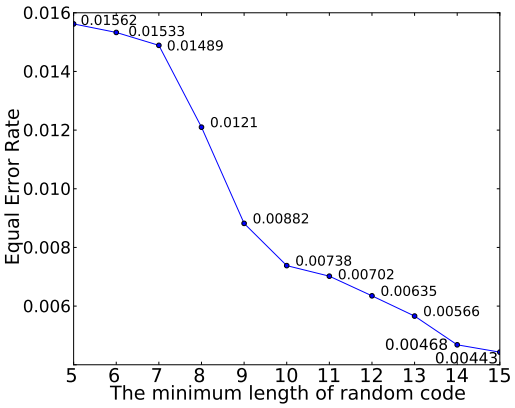


Figure 5.5: The relationship between ERR and the minimum length of random code.

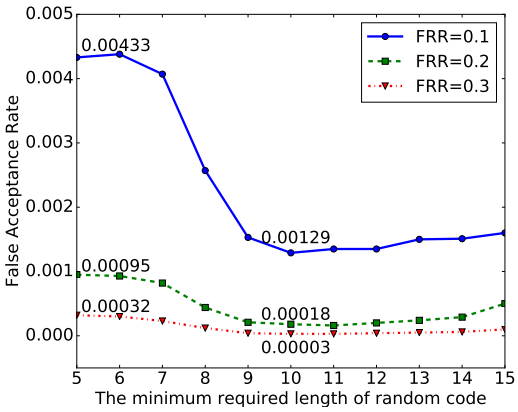


Figure 5.6: The relationship between FAR and the minimum length of random code when FRR is fixed.

is acceptable to resort to the backup solution by 30% of chance, the FAR is 0.003% for 10-digit or longer random codes.

In the following evaluations, we provide the performance of Typing-Proof under two configurations: a general configuration and a recommended configuration. The general configuration is set according to ERR: $t_{max} = 200ms$, $l_{min} = 5$, and $\tau_{sim} = 0.365235$. In practice, we recommend service providers to value security higher than usability and require users to type at least 10 characters for the second-factor. Our recommended configuration sets $\alpha = 0.1, \beta = 0.9$, $t_{max} = 200ms$, $l_{min} = 10$, and $\tau_{sim} = 0.37$.

5.5.3 False Rejection Rate

We evaluate the impacts of settings, including different environments, phone positions, and keyboard models, to FRR if the backup solution is not in use. The results are shown in Figure 5.7. The overall FRR is 0.015625 in the general configuration and 0.01847 in the recommended configuration. This implies that the frequency of Typing-Proof resorting to the backup solution is relatively low since users do not need to interact with their registered phones in most cases (i.e., over 98%). As a comparison, the FRR due to mistyped passwords is around 0.04 [70, 67].

Typing-proof performs equally well in one-person office and research lab, which

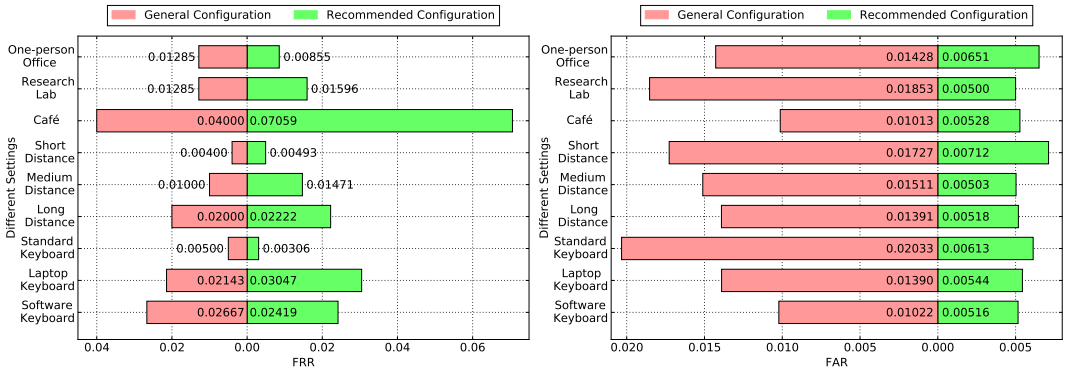


Figure 5.7: Impacts of different environments, phone positions, keyboard models to FRR and FAR in different configurations.

implies that the sounds of many users' typings at the same time do not affect the performance of Typing-Proof. In terms of phone positions, Typing-Proof performs best when the registered phone is placed 20cm away from the keyboard of a login computer. The performance of the long distance (i.e., 100cm) is 5 times worse than that of the short distance (i.e., 20cm). As for keyboard models, the standard QWERTY keyboard performs the best since this kind of keyboards produces loudest and clearest keystroke sound while laptop keyboard and software keyboard perform much worse. However, even in the worst case, the FRR is around 3% which is still low for resorting to the backup solution.

5.5.4 False Acceptance Rate

We further evaluate FAR in Figure 5.7. The overall FAR is 0.015625 in the general configuration and 0.00569 in the recommended configuration. In comparison, the FAR of Sound-Proof is 0.00200. It is worth noting that the FAR of Typing-Proof is measured in the worst case scenario where a victim is typing at the time of an attack. In practice, if the victim is not typing or his/her phone is put away from the victim's keyboard at the time of the attack, the attack may easily fail unless the attacker's keyboard is close enough to the victim's phone. We argue that the FAR of Typing-Proof is small enough for protecting not-so-sensitive user accounts such as those in online social networks. While for highly sensitive user accounts such

as those in financial services, FAR of Typing-Proof can be further reduced (e.g., 0.003%) by setting higher FRR (e.g., 30%) as shown in Figure 5.6.

With regard to FAR in different settings, we observe that the setting which leads to lower FRR would make FAR higher. In particular, Typing-Proof has a lower probability to be attacked when a victim stays in a noisier environment, places his/her registered phone farther away from the keyboard, and uses sound-lighter keyboard.

5.6 Security Analysis

Remote Attack. A remote attack requires an attacker to obtain a victim' username and password, pass the first-factor authentication, submit a keystroke timing sequence as the second factor on a login computer. The keystroke timing sequence is then sent to a victim's registered phone for similarity comparison. It also requires the victim to type at the same time so that the keystroke timing sequence x submitted by the attacker and the keystroke sound y recorded by the victim's registered phone are highly correlated within certain time lag t_{max}, that is, $max_l(CC'_{x,y}) > \tau_{sim}$ with $l < t_{max}$.

The security of Typing-Proof against remote attack stems from the attacker's inability to know whether the victim is typing and guess what the victim is typing at the time of the attack. We bound the time lag l between 0 and t_{max} to enhance the security of Typing-Proof.

Known Typed-Text Attack. A known typed-text attack is that a remote adversary could correctly guess what a victim is typing or predict what a victim will type at some point in time and submit the same typing sequence at the same time. Note that during such attack, the victim might be typing certain meaningful text rather than random code. However, it is still difficult to bypass Typing-Proof because the keystroke patterns of typing a same code are typically different for different users [90, 17]. To prove this, we conduct an experiment to evaluate the success rate of known typed-text attacks.

In this experiment, we select 25 frequently-used 5-letter words (e.g., 'there', 'would', 'about', and etc.) and 25 frequently-used words or phrases with no less than 10 characters (e.g., 'thanks so much', 'for instance', and etc.). One volunteer (acting as the victim) is required to type these words or phrases using Typing-Proof. The audio samples of his typing are collected. Another 9 volunteers (acting as attackers) are asked to type the same words or phrases as the victim has typed. Each word or phrase is typed for 5 times per volunteer and the corresponding keystroke timing sequences are recorded. In total, $50 * 5 * 9 = 2250$ attacking cases are generated.

We calculate the similarities between an attacker's keystroke timing sequences and the victim's audio samples for typing a same word or phrase. All similarities are lower than the thresholds selected in the Section 5.5.2 (i.e., $\tau_{sim} = 0.365235$ for the general configuration and $\tau_{sim} = 0.37$ for the recommended configuration), which implies that all known typed-text attacks failed. It is observed that if the minimum required length of random code is longer, the similarities in the attacking cases are lower. In particular, the average similarity of attacking a 5-letter word and attacking a word or a phrase no shorter than 10 characters are 0.17750, 0.12895, respectively. This indicates that service providers may set the minimum required length of random code to 10 or more so as to provide better protection against known-typed text attacks.

Co-located Attack. In a co-located attack, an attacker logins to a victim's account and types a random code in the same environment where the victim stays. Typing-Proof can withstand such attack since it is difficult for the victim's registered phone to capture the attacker's keystroke sound unless the victim's registered phone is very close to the attacker's keyboard. We conduct an additional experiment to evaluate the success rate of such co-located attack.

In this experiment, we assume that a victim and an attacker are located in a same environment and there is a certain distance between the attacker's keyboard and the victim's registered phone. In particular, in the one-person office and research lab

environments, we set the distance between the attacker's keyboard and the victim's registered phone as 150cm and 200cm, respectively. In the café environment, we let the attacker sit at the same table with the victim (phone-keyboard distance is around 50cm), and the attacker sit at the next table to the victim (phone-keyboard distance is around 100cm). We also consider the cases where the attacker and the victim typing at the same time and the cases where the victim is not typing at the time of co-located attack in each environment and for each phone position. In each test case, we run the attack 50 times and calculate the similarity between each audio sample collected by the victim's registered phone and the corresponding keystroke timing sequence generated by the attacker.

Our results show that the success rate of co-located attack is 0.00667 (4 out of 600 cases). In particular, 3 successful cases occur when the attacker sits at the same table with the victim in a Starbucks, and the rest successful case occurs when the attacker sits 150cm away from the victim in the one-person office. The victim is not typing at the time of attack in all four successful attacking cases. In order to launch a successful attack, the attacker needs to sit very close to the victim's registered phone and ensures that the victim himself/herself does not make any keystroke sound. However, this is likely to raise the victim's suspicion anyway. In suspicious cases, vigilant users can simply move their registered phones farther away (e.g., larger than 150cm) from the keyboards used by suspicious attackers.

Relay Attack. A relay attack is that an attacker obtains a victim' username and password, passes the first-factor authentication, records the keystroke sound of his/her typing of a random code, and plays the keystroke sound near a victim's registered phone with a speaker in real time. In a legitimate authentication, the real keystrokes are produced from different positions on a keyboard unless a user types repeated keys as random code. In a relay attack, however, the sounds played from a speaker come from the same source. Such relay attack can be detected by analyzing the time-difference-of-arrival of keystroke sound and determining whether the sound source is moving. This approach was first proposed by Zhang et al. for voice

liveness detection [153]. It requires that the registered phone is equipped with two microphones, which is met by most popular smartphones (the smartphone products of Samsung, Apple, Huawei, and Xiaomi whose total market share is more than 62.32% [122] are equipped with at least two microphones). Although an attacker may deliberately choose to type repetitive "random" code like "aaaaa" in a relay attack, we recommend users not to type repetitive codes and if such codes are sent to the registered phones, Typing-Proof is switched to the backup solution.

Sound-Danger Attack. A sound-danger attack [153] is that an attacker deliberately makes a victim's registered phone to produce previously known sounds (see Section 2.2). In order to launch a successful sound-danger attack against Typing-Proof, an attacker needs to trigger the victim's registered phone to produce the keystroke sound that matches the timing sequences of the attacker's typing for 2FA.

While an attacker may use ringtone, notification tones or notification vibrations to simulate keystroke sound on the victim's phone, such attack can be detected by checking the signatures of keystroke sound which are different from the signatures of triggered tones/vibrations [131]. Another countermeasure is to temporarily disable the function of a Typing-Proof application on a registered phone whenever a notification is received. Android platform provides Class `NotificationListenerService` [51] to monitor whether any new notification is received.

Alternatively, an attacker may choose a random code, craft an audio signal which contains the keystroke sound for typing this code, hide the audio signal into a video or audio recording, and trick a victim to play the recording (e.g., through a manipulated website or YouTube video). At the same time when the victim plays the recording, the attacker submits the chosen random code to the victim's account. Since Typing-Proof uses only the frequency higher than 15000Hz from an audio sample, it is even easier to hide the high-frequency part of the keystroke sound, which is inaudible to human. However, such attacks can still be detected by analyzing the time-difference-of-arrival of keystroke sound, which is the same as the

countermeasure of relay attacks.

5.7 User Study

An IRB-approved user study was conducted to evaluate the usability of Typing-Proof and to compare it with the usability of Sound-Proof and SMS-based 2FA.

5.7.1 Procedure

Our user study involved 25 participants, including 16 males and 9 females with ages from 21 to 27. All participants were students or staff in a university. All participants were informed that no personal information is collected and that the survey in the user study is anonymous.

The user study took place in a classroom. Most participants used their own laptops and their own Android smartphones for 2FA logins. For the participants who did not have any Android phones, we provided our test-phones for them. All devices were connected to the Internet through WiFi. We set up a server on a desktop, Acer Veriton M4630G running Windows 7, and created a website that integrates Typing-Proof, Sound-Proof, and SMS-based 2FA. All participants were required to install our application on their phones which supports all three authentication mechanisms.

During the user study, each participant was asked to log in to the server using all three mechanisms in random order and for several times. After using all 2FA mechanisms, participants were required to fill in a survey. The survey includes three parts: demographic information, System Usability Scale (SUS) [23], and a post-test questionnaire which covers various aspects of 2FA mechanisms that are not covered by the SUS.

Table 5.3: The items of the System Usability Scale [23]. All items are answered with a 5-point Likert-scale from *Strongly Disagree* to *Strongly Agree*.

Q1	I would like to use this system frequently.
Q2	I found the system unnecessarily complex.
Q3	I thought the system was easy to use.
Q4	I would need the support of a technical person to be able to use this system.
Q5	I found the various functions in this system were well integrated.
Q6	I thought there was too much inconsistency in this system.
Q7	I would imagine that most people would learn to use this system very quickly.
Q8	I found the system very cumbersome to use.
Q9	I felt very confident using the system.
Q10	I needed to learn a lot of things before I could get going with this system.

Table 5.4: The items of the post-test questionnaire. All items are answered with a 5-point Likert-scale from *Strongly Disagree* to *Strongly Agree*.

Q1	I thought this system was quick. (2FA-quick)
Q2	If 2FA were mandatory, I would use this system to log in (2FA-mandatory).
Q3	If 2FA were optional, I would use this system to log in (2FA-optional).
Q4	I would feel comfortable using this system at home (Use @ home).
Q5	I would feel comfortable using this system at workplace (Use @ workplace).
Q6	I would feel comfortable using this system at café (Use @ café).
Q7	I would feel comfortable using this system at library (Use @ library).

5.7.2 Usability

Survey Results. All participants had ever used 2FA for online banking and 88% of them had the experience of using 2FA for online payment. Only 28% and 24% of them respectively ever used 2FA in Google Services (e.g., Gmail) and Apple Services (e.g., iCloud). Our experience is that many finance-related services, like online banking or online payment, enforce users to use 2FA, while other services, such as Gmail or iCloud, make 2FA optional, in which case many users choose to opt out.

The System Usability Scale (SUS) is widely used to assess the usability of IT systems [16]. Its score ranges from 0 to 100, where a higher score indicates better usability. Table 5.3 reports the items of SUS. The mean SUS scores for Typing-Proof, Sound-Proof, and SMS-based 2FA are 81.7 (± 14.68), 69.2 (± 18.02), 73.4 (± 12.62), respectively. The result shows that the usability of Typing-Proof is

obviously better than the other two mechanisms. One interesting observation is that the mean SUS score of Sound-Proof is a little bit lower than that of SMS-based 2FA while the standard deviation of Sound-Proof's SUS score is larger than that of SMS-based 2FA. One potential reason is that some participants' browsers do not support audio recording, which contributes to the low scores on the usability evaluation of Sound-Proof.

The post-test questionnaire is similarly designed as that in [67], aiming to collect information on the perceived quickness of the three mechanisms (2FA-quick for short in Figure 5.8) and participants' willingness to adopt them (2FA-mandatory in mandatory setting, and 2FA-optional in optional setting for short in Figure 5.8). It also inquires of participants whether they feel comfortable using the mechanisms in different environments, including use @ home, use @ workplace, use @ café, use @ library in Figure 5.8. The full post-test questionnaire is listed in Table 5.4. Figure 5.8 summarizes the participants' answers on 5-point Likert-scales in a radar chart plot. In general, participants show the strongest willingness to adopt Typing-Proof. Most participants evaluated that both Typing-Proof and Sound-Proof are much quicker than SMS-based 2FA. Similar to [67], our results show that participants tend not to use SMS-based 2FA if it is optional, while for Typing-Proof and Sound-Proof, the difference in users' acceptance between mandatory setting and optional setting is much less significant. More than 88% of participants evaluated that Typing-Proof is suitable to be used at home, at their workplace, while fewer participants would use Typing-Proof in a public place (i.e., at a café or library). As for SMS-based 2FA, participants shared a similar willingness in various scenarios.

Login Time. The login time we measured in our user study is from the start of the second-factor authentication (i.e., after username and password is verified), to the moment when the login attempt is accepted. We did not witness any login failure in our experiment. Therefore, the login time of Typing-Proof does not include the potential time of one-button authentication in the backup solution, which takes 7.1 seconds on average as measured separately in our experiments. The averaged login

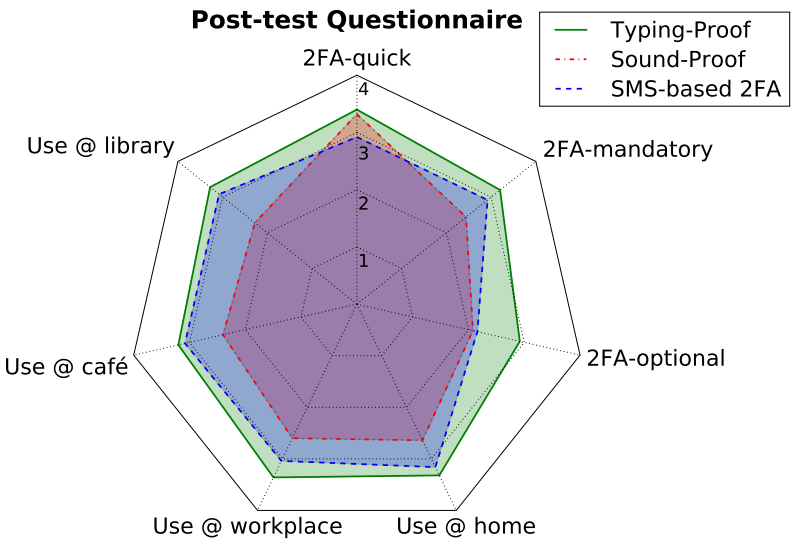


Figure 5.8: Answer to the post-test questionnaire.

time for Typing-Proof is 4.3 seconds while it is 5.0 seconds for Sound-Proof and 10.4 seconds for SMS-based 2FA. Among the three mechanisms, Typing-Proof is the most favorable in terms of login time.

5.8 Discussion

Adjustable Security. When Typing-Proof is used, a longer random code makes it more difficult for an attacker to launch a successful attack. On the other hand, typing longer random codes will lower the usability of Typing-Proof. Therefore, users may take different typing strategies according to their needs. For important authentications in a less secure environment, users may choose to type longer random codes, while for less important authentications in a more secure environment, users may choose to type shorter random codes.

Transaction Authentication. Transaction authentication is another important application of 2FA. Transaction authentication may suffer from Man-in-the-Mobile attack and Man-in-the-Browser attack [1] in which an attacker may change the content of a transaction such as destination account number and transaction amount. To solve this problem in Typing-Proof, user's transaction details should be sent to

user's registered phones via the server. Users should use the backup solution of Typing-Proof and check it out before pressing the 'Approve' button for transaction authentication.

CAPTCHA. Many existing authentication systems involve the use of CAPTCHA. Typing-Proof can be easily used when CAPTCHA is involved. Instead of typing random codes, users may type CAPTCHA codes for 2FA. The overall user experience of Typing-Proof is same with CAPTCHA-based password authentication if the backup solution is not triggered.

Keyboard Protector. Some users may place keyboard protectors on their keyboards in order to prevent dust entry or liquids. Commercial keyboard protectors made of silicone can effectively reduce keystroke sound. It may lead to high FRR when users login to their accounts using Typing-Proof. In the light of this, we recommend users to take the keyboard protector off when they use Typing-Proof (otherwise they need to resort to the backup solution).

Combined with other mechanisms. Typing-Proof can be combined with other 2FA mechanisms, including Sound-Proof, hardware token based 2FA, and SMS-based 2FA. Furthermore, Typing-Proof and Sound-Proof can work simultaneously to make authentications more usable and secure. In particular, if a user logins to his/her account in a quiet environment where Sound-Proof does not work, the server can rely on Typing-Proof for 2FA. On the other hand, if the user uses 2FA in a noisy environment, Sound-Proof activated for a better decision.

Alternative Devices. Currently, Typing-Proof uses a smartphone as a software token. It is straightforward to replace it with other smart devices such as smartwatch for 2FA. Compared to using smartphone, the use of smartwatch in Typing-Proof may further lower FRR since the distance between the keyboard of a login computer and the smartwatch of a user who logins to his/her account wearing the smartwatch should be shorter than the short distance (i.e., 20cm) that is used in our experiments.

Comparative Analysis. The framework of Bonneau et al. [21] can be used to compare Typing-Proof, Sound-Proof and SMS-based 2FA in terms of usability, deploy-

Table 5.5: Comparison of Typing-Proof against Sound-Proof [67] and SMS-based 2FA [50] using the framework of Bonneau et al. [21]. We use 'Y' to denote that the benefit is provided and 'S' to denote that the benefit is somewhat provided.

	Usability								Deployability						Security											
Scheme	Memorywise-Effortless	Scalable-for-Users	Nothing-to-Carry	Physically Effortless	Easy-to-Learn	Efficient-to-Use	Infrequent-Errors	Easy-Recovery-from-Loss	Accessible	Negligible-Cost-per-User	Server-Compatible	Browser-Compatible	Mature	Non-Proprietary	Resilient-to-Physical-Observation	Resilient-to-Targeted-Impersonation	Resilient-to-Throttled-Guessing	Resilient-to-Unthrottled-Guessing	Resilient-to-Internal-Observation	Resilient-to-Leaks-from-Other-Verifiers	Resilient-to-Phishing	Resilient-to-Theft	No-Trusted-Third-Party	Requiring-Explicit-Consent	Unlinkable
Typing-Proof		Y	S	Y	Y	Y	S	S	Y	Y		Y		Y	S	S	Y	Y		Y	Y	Y	Y	Y	
Sound-Proof		Y	S	Y	Y	Y	S	S	Y	Y		S		Y	S		S	Y		Y	Y	Y	Y	Y	
SMS-based 2FA			S		Y	S	S	S	S	S		Y	Y	Y	S	S	Y	Y		Y	Y	Y	Y	Y	

ability, and security. Table 5.5 shows the comparison. In general, both Typing-Proof and Sound-Proof achieve better usability than SMS-based 2FA. The deployability of Typing-Proof is better than Sound-Proof since Sound-Proof may not be exactly browser-compatible. We observed several cases where participants' browsers did not support audio recording in our user study. As for the security aspect of the comparison, Typing-Proof is better than Sound-Proof.

Chapter 6

Dissertation Conclusion and Future Work

6.1 Summary of Contribution

This dissertation makes contributions to understanding the potential risk of inter-keystroke timing information disclosure and designing a secure, usable and low-cost two-factor authentication systems.

Our first work proposed a user-independent inter-keystroke timing attack on PINs based on a human cognitive model. The human cognitive model allows an attacker to build a timing dictionary of all possible PINs ranked according to the cosine similarity between the observed timing sequence of a target PIN and each possible PIN's predicted timing sequence. We examined the effectiveness of our attacks to the PINs at different PIN strength levels in different online attack settings. The results demonstrated that our attacks achieve satisfactory performance. We also suggested several countermeasures to mitigate our attacks. This work has been published in Computers & Security journal [79].

We proposed UltraPIN to infer PIN entries via inaudible ultrasounds in the second work. UltraPIN is practical as it can be launched from commodity smartphones which are widely available today. UltraPIN is effective since the success

rate of recovering a PIN within three attempts reaches 75% in our recommended setting. Rigorous experiments showed that UltraPIN is robust with respect to different keypad layouts, keypad sizes, keypad angles, smartphone quantities and positions, smartphone-keypad distances, and experimental environments. We hope UltraPIN would help raise the awareness that PIN-based user authentication systems, including ATM, POS, and electronic door locks, may not be as secure as previously considered. It may also help promote further studies and the adoption of more secure solutions such as using randomized keypads in the near future.

In the third work, we proposed Typing-Proof, a usable, secure, and low-cost two-factor authentication mechanism. Typing-Proof does not require a user to interact with his/her phone in most cases and does not have any memory demand. It can be used in any environment and is compatible with major browsers, PCs, and phones without requiring any additional plug-ins or hardware. Typing-Proof is secure against practical attacks, including remote attack, sound-danger attack, co-located attack, and relay attack. Compared to hardware token based 2FA, SMS-based 2FA and Sound-Proof, Typing-Proof enables significant cost saving for both service providers and users. This brings in high commercial potential which may foster large-scale adoptions. This work has been published in 2018 Annual Computer Security Applications Conference (ACSAC 2018) [78].

6.2 Future Research Plan

One of the main weaknesses of our proposed two attacks is that our attacks may have low performance or even become invalid when a victim uses more than one finger. Although according to our survey conducted in 2018, a majority of users (about 63.2%) prefer using a single finger for PIN entry, it is easy to mitigate our attacks by warning the user to intentionally enter his/her PIN using multiple fingers. Furthermore, our attacks are not powerful enough to infer the inputs on a standard QWERTY keyboard, which involves more complicated hand and finger

movements. To solve this challenge, It may require to launch a targeted attack and train a customized model for other typing styles.

In addition, keystroke is a common behavior for inputting sensitive information, including password, PIN or other secret documents. It inevitably becomes an attractive target to cyber-criminals. On the rise of various sensors embedded in commodity off-the-shelf mobile devices (e.g., smartphones, smartwatches), certain side channels may emit more useful information about a victim's typing behaviors. For instance, Google Pixel 4 which will be released in the near future, has embedded a radar chip called soli [124]. This sensor enables gesture recognition, but may also be abused by an attacker. More powerful sensors may introduce larger sensitive information leakage in real world.

On one hand, it is urgent to seek for effective defense mechanism to address the above problems. Although many efforts have been made to design a securer authentication system, few of them are adopted by existing service providers. The underlying reason is that a securer authentication system will lead to user-unfriendly to some extent. Therefore, it is important for an authentication system designer to balance the trade-off between security and usability.

On the other hand, as revealed in Chapter 5 in this dissertation, side-channel information can also be leveraged as a second authentication factor or biometric factor. In the future, as more and more Internet of Thing (IoT) devices, including vehicles, becoming a part of people's life gradually, passive keyless authentication will get a lot of attention. The current passive keyless entry systems for vehicles are based on Bluetooth communication between car and car owner's trusted smartphone. However, this method is subject to relay attacks [42]. Although sophisticated anti relay attack approaches have been proposed [35, 85], all of them are based on professional hardware, which costs considerably. It is still a challenge to design a software-based authentication against relay attacks. Proximity-based authentication via side-channel information may be a promising solution.

Bibliography

[1] M. Adham, A. Azodi, Y. Desmedt, and I. Karaolis. How to attack two-factor authentication internet banking. In *International Conference on Financial Cryptography and Data Security*, pages 322–328. Springer, 2013.

[2] Aladdin. Two-factor authentication – the real cost of ownership. `https://mpa.co.nz/media/4410/twofactorauthentication-therealcostofownership.pdf`.

[3] K. Ali, A. X. Liu, W. Wang, and M. Shahzad. Keystroke recognition using wifi signals. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, pages 90–102. ACM, 2015.

[4] J. R. Anderson. *The Architecture of Cognition*. Psychology Press, 2013.

[5] Apple. Airpods. `https://www.apple.com/airpods/`, 2019.

[6] H. J. Asghar, S. Li, R. Steinfeld, and J. Pieprzyk. Does counting still count? Revisiting the security of counting based user authentication protocols against statistical attacks. In *Proceedings of the 20th Annual Network and Distributed System Security Symposium, NDSS*, 2013.

[7] H. J. Asghar, J. Pieprzyk, and H. Wang. A new human identification protocol and coppersmith's baby-step giant-step algorithm. In *ACNS*, volume 10, pages 349–366. Springer, 2010.

[8] M. Aslam, R. N. Idrees, M. M. Baig, and M. A. Arshad. Anti-hook shield against the software key loggers. In *Proceedings of the 2004 National Conference on Emerging Technologies*, page 189, 2004.

[9] D. Asonov and R. Agrawal. Keyboard acoustic emanations. In *IEEE Proceedings of the 2004 Symposium on Security and Privacy*, pages 3–11, 2004.

[10] A. J. Aviv, B. Sapp, M. Blaze, and J. M. Smith. Practicality of accelerometer side channels on smartphones. In *Proceedings of the 28th Annual Computer Security Applications Conference*, pages 41–50. ACM, 2012.

[11] AWS. Amazon EC2 pricing. `https://aws.amazon.com/cn/ec2/pricing/on-demand`.

[12] AWS. Worldwide SMS pricing. `https://aws.amazon.com/cn/sns/sms-pricing`.

[13] M. Backes, T. Chen, M. Dürmuth, H. P. Lensch, and M. Welk. Tempest in a teapot: Compromising reflections revisited. In *IEEE Proceedings of the 2009 Symposium on Security and Privacy*, pages 315–327. IEEE, 2009.

[14] M. Backes, M. Dürmuth, and D. Unruh. Compromising reflections-or-how to read LCD monitors around the corner. In *IEEE Proceedings of the 2008 Symposium on Security and Privacy*, pages 158–169. IEEE, 2008.

[15] X. Bai, W. Gu, S. Chellappan, X. Wang, D. Xuan, and B. Ma. Pas: predicate-based authentication services against powerful passive adversaries. In *Computer Security Applications Conference, 2008. ACSAC 2008. Annual*, pages 433–442. IEEE, 2008.

[16] A. Bangor, P. T. Kortum, and J. T. Miller. An empirical evaluation of the system usability scale. *Intl. Journal of Human–Computer Interaction*, 24(6):574–594, 2008.

[17] F. Bergadano, D. Gunetti, and C. Picardi. User authentication through keystroke dynamics. *ACM Transactions on Information and System Security (TISSEC)*, 5(4):367–397, 2002.

[18] Y. Berger, A. Wool, and A. Yeredor. Dictionary attacks using keyboard acoustic emanations. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, pages 245–254. ACM, 2006.

[19] A. Beutel, P. Covington, S. Jain, C. Xu, J. Li, V. Gatto, and E. H. Chi. Latent cross: Making use of context in recurrent recommender systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 46–54. ACM, 2018.

[20] Blizzard. Introducing the one-button authenticator. `http://us.battle.net/heroes/en/blog/20152210/introducing-the-one-button-authenticator-6-16-2016`.

[21] J. Bonneau, C. Herley, P. C. Van Oorschot, and F. Stajano. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 553–567. IEEE, 2012.

[22] J. Bonneau, S. Preibusch, and R. J. Anderson. A birthday present every eleven wallets? the security of customer-chosen banking PINs. In *Financial Cryptography*, volume 7397, pages 25–40, 2012.

[23] J. Brooke et al. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.

[24] L. Cai and H. Chen. Touchlogger: Inferring keystrokes on touch screen from smartphone motion. In *Proceedings of the 6th USENIX Conference on Hot Topics in Security*, 2011.

[25] S. K. Card, T. P. Moran, and A. Newell. The keystroke-level model for user performance time with interactive systems. *Communications of the ACM*, 23(7):396–410, 1980.

[26] S. K. Card, T. P. Moran, and A. Newell. The model human processor: An engineering model of human performance. In *Handbook of Perception and Human Performance*, volume 2, pages 45–1, 1986.

[27] V. C. Chen, F. Li, S.-S. Ho, and H. Wechsler. Micro-doppler effect in radar: phenomenon, model, and simulation study. *IEEE Transactions on Aerospace and electronic systems*, 42(1):2–21, 2006.

[28] Y. Cheng, Y. Li, R. Deng, L. Ying, and W. He. A study on a feasible no-root approach on android. *Journal of Computer Security*, pages 1–23, 2017.

[29] A. Czeskis, M. Dietz, T. Kohno, D. Wallach, and D. Balfanz. Strengthening user authentication through opportunistic cryptographic identity assertions. In *Proceedings of the 19th ACM Conference on Computer and Communications Security*, pages 404–414. ACM, 2012.

[30] A. De Luca, M. Langheinrich, and H. Hussmann. Towards understanding ATM security– A field study of real world ATM use. In *Proceedings of the 6th Symposium on Usable Privacy and Security*, pages 16:1–16:10. ACM, 2010.

[31] G. de Souza Faria and H. Y. Kim. Identification of pressed keys from mechanical vibrations. *IEEE Transactions on Information Forensics and Security*, 8(7):1221–1229, 2013.

[32] G. de Souza Faria and H. Y. Kim. Identification of pressed keys by time difference of arrivals of mechanical vibrations. *Computers & Security*, 57:93–105, 2016.

[33] W. Diao, X. Liu, Z. Li, and K. Zhang. No pardon for the interruption: New inference attacks on android through interrupt timing analysis. In *IEEE Proceedings of the 2016 Symposium on Security and Privacy*, pages 414–432. IEEE, 2016.

[34] R. J. Doviak et al. *Doppler radar and weather observations*. Courier Corporation, 2006.

[35] S. Drimer, S. J. Murdoch, et al. Keep your enemies close: Distance bounding against smartcard relay attacks. In *USENIX security symposium*, volume 312, 2007.

[36] Duo-Security. DUO PUSH: Quickly verify your identity. `https://duo.com/product/trusted-users/two-factor-authentication/authentication-methods/duo-push`.

[37] S. Estes. Cogulator: A cognitive modeling calculator. `http://cogulator.io`, 2017.

[38] Facebook. Security key for safer logins with a touch. `https://www.facebook.com/notes/facebook-security/security-key-for-safer-logins-with-a-touch/10154125089265766`, 2017.

[39] V. Filonenko, C. Cullen, and J. Carswell. Investigating ultrasonic positioning on mobile phones. In *2010 International Conference on Indoor Positioning and Indoor Navigation*, pages 1–8. IEEE, 2010.

[40] P. M. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47(6):381, 1954.

[41] D. Florêncio, C. Herley, and P. C. Van Oorschot. Pushing on string: The 'don't care' region of password strength. *Communications of the ACM*, 59(11):66–74, 2016.

[42] A. Francillon, B. Danev, and S. Capkun. Relay attacks on passive keyless entry and start systems in modern cars. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*. Eidgenössische Technische Hochschule Zürich, Department of Computer Science, 2011.

[43] D. M. Freeman, S. Jain, M. Dürmuth, B. Biggio, and G. Giacinto. Who are you? A statistical approach to measuring user authenticity. In *Proceedings of the 23th Annual Network and Distributed System Security Symposium, NDSS*. Internet Society, 2016.

[44] Futurea. Futurae authentication suite. `https://futurae.com/product/`.

[45] M. Gallagher. 7 questions about sample rate. `https://www.sweetwater.com/insync/7-things-about-sample-rate/`, 2019.

[46] S. Garera, N. Provos, M. Chew, and A. D. Rubin. A framework for detection and measurement of phishing attacks. In *Proceedings of the 2007 ACM workshop on Recurring malcode*, pages 1–8. ACM, 2007.

[47] D. R. Gentner. Evidence against a central control model of timing in typing. *Journal of Experimental Psychology: Human Perception and Performance*, 8(6):793–810, 1982.

[48] D. R. Gentner. The acquisition of typewriting skill. *Acta Psychologica*, 54(1):233–248, 1983.

[49] GOOGLE. Firebase cloud messaging. `https://firebase.google.com/docs/cloud-messaging`.

[50] GOOGLE. Google 2-Step Verification. `https://www.google.com/landing/2step`.

[51] Google. Notificationlistenerservice. `https://developer.android.com/reference/android/service/notification/NotificationListenerService.html`.

[52] Google. Tensorflow lite. `https://www.tensorflow.org/mobile/tflite/`, 2017.

[53] K. B. Grant. Here's another reason to think twice before using your debit card. `https://www.cnbc.com/2018/03/06/protect-your-bank-accounts-from-rising-debit-card-fraud.html`, 2018.

[54] P. A. Grassi and J. L. Fenton. NIST SP800-63-2: Electronic authentication guideline. Technical report, NIST, Reston, VA. `http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63-2.pdf`, 2013.

[55] P. A. Grassi and J. L. Fenton. NIST SP800-63B: Digital authentication guideline. Technical report, NIST, Reston, VA. `https://pages.nist.gov/800-63-3/sp800-63b.html`, 2016.

[56] D. Gruss, C. Maurice, K. Wagner, and S. Mangard. Flush+ flush: a fast and stealthy cache attack. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 279–299. Springer, 2016.

[57] D. Gruss, R. Spreitzer, and S. Mangard. Cache template attacks: Automating attacks on inclusive last-level caches. In *Proceedings of the 24th Conference on USENIX Security Symposium*, pages 897–912. USENIX Association, 2015.

[58] R. Hackett. Linkedin lost 167 million account credentials in data breach. `http://fortune.com/2016/05/18/linkedin-data-breach-email-password`.

[59] X. He and T.-S. Chua. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 355–364. ACM, 2017.

[60] X. He, X. Du, X. Wang, F. Tian, J. Tang, and T.-S. Chua. Outer product-based neural collaborative filtering. *arXiv preprint arXiv:1808.03912*, 2018.

[61] N. J. Hopper and M. Blum. Secure human identification protocols. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 52–66. Springer, 2001.

[62] HuaWei. Huawei freebuds 2 wireless earphone. `https://consumer.huawei.com/cn/accessories/freebuds2/`, 2019.

[63] A. Inc. If the microphones on your iPhone, iPad, and iPod touch aren't working. `https://support.apple.com/en-us/HT203792`, 2019.

[64] P. Jaiswar. Debit card holder? never do this - what happens after entering wrong ATM PIN 3 times in a row. `https://www.zeebiz.com/personal-finance/news-debit-card-holder-never-do-this-what-happens-after-entering-wrong-atm-pin-3-times-in-a-row-83904`, 2019.

[65] B. E. John. Typist: A theory of performance in skilled typing. *Human-Computer Interaction*, 11(4):321–355, 1996.

[66] N. Karapanos and S. Capkun. On the effective prevention of tls man-in-the-middle attacks in web applications. In *Proceedings of the 23th Conference on USENIX Security Symposium*, 2014.

[67] N. Karapanos, C. Marforio, C. Soriente, and S. Capkun. Sound-proof: Usable two-factor authentication based on ambient sound. In *Proceedings of the 24th Conference on USENIX Security Symposium*, pages 483–498, 2015.

[68] S. Khandelwal. Download: 68 million hacked dropbox accounts are just a click away! `https://thehackernews.com/2016/10/dropbox-password-hack.html`.

[69] D. Kumar. Top 4 advantages and disadvantages of support vector machine or svm. `https://medium.com/@dhiraj8899/top-4-advantages-and-disadvantages-of-support-vector-machine-or-svm-a3c06a2b107`, 2019.

[70] M. Kumar, T. Garfinkel, D. Boneh, and T. Winograd. Reducing shoulder-surfing by using gaze-based password entry. In *Proceedings of the 3rd symposium on Usable privacy and security*, pages 13–19. ACM, 2007.

[71] D. F. Kune and Y. Kim. Timing attacks on PIN input devices. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, pages 678–680. ACM, 2010.

[72] P. Lazik and A. Rowe. Indoor pseudo-ranging of mobile devices using ultrasonic chirps. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, pages 99–112. ACM, 2012.

[73] M. Li, Y. Meng, J. Liu, H. Zhu, X. Liang, Y. Liu, and N. Ruan. When CSI meets public WiFi: Inferring your mobile phone password via WiFi signals. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1068–1079. ACM, 2016.

[74] S. Li and H.-Y. Shum. Secure human-computer identification (interface) systems against peeping attacks: Sechci. *Computer Science Preprint Archive*, pages 21–69, 2005.

[75] M. Lipp, D. Gruss, M. Schwarz, D. Bidner, C. Maurice, and S. Mangard. Practical keystroke timing attacks in sandboxed javascript. In *European Symposium on Research in Computer Security*. Springer, 2017.

[76] M. Lipp, D. Gruss, R. Spreitzer, C. Maurice, and S. Mangard. Armageddon: Cache attacks on mobile devices. In *Proceedings of the 25th Conference on USENIX Security Symposium*, pages 549–564, 2016.

[77] J. Liu, Y. Wang, G. Kar, Y. Chen, J. Yang, and M. Gruteser. Snooping keystrokes with mm-level audio ranging on a single phone. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, pages 142–154. ACM, 2015.

[78] X. Liu, Y. Li, and R. H. Deng. Typing-proof: Usable, secure and low-cost two-factor authentication based on keystroke timings. In *Proceedings of the 34th Annual Computer Security Applications Conference*, pages 53–65. ACM, 2018.

[79] X. Liu, Y. Li, R. H. Deng, B. Chang, and S. Li. When human cognitive modeling meets pins: User-independent inter-keystroke timing attacks. *Computers & Security*, 80:90–107, 2019.

[80] X. Liu, Z. Zhou, W. Diao, Z. Li, and K. Zhang. When good becomes evil: Keystroke inference with smartwatch. In *Proceedings of the 22nd ACM Conference on Computer and Communications Security*, pages 1273–1285. ACM, 2015.

[81] A. Maiti, O. Armbruster, M. Jadliwala, and J. He. Smartwatch-based keystroke inference attacks and context-aware protection mechanisms. In *Proceedings of the 11th ACM Asia Conference on Computer and Communications Security*, pages 795–806. ACM, 2016.

[82] A. Manohar. Atm, debit, credit card cloning fraud: Beware! avoid doing these mistakes, and save your hard-earned money. `https://www.zeebiz.com/india/news-atm-debit-credit-card-cloning-fraud-beware-avoid-doing-these-mistakes-and-save-your-hard-earned-money-80237`, 2019.

[83] P. Marquardt, A. Verma, H. Carter, and P. Traynor. (sp)iPhone: Decoding vibrations from nearby keyboards using mobile phone accelerometers. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, pages 551–562. ACM, 2011.

[84] I. Martinovic, K. Rasmussen, M. Roeschlin, and G. Tsudik. Authentication using pulse-response biometrics. In *Proceedings of the 21st Annual Network and Distributed System Security Symposium, NDSS*, 2014.

[85] S. Mauw, Z. Smith, J. Toro-Pozo, and R. Trujillo-Rasua. Distance-bounding protocols: Verification without time and location. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 549–566. IEEE, 2018.

[86] MDN. Bluetooth.requestdevice(). `https://developer.mozilla.org/en-US/docs/Web/API/Bluetooth/requestDevice`.

[87] MDN. Mediadevices.getusermedia(). `https://developer.mozilla.org/zh-CN/docs/Web/API/MediaDevices/getUserMedia`.

[88] Microsoft. One easy-to-use app for all your multi-factor authentication needs. `https://dirteam.com/sander/2016/08/15/microsoft-authenticator-one-easy-to-use-app-for-all-your-multi-factor-authentication-needs/`.

[89] E. Miluzzo, A. Varshavsky, S. Balakrishnan, and R. R. Choudhury. TaPrints: Your finger taps have fingerprints. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, pages 323–336. ACM, 2012.

[90] F. Monrose and A. D. Rubin. Keystroke dynamics as a biometric for authentication. *Future Generation Computer Systems*, 16(4):351–359, 2000.

[91] D. Morawiec. sklearn-porter. Transpile trained scikit-learn estimators to C, Java, JavaScript and others.

[92] NetApplications. Browser market share. `https://www.netmarketshare.com/browser-market-share.aspx`.

[93] Y. Oren, V. P. Kemerlis, S. Sethumadhavan, and A. D. Keromytis. The spy in the sandbox: Practical cache attacks in javascript and their implications. In *Proceedings of the 22nd ACM Conference on Computer and Communications Security*, pages 1406–1418. ACM, 2015.

[94] S. Ortolani and B. Crispo. Noisykey: Tolerating keyloggers via keystrokes hiding. In *HotSec*, 2012.

[95] S. Ortolani, C. Giuffrida, and B. Crispo. Bait your hook: a novel detection technique for keyloggers. In *International Workshop on Recent Advances in Intrusion Detection*, pages 198–217. Springer, 2010.

[96] E. Owusu, J. Han, S. Das, A. Perrig, and J. Zhang. ACCessory: Password inference using accelerometers on smartphones. In *Proceedings of the 12th Workshop on Mobile Computing Systems & Applications*, page 9. ACM, 2012.

[97] E. W. Patton and W. D. Gray. SANLab-CM: A tool for incorporating stochastic operations into activity network modeling. *Behavior Research Methods*, 42(3):877–883, 2010.

[98] G. Peeters. A large set of audio features for sound description (similarity and classification) in the cuidado project. *CUIDADO IST Project Report*, 54(0):1–25, 2004.

[99] P. Pessl, D. Gruss, C. Maurice, M. Schwarz, and S. Mangard. Drama: Exploiting dram addressing for cross-cpu attacks. In *Proceedings of the 25th Conference on USENIX Security Symposium*, pages 565–581. USENIX Association, 2016.

[100] C. J. Plack. *The sense of hearing*. Routledge, 2018.

[101] T. Ranosa. How many times do you unlock your iPhone per day? Here's the answer from Apple. Tech Times. http://www.techtimes.com/articles/151633/20160420/how-many-times-do-you-unlock-your-iphone-per-day-heres-the-answer-from-apple.html, 2016.

[102] E. Ravenscraft. Lastpass authenticator now has a one-button approval option. https://lifehacker.com/lastpass-authenticator-now-has-a-one-button-approval-op-1785138823.

[103] R. B. Research. Number of ATMs worldwide drops for the first time as demand for cash decreases. https://www.rbrlondon.com/wp-content/uploads/2019/05/GA24_Press_Release_200519.pdf, 2019.

[104] S. Rosen and P. Howell. *Signals and systems for speech and hearing*, volume 29. Brill, 2011.

[105] V. Rout. Security issues with android accessibility. https://android.jlelse.eu/android-accessibility-75fdc5810025, 2016.

[106] D. E. Rumelhart and D. A. Norman. Simulating a skilled typist: A study of skilled cognitive-motor performance. *Cognitive Science*, 6(1):1–36, 1982.

[107] Y. S. Ryu, D. H. Koh, B. L. Aday, X. A. Gutierrez, and J. D. Platt. Usability evaluation of randomized keypad. *Journal of Usability Studies*, 5(2):65–75, 2010.

[108] SAASPASS. Two-factor authentication with proximity uses ibeacon bluetooth low energy (ble) to authenticate users instantly. https://saaspass.com/technologies/proximity-instant-login-two-factor-authentication-beacon.html.

[109] T. A. Salthouse. Effects of age and skill in typing. *Journal of Experimental Psychology: General*, 113(3):345, 1984.

[110] T. A. Salthouse. Perceptual, cognitive, and motoric aspects of transcription typing. *Psychological Bulletin*, 99(3):303, 1986.

[111] R. Sandhu. Good-enough security. *IEEE Internet Computing*, 7(1):66–68, 2003.

[112] E. Sejdić, I. Djurović, and J. Jiang. Time–frequency feature representation using energy concentration: An overview of recent advances. *Digital signal processing*, 19(1):153–183, 2009.

[113] L. H. Shaffer. Latency mechanisms in transcription. *Attention and Performance IV*, pages 435–446, 1973.

[114] M. Shirvanian, S. Jarecki, N. Saxena, and N. Nathan. Two-factor authentication resilient to server compromise using mix-bandwidth devices. In *Proceedings of the 21st Annual Network and Distributed System Security Symposium, NDSS*, 2014.

[115] D. Shukla, R. Kumar, A. Serwadda, and V. V. Phoha. Beware, your hands reveal your secrets! In *Proceedings of the 21st ACM Conference on Computer and Communications Security*, pages 904–917. ACM, 2014.

[116] T. Simonite. Apple's 'neural engine' infuses the iPhone with AI smarts. https://www.wired.com/story/apples-neural-engine-infuses-the-iphone-with-ai-smarts/, 2017.

[117] A. Smith. 1 billion accounts are leaked from yahoo's database. https://latesthackingnews.com/2016/12/15/1-billion-accounts-leaked-yahoos-database.

[118] S. W. Smith et al. The scientist and engineer's guide to digital signal processing. 1997.

[119] D. X. Song, D. Wagner, and X. Tian. Timing analysis of keystrokes and timing attacks on SSH. In *Proceedings of the 10th Conference on USENIX Security Symposium*. USENIX Association, 2001.

[120] R. Spreitzer. Pin skimming: Exploiting the ambient-light sensor in mobile devices. In *Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices*, pages 51–62. ACM, 2014.

[121] R. Srinivasa Rao and A. R. Pais. Detecting phishing websites using automation of human behavior. In *Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security*, pages 33–42. ACM, 2017.

[122] StatCounter. Mobile vendor market share worldwide. http://gs.statcounter.com/vendor-market-share/mobile/worldwide.

[123] Statista.com. Frequency of credit card usage among students in the united states in 2015. https://www.statista.com/statistics/524914/frequency-of-credit-card-usage-students-usa/, 2015.

[124] S. Stein. Project soli is the secret star of Google's Pixel 4 self-leak. https://www.cnet.com/news/project-soli-is-the-secret-star-of-googles-pixel-4-self-leak/, 2019.

[125] D. Sukhram and T. Hayajneh. Keystroke logs: Are strong passwords enough? In *Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), 2017 IEEE 8th Annual*, pages 619–625. IEEE, 2017.

[126] J. Sun, X. Jin, Y. Chen, J. Zhang, R. Zhang, and Y. Zhang. Visible: Video-assisted keystroke inference from tablet backside motion. In *Proceedings of the 23th Annual Network and Distributed System Security Symposium, NDSS*, 2016.

[127] F. Tari, A. A. Ozok, and S. H. Holden. A comparison of perceived and real shoulder-surfing risks between alphanumeric and graphical passwords. In *Proceedings of the Second Symposium on Usable Privacy and Security*, pages 56–66, 2006.

[128] P. S. Teh, A. B. J. Teoh, and S. Yue. A survey of keystroke dynamics biometrics. *The Scientific World Journal*, 2013.

[129] L.-H. Teo, B. John, and M. Blackmon. CogTool-Explorer: A model of goal-directed user exploration that considers information layout. In *Proceedings of the 2012 SIGCHI Conference on Human Factors in Computing Systems*, pages 2479–2488. ACM, 2012.

[130] D. Tian, X. Jia, J. Chen, and C. Hu. An online approach for kernel-level keylogger detection and defense. *Journal of Information Science and Engineering*, 33(2):445–461, 2017.

[131] Z. Tong, Q. Ma, S. Zhang, and Y. Liu. Context-free attacks using keyboard acoustic emanations. In *Proceedings of the 21st ACM Conference on Computer and Communications Security*, pages 453–464. ACM, 2014.

[132] S. Verma. Invention story of ATM. `https://www.engineersgarage.com/invention-stories/atm-history`, 2014.

[133] P. Vila and B. Köpf. Loophole: Timing attacks on shared event loops in chrome. In *Proceedings of the 26th Conference on USENIX Security Symposium*, Vancouver, BC, 2017. USENIX Association.

[134] W3school. jquery keydown() method. `https://www.w3schools.com/jquery/event_keydown.asp`.

[135] C. Wang, X. Guo, Y. Wang, Y. Chen, and B. Liu. Friend or foe? Your wearable devices reveal your personal PIN. In *Proceedings of the 11th ACM Asia Conference on Computer and Communications Security*, pages 189–200. ACM, 2016.

[136] D. Wang, Q. Gu, X. Huang, and P. Wang. Understanding human-chosen PINs: Characteristics, distribution and security. In *Proceedings of the 12th ACM on Asia Conference on Computer and Communications Security*, pages 372–385. ACM, 2017.

[137] H. Wang, T. T.-T. Lai, and R. Roy Choudhury. Mole: Motion leaks through smartwatch sensors. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, pages 155–166. ACM, 2015.

[138] W. Wang, A. X. Liu, and K. Sun. Device-free gesture tracking using acoustic signals. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*, pages 82–94. ACM, 2016.

[139] C. Warzel. Here's the cold, hard proof that we can't stop checking our phones. Buzzfeed News. `http://www.buzzfeed.com/charliewarzel/heres-the-cold-hard-proof-that-we-cant-stop-checking-our-pho`, 2013.

[140] D. Weinshall. Cognitive authentication schemes safe against spyware. In *Security and Privacy, 2006 IEEE Symposium on*, pages 6–pp. IEEE, 2006.

[141] S. Wiedenbeck, J. Waters, L. Sobrado, and J.-C. Birget. Design and evaluation of a shoulder-surfing resistant graphical password scheme. In *Proceedings of the working conference on advanced visual interfaces*, pages 177–184. ACM, 2006.

[142] Wikipedia. Doppler radar. `http://en.wikipedia.org/w/index.php?title=Doppler%20radar&oldid=899993071`.

[143] Wikipedia. Network time protocol. `https://en.wikipedia.org/wiki/Network_Time_Protocol`.

[144] C. Wu. *Queueing Network Modeling of Human Performance and Mental Workload in Perceptual-Motor Tasks*. PhD thesis, University of Michigan, Ann Arbor, Michigan, 2007.

[145] Y. Xu, J. Heinly, A. M. White, F. Monrose, and J.-M. Frahm. Seeing double: Reconstructing obscured typed input from repeated compromising reflections. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 1063–1074. ACM, 2013.

[146] Z. Xu, K. Bai, and S. Zhu. TapLogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors. In *Proceedings of the 5th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 113–124. ACM, 2012.

[147] Q. Yan, J. Han, Y. Li, and R. H. Deng. On limitations of designing leakage-resilient password systems: Attacks, principles and usability. In *Proceedings of the 19th Annual Network and Distributed System Security Symposium, NDSS*. Internet Society, 2012.

[148] Q. Yan, J. Han, Y. Li, J. Zhou, and R. H. Deng. Designing leakage-resilient password entry on touchscreen mobile devices. In *Proceedings of the 8th ACM Symposium on Information, Computer and Communications Security*, pages 37–48. ACM, 2013.

[149] G. Ye, Z. Tang, D. Fang, X. Chen, K. I. Kim, B. Taylor, and Z. Wang. Cracking android pattern lock in five attempts. In *Proceedings of the 24th Annual Network and Distributed System Security Symposium, NDSS*, 2017.

[150] Q. Yue, Z. Ling, X. Fu, B. Liu, K. Ren, and W. Zhao. Blind recognition of touched keys on mobile devices. In *Proceedings of the 21st ACM Conference on Computer and Communications Security*, pages 1403–1414. ACM, 2014.

[151] K. Zhang and X. Wang. Peeping Tom in the neighborhood: Keystroke eavesdropping on multi-user systems. In *Proceedings of the 18th Conference on USENIX Security Symposium*, pages 17–32. USENIX Association, 2009.

[152] L. Zhang, S. Tan, and J. Yang. Hearing your voice is not enough: An articulatory gesture based liveness detection for voice authentication. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 57–71. ACM, 2017.

[153] L. Zhang, S. Tan, J. Yang, and Y. Chen. Voicelive: A phoneme localization based liveness detection for voice authentication on smartphones. In *Proceedings of the 23rd ACM Conference on Computer and Communications Security*, pages 1080–1091. ACM, 2016.

[154] M. Zhou, Q. Wang, J. Yang, Q. Li, F. Xiao, Z. Wang, and X. Chen. Patternlistener: Cracking android pattern lock using acoustic signals. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1775–1787. ACM, 2018.

[155] L. Zhuang, F. Zhou, and J. D. Tygar. Keyboard acoustic emanations revisited. *ACM Transactions on Information and System Security*, 13(1):3:1–3:26, 2009.