UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

UNDERSTANDING AND SECURING TYPING PRIVACY IN WIRELESS ENVIRONMENTS

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

DOCTOR OF PHILOSOPHY

BY

EDWIN ALBERT YANG
Norman, Oklahoma
2024

UNDERSTANDING AND SECURING TYPING PRIVACY IN WIRELESS ENVIRONMENTS

A DISSERTATION APPROVED FOR THE
SCHOOL OF COMPUTER SCIENCE

BY THE COMMITTEE CONSISTING OF

Dr. Song Fang, Chair

Dr. Qi Cheng

Dr. Dean Hougen

Dr. Ziho Kang

© Copyright by Edwin Albert Yang 2024
All Rights Reserved.

Acknowledgements

Words cannot express my gratitude to my advisor, Dr. Song Fang for his priceless supervision, continual support, patience, and understanding. I also could not have undertaken this crucial and most important part of my life without my Ph. D. committee members, Dr. Cheng, Dr. Hougen, and Dr. Kang, who generously provided invaluable comments, knowledge, and expertise.

I am also grateful to my colleagues in the OU Cybersecurity lab for their editing help, feedback sessions, and moral support. Thanks should also go to the great staff and graduate assistants in the Department of Computer Science, who inspired and motivated me. I also would like to thank the National Science Foundation (NSF) for funding this research.

I hereby want to dedicate this dissertation to my beloved family. Although we are living on the other side of the globe, their belief in me helped me to overcome my own limitations. I do wish I had made you all proud.

Lastly, I would like to thank God, who gave me strength and courage through all the challenging moments in my career. I am truly grateful for His unconditional love, mercy, and grace.

Abstract

Sensitive numbers play an unparalleled role in identification and authentication. Recent research has revealed plenty of side-channel attacks to infer keystrokes. The common idea is that pressing a key of a keyboard can cause a unique and subtle environmental change, which can be captured and analyzed by the eavesdropper to learn the keystrokes. However, these attacks also require either a training phase or a dictionary to build the relationship between an observed signal disturbance and a keystroke. As acquiring the training data about the victim is often unpractical, this research develops a side-channel attack that does not require training procedures.

This dissertation demonstrates that typing a number creates not only a number of observed disturbances in space (each corresponding to a digit), but also a sequence of periods between each disturbance. Based upon existing work that utilizes inter-keystroke timing to infer keystrokes, we build a novel technique that combines the spatial and time domain information into a spatiotemporal feature of keystroke-disturbed wireless signals. With this spatiotemporal feature, the proposed attack can infer typed numbers without the aid of any training.

Experimental results on top of software-defined radio platforms show that this attack vastly reduces the guesses required for breaking certain 6-digit PINs from 1 million to as low as 16, and can infer over 52% of user-chosen 6-digit PINs with less than 100 attempts. This dissertation also discusses feasible countermeasures that can resist the proposed attack and evaluates them in real-world typing environments.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the digital era, identifying numbers permeate every aspect of our daily lives, particularly social security numbers (SSNs) and personal identification numbers (PINs). These numbers grant access to highly sensitive applications and services, and their disclosure to unauthorized parties can lead to serious consequences. A compromised SSN can enable identity theft in the form of fraudulent credit card accounts [50], access to Medicaid or unemployment insurance benefits [90], or fraudulent tax return filing and return claims [71]. Due to the large unemployment benefit expansion during the COVID-19 pandemic, scammers have filed huge numbers of fraudulent unemployment claims with stolen SSNs [8].

People inevitably type those important numbers into computer systems via a keyboard for identification and authentication under many practical and sometimes public scenarios. For example, we must type in our SSNs to do a credit check when filling out a mortgage/credit card/employment application, or to set up/log into mobile banking accounts [35]. PINs are often required to unlock smartphones or other systems enforcing access control, including smart doors, safe boxes, automated teller machines (ATMs), and point of sale (POS) devices. These circumstances provide attacks with the best opportunity to eavesdrop on these valuable numbers.

Traditional invasive keystroke eavesdropping attacks usually require deceiving the victim's computer system to pre-install malware, e.g., a keylogger [47], which intentionally records and sends everything the victim types to a remote site for the adversary to read. However, such invasive attacks can be defended with anti-malware techniques [73]. Recent research focuses on developing non-invasive keystroke inference attacks [5,16,32,58,61,69,70,85,87,96], which are more surreptitious as they only require passive monitoring of corresponding physical disturbances (e.g., brainwave signals [70], vibrations [69], acoustic emanations [16, 85], motion [96], inter-keystroke timing [5, 87] and wireless signals [3, 4, 32, 58, 61, 108]) in the target's vicinity. Specifically, those methods take advantage of the fact that a keystroke creates a unique variation of the monitored physical disturbance. Such a mapping can be then pre-built and utilized later to infer newly typed content.

Many existing side-channel keystroke inference techniques [16, 32, 96] mainly focus on recovering meaningful English words, instead of numbers, which is simpler because a sequence of keystrokes for a word has to follow the word's structure (i.e., alphabetical combinations defined in a dictionary). Also, multiple such sequences comprising a series of consecutive words must follow language-specific syntax, which further narrows down candidate text. In contrast, inferring numbers is much more challenging, as there is no universal "dictionary" or linguistic relationship for digit sequences. All previous work (e.g., [3, 58, 61, 108]) targeting digits rather than words require a supervised learning process, i.e., the inference system must obtain a large amount of training data (keystrokes on different numbers). Considering that the victim is normally in full control of the keyboard and types in a limited number of digits (i.e., a 9-digit SSN or a 6-digit PIN) within a short time, those training-based methods are clearly not suitable for numbers.

Due to the limitations of existing side-channel keystroke inference techniques, the security risk associated with typing a digit sequence in public places has not particularly been addressed. In this dissertation, we systematically investigate the question: *is a typed number really secure just because it is impractical to collect training data?* We discover a novel type of **W**ireless **I**nference attack targeting **N**umerical **K**eystrokes without requiring training, called *WINK*. Our idea comes from the following two observations.

First of all, we can easily obtain repetition information of each observed disturbance, corresponding to an individual keystroke. We refer to this feature as the *spatial property*. Different disturbances are caused by typing different keys, so the repetition of digits in a digit sequence would be reflected in the repeated ambient disturbances. For example, for a typed 4-digit PIN – "2482", the adversary would observe three disturbances different from each other, and two that match. Such structural information enables the adversary to shrink the candidates of the typed PIN. In this example, the number of average guesses to compromise the PIN decreases by almost 93% compared to traditional brute-force attacks, i.e., from 5,000 to 360. However, the spatial property on its own is far from achieving an effective number inference attack, which requires the number of candidates for the typed number to be small enough to avoid triggering a system lockout. This motivates us to find other useful information disclosed during the typing process to help further shrink the candidates for the typed number.

Second, identifying each physical disturbance also derives the temporal difference between two consecutive keystrokes, or "inter-keystroke timing". Some existing work utilizes such inter-keystroke timing for keystroke inference [5, 87] but requires a training process to collect an enormous amount of data for statistical analysis, which as previously mentioned makes them unpractical. Instead of giving a label (e.g., a key pair) for each inter-keystroke timing, we focus on the inner structure of a sequence of inter-keystroke timings, i.e., the relative size of different elements in the sequence. We refer to such a feature as the *temporal property* of keystrokes, which discloses the relationship among inter-key distances for different key pairs. Specifically, the inter-keystroke timing generally increases with the distance between the typed key pair, which can be immediately obtained based on the keyboard layout. For example, the inter-keystroke timing for typing two same digits is usually smaller than typing two different digits.

By utilizing the *spatiotemporal structure* (i.e., spatial property in conjunction with temporal property) of the observed side-channel information, we develop our training-free and context-free technique to infer the typed number. The foremost challenge is then how to quantify the spatiotemporal structure for observed disturbances and also the corresponding typed number. We customize a quantification scheme that can divide all possible numbers into as many sets as possible to achieve high distinguishability, so that each set has minimum elements on average. As a result, given a quantification result for an observed disturbance sequence, the average candidates for the corresponding typed number can be minimized. Also, with some public information of digits on certain positions (such as the 3-digit area code of an SSN), the search space of possible candidates can be further narrowed down.

As wireless signals are ubiquitous, invisible, and able to propagate under non-line-of-sight (NLOS) conditions, we utilize wireless signals as the target disturbance to capture keystrokes. Particularly, a pre-trained model is not suitable

Table 1.1: Comparison with some existing attacks.

Attack	**Target***			**C1**	**C2**	**C3**	**C4**
	D	**T**	**P**				
EyeTell [15]	●	●	●	✓	✗	✓	✗**
MoLe [96]		●		✗	✗	✗	✗
Compagno *et al.* [16]		●		✗	✓	✓	✓
(sp)iPhone [69]		●		✗	✓	✗	✗
WiPOS [108]	●			✗	✓	✓	✗
Fang *et al.* [32]		●		✓	✓	✓	✗
WINK	●			✓	✓	✓	✓

* D: Digit, T: Text, P: Pattern

** A dictionary is required for text inference.

for wireless-based keystroke inference attacks, as the wireless channel is prone to be influenced by environmental changes, making it impossible to collect generic data and establish a universal model for all scenarios. Therefore, wireless-based keystroke inference attacks [3, 14, 58, 108] all utilize frequent training to cope with the environmental changes, instead of relying on a one-time pre-trained model. Also, all these works require (1) performing user-specific training (i.e., pre-obtaining a large/desirable number of labeled data of the victim) and (2) that each user maintains a consistent typing posture during the training and testing. Both requirements are unlikely to be satisfied in reality, especially for typing sensitive numbers. [108] also performs multi-user training (i.e., collecting the data from many users to build a classifier and using it to predict the typing content of another user whose samples are not included in the training data set) and shows unsatisfactory inference results for victims whose typing habits are different from the users whose data are used for training.

This dissertation further compares *WINK* with the existing keystroke inference attacks by considering the type of the exploited side-channel, target information. Additionally, we consider the following properties. First, as aforementioned, techniques without training requirements (**C1**) are more practical, as they reduce the burden of an attacker. Second, the attack schemes without line-of-sight constraints are more practical (**C2**) as the attacker has more freedom of positioning herself or attack devices (e.g., sensors). Third, non-intrusive (**C3**) attacks that do not need to compromise (e.g., install malware) the victim's devices are more practical, as they do not increase the burden on the attacker. Finally, the attacks do not require long input or dictionary (**C4**) are more practical, as they can infer target information with less information disclosed by the victim. The highlighted differences are depicted in Table 1.1.

1.1 Summary of Contributions

The contributions of this dissertation are summarized below:

- Unlike previous extensive research in inferring keystrokes using labor-intensive training or contextual information, this dissertation identifies a new type of attack that can compromise numerical keystrokes with only the instantaneous wireless data collected during those keystrokes.

- We develop an algorithm to map obtained time series of wireless measurements into a digit sequence by modeling, extracting, and correlating their self-contained spatiotemporal features.

- We carry out extensive real-world experiments, demonstrating that *WINK* can consistently and significantly reduce the search space for each PIN or SSN. Specifically, over half of the 6-digit PINs and 85% of the SSNs can be inferred within less than 100 attempts in real-world settings.

- We design a countermeasure to defend against the proposed attack, and evaluate the effectiveness of the proposed defense technique. Experiment results show that typing extra digits that are carefully chosen can improve PIN entropy.

Chapter 2

Related Work

In this chapter, we review prior side-channel keystroke inference attacks can be mainly divided into the following categories.

2.1 Video-based Attacks

An attacker can stealthily record the typing process of the victim. Though the attacker may not directly observe the typed number in many cases, she can still recover keystrokes through computer vision techniques, e.g., tracking hand movement [84] or touching fingertip [103], and analyzing backside motion of input devices [88]. Recent research even shows that keystrokes can be disclosed via a video capturing the eye movement of the victim [15] or a video call [80]. However, the accuracy of video-based techniques highly depends on the camera's field of view covering the victim, as well as the light condition in the environment.

2.2 Sensor-based Attacks

An attacker may use diverse on-board sensors to infer keystrokes, mainly including motion sensors (e.g., accelerometers and gyroscopes) and microphones. Generally, pressing different keys on a keyboard may generate unique acoustic or motion signals. Most of the existing work (e.g., [12, 16, 17, 64, 66, 67, 69, 92, 93]) require a supervised training process to build a correlation model between each keystroke and corresponding sensing signal. By incorporating the statistical constraints of the English language, [110] utilizes an unsupervised training process.

With the model built during the training phase, the attacker can infer keystrokes during the attack phase. For example, the accelerometer on a smartphone can capture vibrations caused by keypresses on a nearby physical keyboard [69] or onscreen keyboard [12], and such vibrational signals can be used for keystroke inferences. Also, recent research efforts (e.g., [64, 67, 92, 93, 96]) reveal that when a user wears a smartwatch on the wrist and types, the accelerometer or gyroscope built within the smartwatch can track the user's hand movement, which can be further utilized to infer keystrokes. However, in order to get data from sensors on a smartwatch or smartphone, the attacker has to first trick the victim into installing malware onto the smartwatch or smartphone. [16] does not require the attacker to pre-hack the user's device, and it collects acoustic emanations of pressed keystrokes through Voice-over-IP (VoIP, e.g., Skype) calls. However, it only works when the following two conditions are satisfied: first, the attacker and the victim join the same VoIP call; second, the victim types in sensitive information during the call.

Meanwhile, [17] exploits the power consumption of smartphones to infer their passcodes. When the victim types a passcode while the smartphone is connected to a charger, the corresponding voltage fluctuations can be observed on the power line. Such a scenario is practical, as people often use public chargers that are available in various places (e.g., an airport, or a cafe). However, this scheme also requires a training phase to achieve the attack. With a trained CNN classifier, [17] achieves 92.8% accuracy for inferring a 4-digit passcode on the first attempt.

Besides, an attacker may leverage Time Difference of Arrival (TDoA) values to localize keystrokes (e.g., [63, 109]). However, such techniques have three disadvantages. First, they require multiple microphones and accurate time synchronization for the microphones. Second, the victim needs to put her phone very close to the target keyboard. Last, the adversary must also pre-infect the victim's phone with malware, so that the intercepted acoustic signals can be sent back to her.

2.3 Inter-keystroke Timing Based Attacks

It has been widely acknowledged that inter-keystroke timing may leak information about the key sequences being typed. In existing work [5,13,87], a training process is necessary to preestablish the relationship with each key pair and corresponding inter-keystroke timing. Our attack, though, also collects inter-keystroke timing, it eliminates the training requirement and takes advantage of the self-contained pattern of observed inter-keystroke timing sequence instead. Besides, our attack deduces the inter-keystroke timing from the intercepted wireless signals. Such a method of obtaining inter-keystroke timing is more flexible and practical than existing work [5, 13, 87]. For example, [87] learns the inter-keystroke timings of the user's typing from the arrival times of SSH packets, and thus it needs to wait until the victim launches an SSH session. [5] uses cameras to record the typing process and extracts inter-keystroke timing from typing feedback on screens in the form of characters (e.g., $*$ or $\bullet$), making it difficult to attack keyboards that provide no clear graphic feedback (e.g., a dim phone). [13] extracts inter-keystroke timing from the feedback sound when users type, and thus like aforementioned sensor-based attacks, it must pre-install malware on the victim's device to record the acoustic signals.

Besides, [65] proposes user-independent inter-keystroke timing based PIN inference attacks. It requires a training procedure to build a human cognitive model, which can be trained with data not coming from the victim. The cost of this relaxation is that [65] only works for skilled typists who share the universal typing behavioral phenomena.

2.4 Wireless-based Attacks

Recently, many studies have shown the success of leveraging wireless signals to infer keystrokes (e.g., [3, 32, 53, 55, 58, 61, 108]). Compared with other side-channel keystroke inference attacks, wireless-based techniques have three advantages. First, wireless signals are ubiquitous and invisible, making wireless-based attacks quite suspicious and easy to set up. Second, they are non-invasive as there is no need to pre-install malware on the victim's device. Third, unlike video-based or sensor-based attacks, they do not require the victim to be in line-of-sight or close proximity to the keystroke inference system. However, most of those wireless-based keystroke inference techniques ([3, 58, 61, 108]) still require a training process to pre-label the observed wireless signal sample with the corresponding keystroke.

WiPOS [108] infers the typed digits by a victim by utilizing training data obtained from another user. However, it requires the attacker to collect lots of data to build an effective classifier, and it does not provide consistent performance if a target victim's finger movement (i.e., typing style) is quite different from the users who provide training datasets. Meanwhile, [32] removes the training process by exploring the context correlation which is strictly constrained by the spelling and grammar of the English language, and thus it cannot be used for inferring numbers, in which digits are usually randomly combined.

BrokenStrokes [72] detects the presence of specific keywords in long keystroke sequences by only eavesdropping on the keyboard-dongle communication links. First, it converts the received signal strength (RSS) peaks to inter-keystroke timings, and then infers the typed keywords from the timing information. However, this attack is not applicable to keyboards without wireless dongles (e.g., wired keyboard, touch screen device), and requires a training phase to infer specific English words or sentences.

Another work [59] leverages the liquid crystal nematic response effect under mmWave sensing to infer on-screen contents. It requires training in two phases (content-type recognition, and information retrieval), but the training data can be obtained from any user. Also, many systems display nothing or asterisks rather than the typed digits, leading [59] to fail in such scenarios.

ClickLeak [55] exploits wireless CSI with vibration and audio side-channel information to infer the typed PINs. The key idea is exploiting vibration and audio observations to identify the input window (i.e., typing period), and utilize a trained k-Nearest Neighbor (kNN) classifier to infer which keys are pressed by the victim.

One recent work [49] achieves a PIN inference attack using Beamforming Feedback Information (BFI), which is a compressed form of CSI. BFI can be obtained from specific WiFi hardware (higher than WiFi 5). [49] first determines the victim's keystroke with the prior knowledge of the victim (i.e., MAC address), and performs keystroke inference with a trained classifier. Although [49] achieves reliable attack performance, it still requires training data obtained from the victims.

Chapter 3

Wireless Inference of Numerical Keystrokes via Spatiotemporal Analysis

In this chapter, this dissertation proposes a novel numeric keystroke inference attack, *WINK* [102], which utilizes both spatial and temporal features (i.e., spatiotemporal feature) that can be obtained from the side-channel.

3.1 Preliminaries

This section introduces how existing wireless-based keystroke inference attacks work, and presents a general workflow of them. As the proposed attack targets inferring sensitive numbers, this section additionally presents basics about Social Security Number (SSN) which can help us narrow down candidates for that type of typed number.

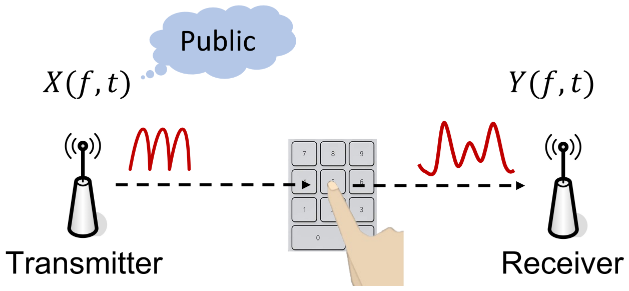


Figure 3.1: General wireless-based keystroke inference setup.

3.1.1 Wireless-based Keystroke Inference

There are emerging research efforts in wireless-based keystroke inference [3, 4, 32, 58]. The common underlying principle is that hand movement during typing changes multipath signals scattered from walls or surrounding objects and may also create new multipath signals. The received signal, as the resultant of all those multipath signals, will be altered accordingly. The impact of a wireless channel on the transmitted signal can be quantified by the *channel state information (CSI)* measurement, which can be in turn used to infer keystrokes.

Orthogonal frequency-division multiplexing (OFDM) encodes digital data on multiple carrier frequencies, and has been widely employed in mainstream WiFi systems such as 802.11 a/g/n/ac. The Channel Frequency Responses (CFRs) obtained from the subcarriers compose the CSI of OFDM. The CFR for a subcarrier with frequency f at time t can be denoted with $H(f,t)$, which can be calculated by transmitting a publicly known preamble of OFDM symbols between the transmitter and the receiver [37]. Let $X(f,t)$ and $Y(f,t)$ represent the transmitted preamble and correspondingly received signal, respectively, for the subcarrier frequency f, as shown in Figure 3.1. An attacker can utilize a transmitter and a receiver to create a radio environment. The transmitter transmits signals that are distorted by the typing activity, while the receiver can quantify such distortion by launching channel estimation. With the received signal and the publicly known preamble, the receiver can compute $H(f,t) = \frac{Y(f,t)}{X(f,t)}$.

The received signal reflects the constructive and destructive interference of multipath signals. Therefore, a certain human activity creates a unique multipath environment and generates a unique pattern in the time series of CSI values, i.e., CSI is popularly utilized to achieve fine-grained activity recognition. For example, CSI can be used for recognizing human identity (e.g., [2,48,82,104,105]), moving humans (e.g., [77, 78]), human vital signs [62], and various human motions or gestures [56,89,91,95].

General Workflow: Existing keystroke inference methods using CSI [3,4,58] usually rely on three phases, including signal pre-processing, training, and testing. The first phase segments the collected CSI time series into a sequence of waveforms, each corresponding to a keystroke, through three steps: (1) noise removal, to make the estimated CSI more accurate; (2) dimension reduction, to find subcarriers which show the strongest correlation with the typing activity; and (3) waveform extraction, to detect the start and end points of CSI time series for a keystroke. The second phase gathers data on CSI waveforms for all keystrokes and trains a classification model, with which the third phase maps each observed unlabelled CSI waveform into the corresponding keystroke.

3.1.2 SSN Basics

A nine-digit SSN is uniquely issued to an individual by the Social Security Administration (SSA) of the United States and usually follows a person over a lifetime. It can be broken into three parts with a format "*AAA-GG-SSSS*": (1) the first

3 digits, known as the area number, indicate the applicant's state of residence before the SSA changed the SSN assignment process to SSN randomization in June 2011 [86], and they no longer reflect the geographical region since then; (2) the next 2 digits, called the group number, break the numbers into convenient blocks (no group contains only zeros); and (3) the last 4 digits, referred to as the serial number, are assigned sequentially from 0001 through 9999. The purpose of an SSN has expanded from tracking earnings for Social Security entitlement and benefit computation at its inception in 1936 [76] to ubiquitous identification throughout government and the private sector nowadays. Consequently, an SSN has become a "skeleton key", which may swiftly open the door to identity theft as mentioned previously.

3.2 Adversary Model

In general, an attacker can control a wireless transmitter (TX) and receiver (RX) pair to launch the attack, as shown in Figure 3.1. The effective distance between the attacker's TX/RX and the victim is determined by the transmit power, antenna gain at TX/RX, as well as the nearby environment. The transmitter can constantly transmit the wireless signal or just whenever typing activity is detected (e.g., via a WiFi packet analyzer [58]).

As this work focuses on inferring numeric keystrokes, we consider typing on either a traditional physical numeric keyboard or an on-screen one. Specifically, three typical layouts of digit keys are discussed, as shown in Figure 3.2: (a) a physical palm-sized numeric keypad with the 7-8-9 keys at the top of other digit keys, which is usually on the far right side of a standard computer keyboard; (b) a physical POS keyboard with the 1-2-3 keys on top; (c) a smartphone's touchscreen

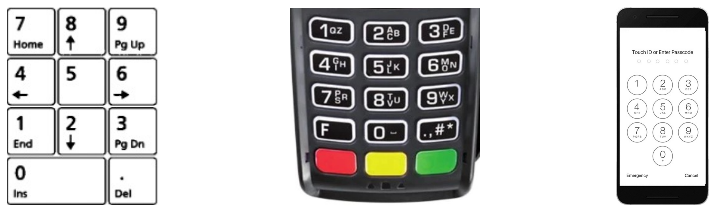
PIN pad with the 1-2-3 keys on top. We assume that the victim types with the same finger of the same hand. In practice, those numeric keypads are operated by one same finger in most cases, as the limited keypad size makes it inconvenient for the keypad to hold two hands simultaneously, and also multi-finger typing is prone to error [60].

3.3 Attack Design

3.3.1 Attack Overview

To launch *WINK*, the attacker first estimates CSI with received signals (as described in Section 3.1.1) and then utilizes the general *pre-processing* phase to divide the estimated CSI time series into individual waveforms. Each waveform corresponds to the action of pressing a key, so we call it *single-keystroke waveform*. Meanwhile, *WINK* also records the start and end times of each extracted keystroke to calculate the *inter-keystroke flight interval*, i.e., the time between releasing the current key and pressing the next key.

In lieu of "train-then-test" paradigms used by existing methods, *WINK* uses the single-keystroke waveforms and inter-keystroke flight intervals in the following two phases: *typing session segmentation* and *spatiotemporal correlation*. The first phase partitions the stream of single-keystroke waveforms into segments, each corresponding to a "typing session" during which the user types continuously without interruption. Occasionally, a user may stop for a while during input to recall the following digits in the number, causing a longer than usual inter-keystroke flight interval not indicative of an inter-key distance. This phase identifies such abnormally long inter-keystroke flight intervals to separate neigh-



(a) Standard numeric pad (b) POS keyboard (c) Touch-screen PIN pad

Figure 3.2: Sketches of typical typing scenarios.

boring typing sessions. The second phase extracts the spatiotemporal feature for each typing session, and correlates it with a digit sequence. The correlation results enable the attacker to derive the mapping between single-keystroke waveforms and keystrokes as well as the mapping between inter-keystroke flight intervals and digit pairs. Such mappings obtained through different typing sessions can be combined to further shrink the candidates of the typed number.

3.3.2 Typing Session Segmentation

Typing session segmentation classifies single-keystroke waveforms and inter-keystroke flight intervals into different categories respectively, and also segments intermittent typing (if any) into multiple typing sessions. Specifically, the following three steps are involved.

3.3.2.1 Spatiotemporal Classification

Different keystrokes usually lead to different key waveforms while the same keystrokes generate highly similar ones. We thus perform spatial classification to cluster different single-keystroke waveforms, and each cluster represents a different keystroke. Meanwhile, when the user's typing finger moves similar distances between two consecutive digit keys during typing, the resultant flight intervals are similar. Temporal classification is then conducted to aggregate comparable inter-keystroke flight intervals into a separate set. The above two classification tasks together form the spatiotemporal classification.

Spatial Classification: To compare two single-keystroke waveforms, *WINK* utilizes the technique of Dynamic Time Warping (DTW), which has been widely utilized to quantify the similarity between two waveforms through dynamic programming [3, 32, 58]. With two single-keystroke waveforms as input, DTW outputs the distance between them. A short distance indicates that the two waveforms are highly similar and originated from typing the same key, while a long distance denotes that they exhibit different patterns and are caused by pressing two different keys.

Temporal Classification: If two inter-keystroke flight intervals are close, they will be classified as a group. Similar to the above spatial classification, temporal classification computes the difference between a pair of inter-keystroke flight intervals. A small difference shows that the two flight intervals are similar and are placed in the same set, while a large difference makes the two flight intervals classified into separate sets.

3.3.2.2 Abnormal Inter-keystroke Flight Interval Detection

Typing a pair of digit keys (k_i, k_{i+1}) consecutively generates four events: the press of k_i at time t_i^s, the release of k_i at time t_i^e, the press of k_{i+1} at time t_{i+1}^s, and the release of k_{i+1} at time t_{i+1}^e. Accordingly, two single-keystroke waveforms can be observed along with their respective start and end times. The flight interval for typing this key pair is thus $I_{i,i+1} = t_{i+1}^s - t_i^e$.

During a consecutive typing period, the inter-keystroke flight interval is highly correlated with the physical distance between the two keys (referred to as inter-

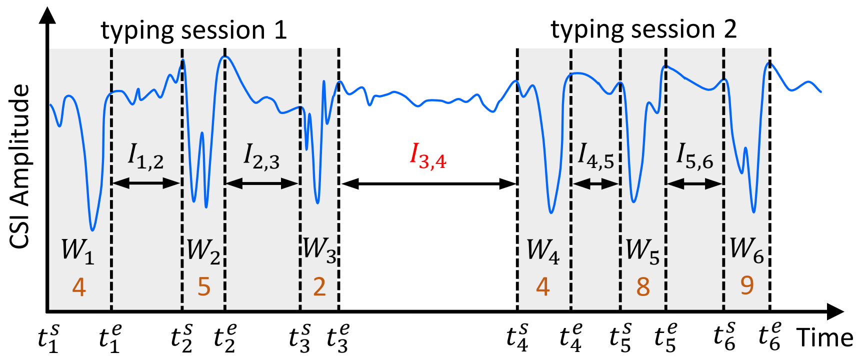


Figure 3.3: Two typing sessions for inputting the number '452489' with a long delay after the first 3 digits.

key distance) on the keypad. On the other hand, as aforementioned, if the user performs intermittent typing due to sudden interruption (e.g., pausing to recall or check the typed numbers), the resultant inter-keystroke flight interval becomes abnormally long and the corresponding interval-distance correlation is broken. Therefore, if an obtained inter-keystroke flight interval is quite long (exceeding the required time for the user to move the finger across the two keys farthest apart on the keypad), it will be regarded as an abnormal flight interval.

The temporal classification outputs N sets, each consisting of similar inter-keystroke flight intervals, which we sort by their mean interval length. We begin by assuming that all flight intervals are normal and perform the remainder of the number inference process. If this assumption is incorrect, we will ultimately not recover any numbers when the process is complete. In that case, we know at least one set of inter-keystroke flight intervals is abnormal, so we label the largest N_a sets as abnormal and try again. We try N_a from 1 to N until we succeed in obtaining candidates for the typed number or exhaust all sets.

3.3.2.3 CSI Session Separation

The whole CSI time series would be divided into typing sessions with detected abnormal inter-keystroke flight intervals. Within each typing session, every single-keystroke waveform and normal inter-keystroke flight interval are grouped together. We call such a group a *CSI session.* Those CSI sessions are then inputted into the phase of spatiotemporal correlation, aiming to build the correlation between a sequence of single-keystroke waveforms and a digit sequence. Figure 3.3 is an example of two typing sessions when the user first types three digits, pauses for a while, and continues to type another three digits. W_i $(i \in \{1, \cdots, 6\})$ is the i^{th} single-keystroke waveform, and $I_{i,i+1}$ is the flight interval between the i^{th} and $(i+1)^{\text{th}}$ keystrokes.

3.3.3 Spatiotemporal Correlation

Spatiotemporal correlation is a function to convert the sequence of single-keystroke waveforms to the typed number. We begin by exploring a common feature to build a correlation between a CSI session and a digit sequence. Next, we consider recovering the digits typed within a period consisting of multiple CSI sessions.

3.3.3.1 Qualifying Spatiotemporal Structure

We aim to find a feature to characterize the spatiotemporal structure of a CSI session. Ideally, this feature can uniquely determine the corresponding sequence of digits. For a sequence with up to n digits, there are $K_{max} = 10+10^2+\cdots+10^n = \frac{10(10^n-1)}{9}$ possibilities in total. A perfect feature classifies the K_{max} candidates into K_{max} subsets, each having one element only, such that an input CSI session can find a unique match based on this feature.

We use the selected feature to divide all K_{max} initial candidates into K subsets. To quantify the distinguishability of this feature, we define a new metric, called *partition rate*, denoted with η, as the ratio between K to K_{max}, i.e., $\eta = K/K_{max}$ ($\eta \in (0, 1]$ as $K \leq K_{max}$). When η is closer to 1, we can obtain smaller subsets on average. Therefore, our goal becomes to develop a feature with high distinguishability that is able to divide all possibilities into the maximum amount of subsets, i.e., to maximize η.

Intuitively, with single-keystroke waveforms, we can determine the number of constituent digits and whether or not any digits in the sequence are repeated. These two pieces of information yield the spatial feature that can be used to partition all candidates of the typed digit sequence. On the other hand, with inter-keystroke flight intervals, we can determine whether or not any inter-keystroke flight intervals appear again. Two inter-keystroke flight intervals belonging to the same set indicate that the two corresponding key pairs have similar inter-key distances. This piece of information yields the temporal feature that can also be used to partition all number candidates. In the following, we evaluate the partition rate when employing different features.

Structural Vector: Let $\mathbf{x} = [x_1, x_2, \cdots, x_n]$ denote a sequence of n elements which can be single-keystroke waveforms, inter-keystroke flight intervals, or digits. We define its *structural vector* as

$$V : \mathbf{x} = [x_1, x_2, \cdots, x_n] \mapsto \mathbf{y} = [y_1, y_2, \cdots, y_n]. \tag{3.1}$$

To construct V, for a sequence of single-keystroke waveforms, we set the elements of the vector $y_1 = 1$ and $y_i = y_j$ $(i > 1, j < i)$ if x_i and x_j are similar

Algorithm 1 Structural Vector Generation

```
 1: procedure S_VECTOR_GEN($[x_1, x_2, \cdots, x_n]$)
 2:     $y_1 \leftarrow 1$
 3:     $y_{max} \leftarrow 1$
 4:     for $i$ in $\{2, 3, \cdots, n\}$ do
 5:         for $j$ in $\{1, 2, \cdots, i-1\}$ do
 6:             if ISSAMESET($x_j, x_i$) then
 7:                 $y_i \leftarrow y_j$                  # belong to a same subset
 8:             else
 9:                 if $j = i - 1$ then                # belong to a new subset
10:                     $y_i \leftarrow y_{max} + 1$
11:                     $y_{max} \leftarrow y_{max} + 1$
12:                 end if
13:             end if
14:         end for
15:     end for
16:     return $[y_1, y_2, \cdots, y_n]$
17: end procedure
```

waveforms as classified during spatial classification (Section 3.3.2.1); otherwise, we set $y_i = \max(y_1, y_2, \cdots, y_{i-1}) + 1$, where $\max(\cdot)$ is a function which returns the maximum among a set of given values. By applying structural vectors to both CSI sessions and digit sequences, we can then extract their spatial and temporal features. Algorithm 1 shows the vector generation procedure, where the pre-defined function IsSameSet(x_j, x_i) determines whether the two elements x_j and x_i are similar (i.e., classified into a same subset), and it returns 1 if yes; otherwise 0.

Spatial Feature Extraction: For a sequence of single-keystroke waveforms $\mathbf{w} = [w_1, w_2, \cdots, w_n]$ of a CSI session, we obtain its spatial feature $\mathbf{s} = V(\mathbf{w})$. Similarly, for a digit sequence, we regard that the same digits are in the same set. We thus obtain its spatial feature accordingly. For example, for an up to 6-digit sequence, there are in total of $K_{max} = \frac{10(10^6-1)}{9} = 1,111,110$ possibilities with brute-force guessing. While using this spatial feature, we can then divide

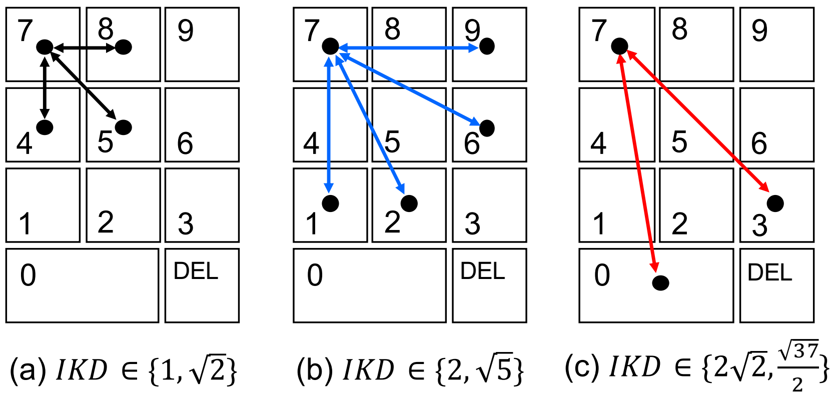

7
8
9
4
5
6
1
2
3
0
DEL
(a) $IKD \in \{1, \sqrt{2}\}$
(b) $IKD \in \{2, \sqrt{5}\}$
(c) $IKD \in \{2\sqrt{2}, \frac{\sqrt{37}}{2}\}$

Figure 3.4: Distances between digit '7' and other keys.

all K_{max} candidates into 277 subsets (i.e., 277 different structural vectors are obtained), such that members of each subset share the same spatial feature. On average, each set has $K_{max}/277 \approx 4,011$ numbers, so an input CSI session would be mapped to one of 4,011 candidates. The partition rate then equals $277/K_{max} = 2.5 \times 10^{-4}$.

Temporal Feature Extraction: Empirically, the inter-keystroke flight interval is generally proportional to the distance between the typed two keys, i.e., inter-key distance (*IKD*). For a sequence of inter-keystroke flight intervals, denoted with $\mathbf{I} = [I_{1,2}, I_{2,3}, \cdots, I_{n-1,n}]$, we can obtain its temporal feature (i.e., structural vector) as $\mathbf{t} = V(\mathbf{I})$. We then introduce how to obtain the temporal feature of a digit sequence. We take a standard number keypad as an example to present the relationship between inter-keystroke flight intervals and digit pairs, while such a relationship can be derived in a similar way for other keypad layouts.

Normally, the horizontal or vertical center-to-center key spacing (referred to as *unit*) between adjacent keys is 19 mm ± 1 mm [51,75]. The movement distance of the typist's finger for typing two keys approximately equals the distance between the centers of these two keys, i.e., *IKD*. Accordingly, all *IKD*s (units) form a set

$\{0, 1, \frac{\sqrt{5}}{2}, \sqrt{2}, \frac{\sqrt{13}}{2}, 2, \frac{\sqrt{17}}{2}, \sqrt{5}, 2.5, 2\sqrt{2}, \frac{\sqrt{37}}{2}, \frac{3\sqrt{5}}{2}\}$. Figure 3.4 shows the distances between digit '7' and other keys. Considering that some *IKDs* are quite close and the resultant inter-keystroke flight intervals may not show obvious difference, we divide all *IKDs* into the following groups (g_1-g_4): if two *IKDs* belong to a same group, the corresponding inter-keystroke flight intervals are categorized into a same subset, and vice versa.

- g_1: $IKD \in \{0\}$;
- g_2: $IKD \in \{1, \frac{\sqrt{5}}{2}, \sqrt{2}\}$;
- g_3: $IKD \in \{\frac{\sqrt{13}}{2}, 2, \frac{\sqrt{17}}{2}, \sqrt{5}, \frac{5}{2}\}$;
- g_4: $IKD \in \{2\sqrt{2}, \frac{\sqrt{37}}{2}, \frac{3\sqrt{5}}{2}\}$.

Based on the *IKDs* of a digit sequence, we can thus derive its temporal feature by calculating the structural vector. Similarly, for an up to 6-digit secret number, we can then obtain 513 subsets in total. Each subset has $K_{max}/513 \approx 2,145$ digit sequences. Correspondingly, the partition rate is $513/K_{max} = 4.62 \times 10^{-4}$.

Inter-keystroke Latency VS. Flight Interval: Inter-keystroke (or press-press) latency has been widely exploited for keystroke inference (e.g., [5, 79, 87, 106]), which equals the time difference between two key presses. One concern is thus why we choose inter-keystroke flight interval to characterize the inter-key distance (*IKD*), rather than inter-keystroke latency. Based on both theoretical and empirical analysis, we find that inter-keystroke flight interval can reflect the corresponding *IKD* more accurately compared with inter-keystroke latency.

We compute the squared Pearson correlation coefficient (SPCC, referred to as ρ^2) [7] to evaluate the degree of linear association between two zero-mean variables (v_1 and v_2), formalized as

$$\rho^2 = \frac{E^2(v_1 \cdot v_2)}{\sigma_{v_1}^2 \cdot \sigma_{v_2}^2}. \tag{3.2}$$

We thus have $\rho^2 \in [0, 1]$. The closer ρ^2 is to 1, the stronger the linear correlation between the two variables is. Lemma 3.1 demonstrates that inter-keystroke flight interval has a more linear relationship with *IKD*.

Lemma 3.1. *$\rho_{f,d}^2 \geq \rho_{l,d}^2$ holds, where $\rho_{f,d}^2$ represents the SPCC between inter-keystroke flight interval f and IKD (denoted with d), and $\rho_{l,d}^2$ denotes the SPCC between inter-keystroke latency l and d, respectively.*

Proof. The inter-keystroke latency is constituted of the duration δ of a single-keystroke waveform and the following inter-keystroke flight interval (i.e., $l = f + \delta$).

With a sequence of n-sample inter-keystroke flight intervals, $f_1, \cdots, f_n$, we calculate the mean value $\bar{f} = \frac{1}{n}\sum_{i=1}^{n} f_i$, and subtract the corresponding mean value to obtain $f_c = f - \bar{f}$. Analogously, we have the mean values $\bar{l}$, $\bar{d}$, and $\bar{\delta}$. After the centralization step, we get the processed values $l_c = l - \bar{l}$, $d_c = d - \bar{d}$, and $\delta_c = \delta - \bar{\delta}$. We then have $\rho_{f,d}^2 = \frac{E^2(d_c \cdot f_c)}{\sigma_{d_c}^2 \cdot \sigma_{f_c}^2} = \frac{E^2(d_c \cdot f_c)}{E(d_c^2) \cdot E(f_c^2)}$.

Note that δ (δ_c) normally relies on the key-press force and individual key design. It is thus a variable independent of f or d (f_c or d_c). Accordingly, we obtain $E(d_c \cdot \delta_c) = E(d_c) \cdot E(\delta_c) = 0$ and $E(f_c \cdot \delta_c) = E(f_c) \cdot E(\delta_c) = 0$. As a

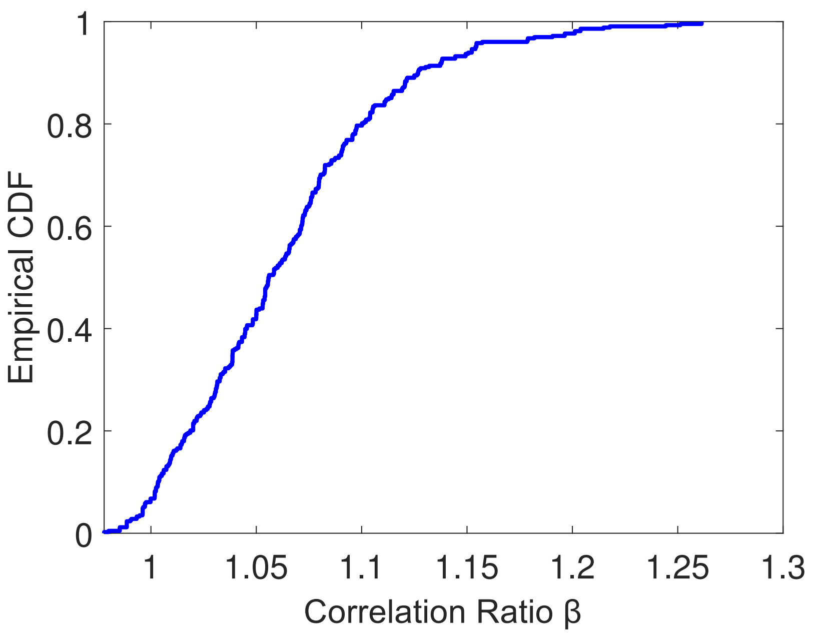


Figure 3.5: CDF of β for all users.

result, we have

$$\begin{aligned}\rho_{l,d}^2 &= \frac{E^2(d_c \cdot (f_c+\delta_c))}{\sigma_{d_c}^2 \cdot \sigma_{f_c+\delta_c}^2} \\ &= \frac{E^2(d_c \cdot f_c + d_c \cdot \delta_c)}{E(d_c^2)\cdot E[(f_c+\delta_c)^2]} = \frac{E^2(d_c \cdot f_c)}{E(d_c^2)\cdot [E(f_c^2)+\Delta]} \\ &\leq \frac{E^2(d_c \cdot f_c)}{E(d_c^2)\cdot E(f_c^2)} = \rho_{f,d}^2,\end{aligned} \tag{3.3}$$

where $\Delta = E[(f_c+\delta_c)^2) - f_c^2] = E(\delta_c^2) + 2E(f_c \cdot \delta_c) = E(\delta_c^2) \geq 0$. □

We further perform real-world experiments to verify Lemma 3.1. We recruit 30 participants (13 self-identified as females and 17 as males) and ask each to type 100 randomly selected 4-digit PINs separately on a standard number pad. For every two successive keystrokes, we record the inter-key distance d and collect corresponding key-press and key-release timestamps to calculate inter-keystroke latency l and flight interval f. For each PIN input, we further compute the

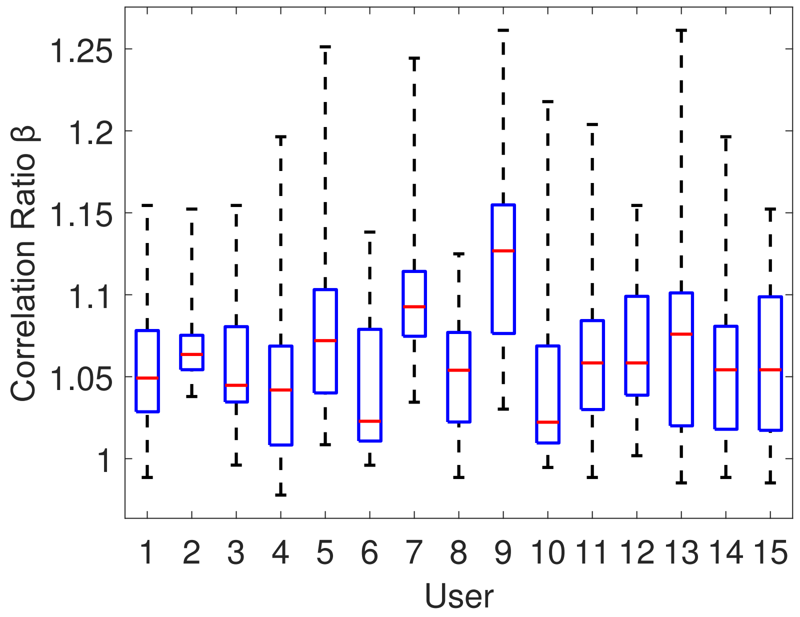


Figure 3.6: User-specific β.

SPCCs $\rho^2_{f,d}$ and $\rho^2_{l,d}$. We introduce a new metric, called *correlation ratio*, denoted with β, to compare two SPCCs, i.e., $\beta = \frac{\rho^2_{f,d}}{\rho^2_{l,d}}$.

Figure 3.5 presents the CDFs of the correlation ratio β for all users. We can see that β ranges from 0.98 to 1.26, and it exceeds 1.0 with a probability of over 93%. Figure 3.6 demonstrates the ranges of β for 10 different users, randomly selected from the pool of all participants. We observe that $\beta \geq 1$ can be achieved in most cases, and even in all cases for some users (i.e., U2, U5, U7, and U9). We have similar observations for the rest 20 users. These results convincingly demonstrate that f is more linearly correlated with d than l, and such a phenomenon is consistent over different users.

Feature Fusion: Feature fusion is the process of combining spatial and temporal features into a spatiotemporal one, which is more discriminative than either input feature. With n keystrokes, we can compute its spatial feature

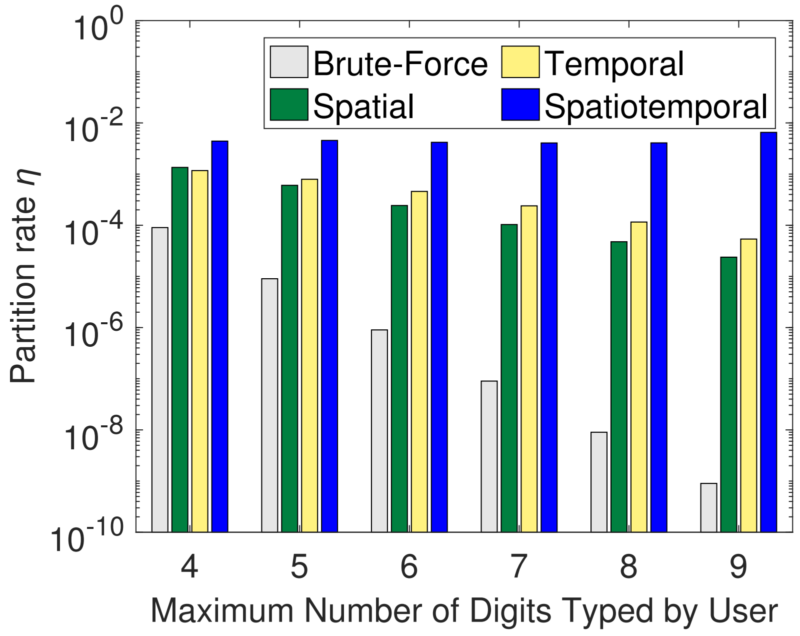


Figure 3.7: Partition rates under different cases.

$\mathbf{s} = [s_1, s_2, \cdots, s_n]$ and temporal feature $\mathbf{t} = [t_1, t_2, \cdots, t_{n-1}]$, enabling us to obtain its spatiotemporal feature $\mathbf{st} = [s_1, t_1, s_2, t_2, \cdots, t_{n-1}, s_n]$. We utilize this spatiotemporal feature to divide all possibilities for an up to 6-digit sequence, obtaining 4,652 subsets in total, which is 15.8 or 8.1 times more than that obtained with the spatial or temporal feature. Thus, each subset has $K_{max}/4,652 = 239$ candidates for the typed digit sequence on average. The corresponding partition rate becomes $4,652/K_{max} = 4.2 \times 10^{-3}$, which is much larger/more discriminative than either the spatial or temporal feature on their own.

Let L_{max} denote the maximum length of the typed digit sequence. Figure 3.7 presents partition rates when we search candidates using traditional brute-force guessing, spatial feature, temporal feature, and spatiotemporal feature, with L_{max} varying from 4 to 9. We see an interesting phenomenon: with L_{max} increasing, the partition rates for brute-force guessing, spatial feature and temporal feature all decrease, indicating that the numeric inference difficulty increases, while the partition rate for spatiotemporal feature does not decrease but gradually increases. This finding demonstrates that the spatiotemporal feature can consistently facilitate narrowing down the search space of the typed digit sequence, and meanwhile, it performs even better for a longer digit sequence.

3.3.3.2 Iterative Joint Decoding

Consider the general case when observing N CSI sessions, denoted by $[C_1, \cdots, C_N]$. Let n_i denote the number of single-keystroke waveforms within C_i ($i \in \{1, \cdots, N\}$). Thus, the initial library for C_i is the set (denoted by $\mathbf{S}_i$) consisting of all possible n_i-digit sequences $S_1, S_2, \cdots, S_{10^{n_i}}$, excluding any S_k ($k \in \{1, \cdots, 10^{n_i}\}$) that is not allowed according to the number composition rules. Our goal is to find a stream of N digit sequences that correspond to the N CSI sessions.

Toward the goal, we first decode each CSI session and then employ iterative joint decoding of multiple CSI sessions. We compare the spatiotemporal feature of C_i to that of each $S_k \in \mathbf{S}_i$, and mark S_k as a candidate if two features are equal. The number of candidates for C_i obtained at this moment is denoted by p_{C_i}. With each candidate, we can build a mapping pair, one mapping between single-keystroke waveforms and digits, and the other between inter-keystroke flight intervals and digit pairs. Different CSI sessions provide extra information (i.e., limitations) for each other, and thus help further shrink the search space. Let M_i denote the concatenation of the first i CSI sessions, and we use $\mathbf{R}_{M_i} = \{R^1_{M_i}, R^2_{M_i}, \cdots, R^{r_i}_{M_i}\}$ to represent its r_i candidates. Starting from the second CSI session, we perform the following steps to decode concatenated CSI sessions. Initially, we set $i = 2$, $M_1 = C_1$, $r_1 = p_{C_1}$, and $\mathbf{R}_{M_1} = \mathbf{S}_1$.

(1) We concatenate the current CSI session with all previous ones, i.e., $M_i = M_{i-1}||C_i$. Thus, $R^u_{M_{i-1}}||S_v$ ($u \in \{1, \cdots, r_{i-1}\}$, $v \in \{1, \cdots, p_{C_i}\}$) could be a potential candidate for the newly concatenated CSI session).

(2) For each $R^u_{M_{i-1}}||S_v$, we compare mappings obtained from $R^u_{M_{i-1}}$ and S_v. If their mapping sets have any contradictions (i.e., two single-keystroke waveforms within the same subset map to different digits, or two different digit pairs share the same subset of inter-keystroke flight intervals while their *IKDs* do not belong to the same group), we then rule out this candidate; otherwise, we mark it as a candidate for M_i, and meanwhile merge all single-keystroke waveform/digit and inter-keystroke flight interval/digit pair mapping information as the new mapping set.

(3) We increment i and jump to step (1).

We take the inference of a 6-digit PIN (937357) as an example. A user types this PIN on a standard number pad with two typing sessions. The user inputs the first three digits (937) and the last three digits (357) in the first and second typing sessions, respectively. We denote the spatiotemporal features corresponding to the two typing sessions by $f_1 = [d_1, t_{1,2}, d_2, t_{2,3}, d_3]$ and $f_2 = [d_4, t_{4,5}, d_5, t_{5,6}, d_6]$, where d_i denotes the i^{th} ($i \in \{1, \cdots, 6\}$) observed single-keystroke waveform and $t_{j,j+1}$ is the calculated inter-keystroke flight interval between the j^{th} and $(j+1)^{\text{th}}$ ($j \in \{1, \cdots, 5\}$) keystrokes.

Due to the aforementioned consistency between spatial features for the same digit, d_2 and d_4 are similar (so are d_3 and d_6). Also, considering the stability between temporal features for close inter-key distances, $t_{4,5}$ and $t_{5,6}$ are close. As a result, we have $f_1 = [1, 1, 2, 2, 3]$ and $f_2 = [1, 1, 2, 1, 3]$. We pre-compute the spatiotemporal feature for each possible 3-digit sequence. By comparing each with f_1 and f_2, we obtain 216 and 288 candidates for the first and second typing sessions, respectively. Each candidate implies a mapping between single-keystroke waveforms and digits, as well as a mapping between inter-keystroke flight intervals and digit pairs. To consider both typing sessions simultaneously, we concatenate each candidate for the first typing session and each for the second one, generating $216 \times 288 = 62,208$ combinations. For some combinations, the mapping information for both typing sessions may contradict each other and we rule out such combinations as candidates for the typed PIN. For example, one combination "937354" is removed as a single-keystroke waveform (d_3 or d_6) maps two different digits (7 and 4). Consequently, only 16 combinations (including the correct PIN) survive as final candidates for the typed PIN. Such performance corresponds to a 62,500-fold improvement compared with the traditional brute force attack which provides 10^6 candidates.

3.3.4 Impact of Non-numeric Keystrokes

Usually, when a user inputs a number, there is no need to use any non-numeric keys. A typical example is an iOS passcode. After typing a 4- or 6-digit PIN on an iOS device (a passcode by default was 4 digits prior to iOS 9, and 6 thereafter [34]), the device would automatically initiate authentication. However, on certain occasions, we may need to use non-numeric keys, including the OK/Enter key, and Backspace/Delete key. For example, if a user customizes an iOS passcode whose length is neither 4 nor 6, after typing this passcode, the user has to press the OK key to get authenticated. Also, to correct typed digits by mistake, the Delete key comes in handy.

Handling OK Key: Since the OK key (if the typist uses it) is always the last key to be pressed, we can regard the last keystroke as it. Thus, we just launch *WINK* by processing the CSI data stream corresponding to other keystrokes.

Handling Delete Key: We consider the most frequent cases when the typist presses the Delete key once to correct one digit or successively for multiple digits. Let L and L' denote the length of the target secret number and the number of observed single-keystroke waveforms, respectively. Accordingly, the Delete key is pressed for $\sigma = (L' - L)/2$ times. We then search for single-keystroke waveforms that appear σ times among all observed ones, excluding the first and the last single-keystroke waveforms (as pressing the Delete key usually does not happen at the beginning or end). Such single-keystroke waveforms are potential candidates for the Delete key. We thus associate the Delete key with one potential candidate in turn until exhausting all candidates. For each association, the Delete-key-labelled single-keystroke waveforms, together with the ahead σ single-keystroke waveforms (corresponding to deleted input), divide the original single-keystroke waveform sequence into two parts. We perform *WINK* for both parts to infer the typed number.

3.3.5 CSI Error Handling

Wireless noise may make pre-processed CSI inaccurate and cause spatiotemporal classification errors, leading to no valid candidates for the typed number. Since the accuracy of inter-keystroke flight interval depends on the detection accuracy for the start and end points of corresponding single-keystroke waveforms, we thus just discuss the cases when spatial classification error happens.

Due to such classification errors, we usually either obtain invalid results or have no candidate at all. The latter case (i.e., failure of inference) is straightforward, signaling the existence of errors; the first case, however, cannot determine whether the CSI error happens or not until all candidates have been tested (i.e., if a candidate passes the authentication, the correct one is found, and thus no error happens). However, empirically, we find that the first case rarely happens. This is because CSI errors often bring exclusive spatial and temporal features, causing the phase of spatiotemporal correlation to output no candidates.

To handle CSI errors, we develop a heuristic algorithm by guessing and removing erroneous single-keystroke waveforms. Specifically, we assume there are e_s erroneous single-keystroke waveforms among a total of L key waveforms. Thus there are ${}^{L}C_{e_s}$ different error cases. Each erroneous waveform would be marked as undetermined (i.e., ranging from 0 to 9) together with neighboring inter-keystroke flight intervals, and would not be used for calculating the spatiotemporal feature. *WINK* is then performed based on the newly calculated spatiotemporal feature. The returned candidates (if any) combine with possibilities for erroneous key waveforms to form the final candidates for the typed number. We can start with $e_s = 1$ and try each corresponding error case until the obtained candidates have the correct typed number.

3.4 Experimental Evaluation

We implement *WINK* using Universal Software Radio Peripherals (USRPs). The prototype system consists of a transmitter (Tx) and a receiver (Rx). Each node is a USRP X300 equipped with a CBX daughterboard [22]. Tx and Rx are placed at opposite positions relative to the keyboard. There is a 10 cm thick cubicle divider between either of them and the keyboard, so that both are not within the line-of-sight of the target user. The distance between Rx and the keyboard is 2 m, while that between Tx and the keyboard is 50 cm. Rx extracts CSI from the received signals to infer numeric keystrokes. We investigate three different types of numeric keyboards, i.e., a standard physical number pad, a typical POS keypad, and a touchscreen one. We put the typing device on a flat table. In this section, we let a single user perform experiments, while in Section 3.5, we consider more typists in a real-world user study.

Metrics: We calculate *entropy* to measure the number strength against brute-force attacks. Suppose there are m candidates for a number X and let $x_i (i \in \{1, 2, \cdots, m\})$ denote one of them. The X's entropy can be then calculated as $H(X) = -\sum_{i=1}^{m} P(x_i) \cdot \log_2 P(x_i)$, where $P(x_i)$ is the probability that $X = x_i$ holds.

To investigate the security inequality of a group of numbers with the same length, we employ the *Gini coefficient* [20], which is most commonly used in economics to measure the inequality among levels of income. We consider a group of N numbers, each with l digits. The average of all N numbers' entropies is represented by $\bar{E} = \frac{\sum_{i=1}^{N} E_i}{N}$, where E_i denotes the entropy of the i^{th} number. We then derive the Gini coefficient G_l for those l-digit numbers as

$$G_l = \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} |E_i - E_j|}{2N^2 \bar{E}}. \tag{3.4}$$

The value of G_l varies from 0 to 1, where 0 indicates perfect equality (when all entropies are the same) and 1 depicts perfect inequality (when one number has a positive entropy while the rest of entropies are 0, leading to $G_l = \frac{N-1}{N} \approx 1$, where $N \gg 1$).

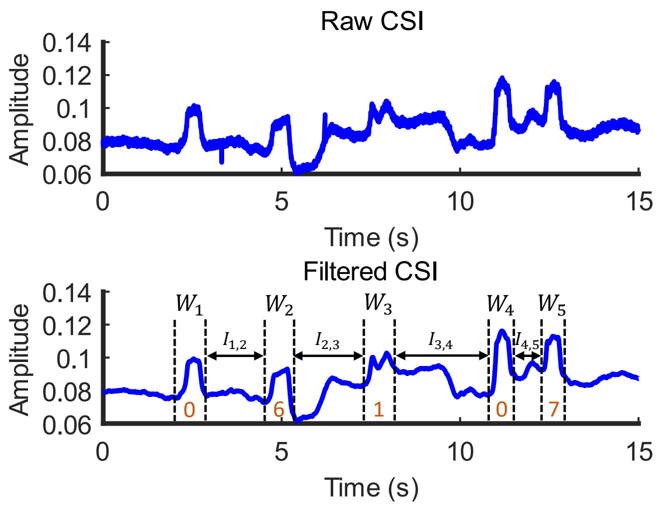


Figure 3.8: CSI for the PIN '06107'.

3.4.1 Case Study

We first demonstrate an example, in which the user types a PIN "06107" on an iPhone 11 Pro Max passcode keypad. We use the same pre-processing methods with existing techniques [3, 32, 58], including noise removal, dimension reduction, and keystroke waveform extraction. To sanitize the CSI data, a weighted moving average filter [57, 98] is applied. Next, we utilize Principal Component Analysis (PCA) [83] to refine the most representative components influenced by keystrokes from CSI collected at all subcarriers. Finally, we extract the corresponding waveform for every single keystroke.

Figure 3.8 presents the raw and filtered CSI time series. We observe five single-keystroke waveforms (W_1 to W_5) and four inter-keystroke flight intervals ($I_{1,2}$ to $I_{4,5}$). W_1 is highly similar to W_4. Meanwhile, either of them and the rest three are different from each other. Accordingly, the spatial feature can be denoted with $[1, 2, 3, 1, 4]$. Also, as $I_{4,5} < I_{1,2} \approx I_{2,3} < I_{3,4}$, the temporal feature can be denoted with $[2, 2, 3, 1]$. By fusing the spatial and temporal features, the phase

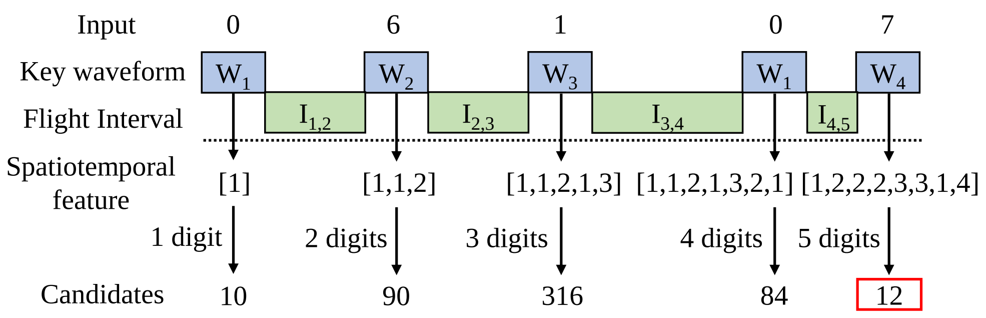


Figure 3.9: Variation of the number of inferred candidates.

of spatiotemporal correlation outputs 12 candidates. Thus, *WINK* reduces the maximum attempts required for breaking a 5-digit PIN to just 12, compared with the brute-force attack which needs 10^5 times, i.e., the PIN entropy is decreased from $5\log_2 10 = 16.61$ bits to $-\sum_{i=1}^{12}\frac{1}{12}\log_2\frac{1}{12} = 3.59$ bits.

Figure 3.9 presents the variation of the amount of inferred candidates as more digits are typed in. We see initially, the number of candidates increases with more key waveforms and flight intervals being processed. This is mainly because the original search space for a longer digit sequence is larger. However, with a richer spatiotemporal feature, the number of candidates dramatically decreases to 84 for 4 digits and 12 for 5 digits, i.e., the speed of shrinking the search space exceeds that of original search space growth.

3.4.2 PIN Inference

To balance security and usability, most authentication systems allow PINs with 4 to 6 digits [52]. To approximate user choices of PINs, we extract leaked real-world PINs with 4 to 6 digits from the *RockYou* database [18], which is widely used in PIN security research (e.g., [9, 94]). For every PIN length, we obtain 100 samples, asking the user to type each extracted PIN separately on the iPhone 11 Pro Max keypad. We launch *WINK* and compute PIN entropies. For each PIN

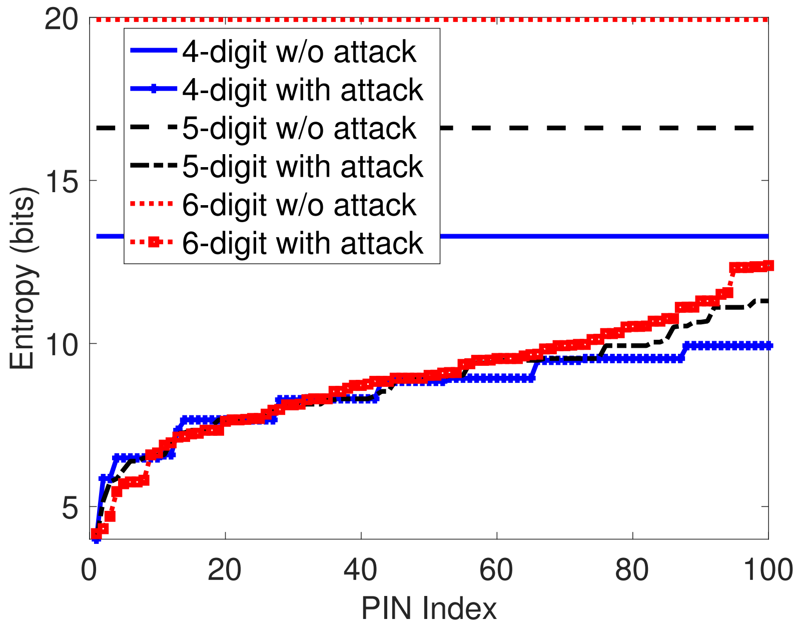


Figure 3.10: Entropy distribution for PINs with different lengths.

length, we sort the PINs in ascending order of the entropies and index them from 1 to 100 in increments of 1. Figure 3.10 shows the PIN entropy distribution with and without applying the proposed attack. We have three major observations.

First, with *WINK*, the search space of the typed PIN with different lengths is significantly shrunk. The attacker decreases the entropy of a 6-digit PIN from 20.0 bits to as low as 4.0 bits, vastly reducing the maximum brute-force attempts for breaking the PIN from 1 million to just 16. Overall, more than 10% of the selected 6-digit PINs can be inferred with an average of fewer than 50 trials.

Second, entropies vary for PINs with the same length. For example, the entropy of an extracted 6-digit PIN ranges from 4.0 to 12.3 bits, which means the average amount of brute-force trials required to guess such a PIN varies from 8 to 2,521. We compute the Gini coefficients (G_4, G_5, and G_6) for each PIN length (4-6), and obtain $G_4 = 0.33$, $G_5 = 0.47$, and $G_6 = 0.57$. To facilitate understanding the Gini coefficient values we get in this experiment, we note that the Gini coefficient for income inequality in the United States in 2022 was 0.488 [11]. These results demonstrate there is notable security inequality among PINs of the same length, and a longer PIN length may introduce more severe security inequality.

Third, longer PINs provide a little increase and sometimes even decrease in security, illustrated by the similar entropy distributions of PINs with different lengths. Longer PINs provide the attacker with a richer spatiotemporal structure, which shrinks the search space more quickly. Specifically, *WINK* lowers entropy by an average of 6.8 bits for 4-digit PINs, 9.3 bits for 5-digit PINs, and 13.4 bits for 6-digit PINs. On average, our attack makes breaking a 6-digit PIN become easier than brute-forcing a 3-digit PIN, while inferring a 4- or 5-digit PIN is reduced to brute-forcing a 2-digit PIN.

3.4.2.1 Impact of PIN Blocklist

Modern authentication systems usually implement a blocklist containing weak PINs. When a user selects a blocklist PIN, the system prompts a warning to suggest or enforce choosing a non-blocklist one. The study [68] reveals the iOS blocklist of passcodes, including 274 4-digit and 2,910 6-digit PINs. We compare these PINs with the most vulnerable 10% PINs against our attack obtained above, and find no overlap. Thus, new weak PINs revealed by *WINK* should be included in the blocklist to improve the minimum PIN security. Also, if the system disallows blocklist PINs, *WINK* can use such information to further shrink the search space by winnowing out candidates appearing in the blocklist. To minimize the negative impact of extending blocklists, for each extension, we should then include all most vulnerable PINs sharing the same spatiotemporal feature

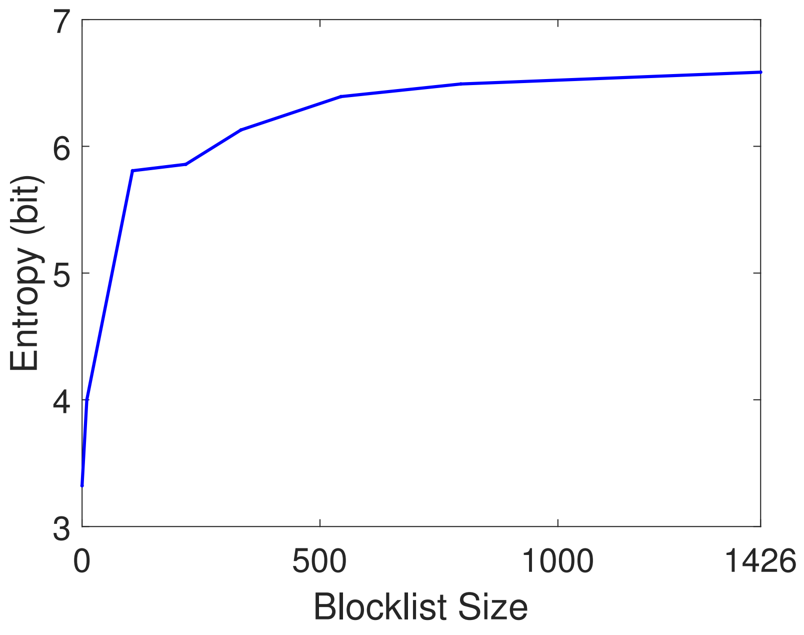


Figure 3.11: Impact of blocklist size on minimum PIN entropy.

in the blocklist. As such inclusion has no additional impact on remaining PINs (whose spatiotemporal features are different from those of the included PINs), the maximum PIN entropy would not be affected while a higher minimum/average PIN entropy can be achieved.

To identify the impact of blocklist size on the minimum and average PIN strength, we conduct simulated attacks for 4-digit PINs as an example. We first define *WINK* vulnerable PINs, where they can be inferred within 90 attempts ($<$ 6.5 bits entropy). Thus, we include 1,426 4-digit PINs (14.26%) in our blocklists. To quantify the impact of blocklist size on PIN security, we measure the minimum and average PIN entropy before and after applying different sizes of blocklists. We increase the size of the blocklist from 0 to 1,426. For each increment, we include a group of PINs that share the same spatiotemporal property that presents the least number of candidates (i.e., the lowest entropy).

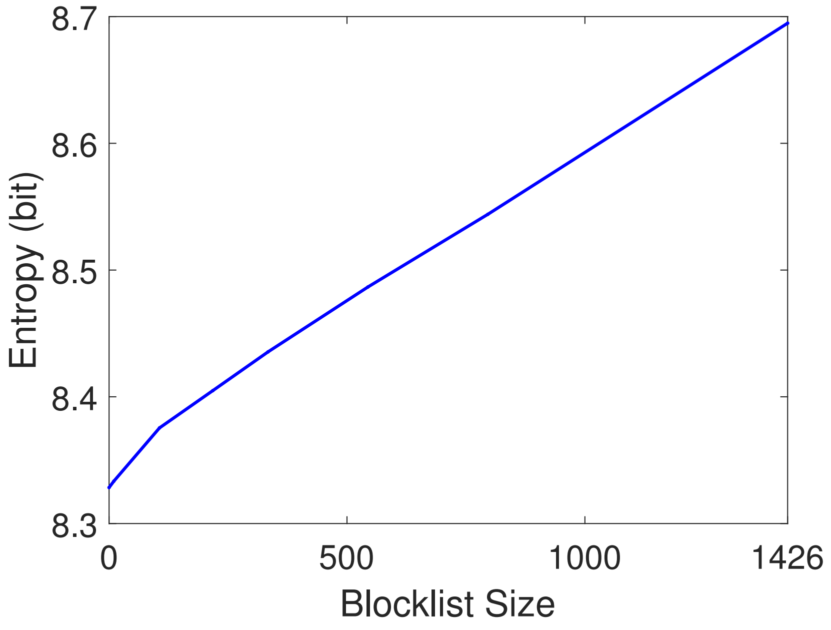


Figure 3.12: Impact of blocklist size on average PIN entropy.

Figure 3.11 demonstrates minimum PIN entropy for different blocklist sizes. First, we can see that the minimum entropy is proportionally increased to the size of the blocklist. As we gradually include more PINs with vulnerable spatiotemporal properties in the blocklist, the minimum entropy is rapidly increased for 0-106 blocklist sizes (3.32-5.81 bits), and then presents significantly reduced increment for blocklist sizes greater than 106. This is because the most vulnerable PINs that present significantly lower entropies than other PINs are already included in the blocklist. The 106 PINs consist of two groups, where they present 3.32 bits and 4.0 bits of entropies. Thus, we can see that the minimum entropy is increased to 5.81 bits after blocklisting the 106 PINs, and the effectiveness of increasing blocklist size afterward is decreased.

Meanwhile, Figure 3.12 shows that the average PIN entropy is increased according to the size of the blocklists. We note that including one group of vulnerable PINs in the blocklist does not affect the individual entropy of the remaining PINs, as the spatiotemporal properties between the blocklisted PINs and the remaining PINs are independent. Based on the result, we recommend adding the 106 *WINK* vulnerable PINs to the existing blocklists. The cut-off point can be adjusted depending on the size of the existing blocklists and usability.

3.4.2.2 Discussion on PIN Strength

Generally, a PIN whose spatiotemporal feature divides all PIN possibilities into a maximum number of subsets would have the strongest strength against *WINK*. Considering the spatial domain, such a PIN should have no digit repetition, as the corresponding spatial feature discloses the least information. Considering the temporal domain, the information disclosed via the temporal feature depends on the IKD sequence of the digits constituting the typed PIN. The strongest PINs, in this respect, would be in the largest temporal subset, i.e., be in the largest group of digit sequences that share the same temporal feature.

3.4.3 SSN Inference

As aforementioned, SSNs issued before June 2011 (without SSN randomization) follow a determined structure (e.g., the first three digits denote the area number assigned by geographical region). Inferring SSNs issued after June 2011 is like inferring a 9-digit PIN.

3.4.3.1 Without SSN Randomization

We select two states from each of the west, middle, and east of the United States, and obtain California (CA), Oregon (OR), Iowa (IA), Texas (TX), New York

Table 3.1: Degrees of security inequality for SSNs with the same area code and SSNs issued within the same state.

State	California					Oregon				
Area Code	546	550	559	561	573	540	541	542	543	544
G_{area}	0.61	0.59	0.52	0.44	0.57	0.52	0.50	0.57	0.55	0.49
G_{state}	0.77					0.54				
State	Texas					New York				
Area Code	449	452	455	463	467	057	088	109	113	126
G_{area}	0.54	0.56	0.52	0.50	0.52	0.55	0.58	0.62	0.57	0.63
G_{state}	0.58					0.60				
State	Iowa					Michigan				
Area Code	479	480	481	484	485	364	367	373	378	384
G_{area}	0.51	0.57	0.49	0.60	0.47	0.54	0.55	0.51	0.55	0.56
G_{state}	0.52					0.59				

(NY), and Michigan (MI). Different states have different 3-digit area codes and a state may have one or multiple area codes. Such information is public [1]. For example, Wyoming has only one area code 520 while area codes 362-386 (i.e., 25 possibilities) are allocated for Michigan. Thus, knowing which state the user comes from, the area code range of the target SSN can be queried.

3.4.3.2 Same-state SSNs

We take Michigan as an example, and randomly select five allocated area codes (364, 367, 373, 378, and 384). With each, we construct 100 SSNs randomly. We let the user type each SSN in a typing session on a standard number pad. Figure 3.13 plots the empirical cumulative distribution functions (CDFs) of the SSN entropies. We observe that SSNs with the same area code exhibit different levels of security. For example, SSNs prefixed by 364 have entropies ranging from 1 to 7.6 bits. For the SSN "364-93-4632", *WINK* outputs only two candidates (i.e., the correct one and a wrong one "364-93-4635"). Specifically, 60% of SSNs with area codes 364 and 367 have entropies below 4.9 and 4.7 bits, respectively,

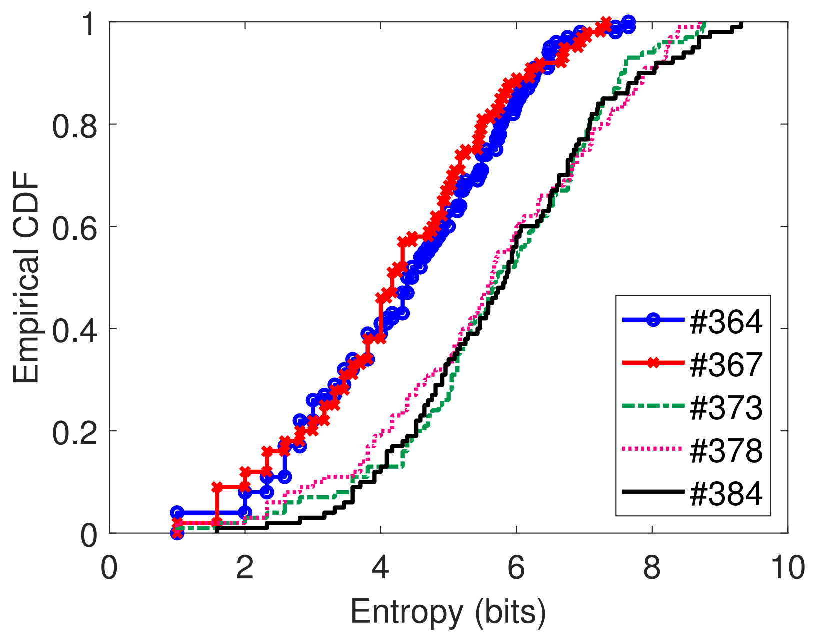


Figure 3.13: CDFs of SSN entropy in the same state (Michigan).

indicating the maximum brute-force attempts required for compromising them are just 30 and 27. Also, *WINK* significantly reduces the search space consistently for all area codes, and obtains entropies ranging from 1 to 9.3 bits. In contrast, traditional brute-force attacks require guessing roughly 25×10^6 times for an SSN assigned in Michigan. As neither the middle two digits nor the last four digits of an SSN can be all zeros, the exact number is $25 \times (10^6 - 10^4 - 10^2 + 1) = 25 \times 989,901$, equivalent to an entropy of 24.6 bits.

3.4.3.3 SSNs across Different States

We randomly generate 100 SSNs allocated for every state, and then let the user type each SSN separately on a standard number pad. Figure 3.14 plots the CDFs of corresponding SSN entropies. We see that our attack consistently decreases SSN entropies, and different states have different entropy ranges. Specifically,

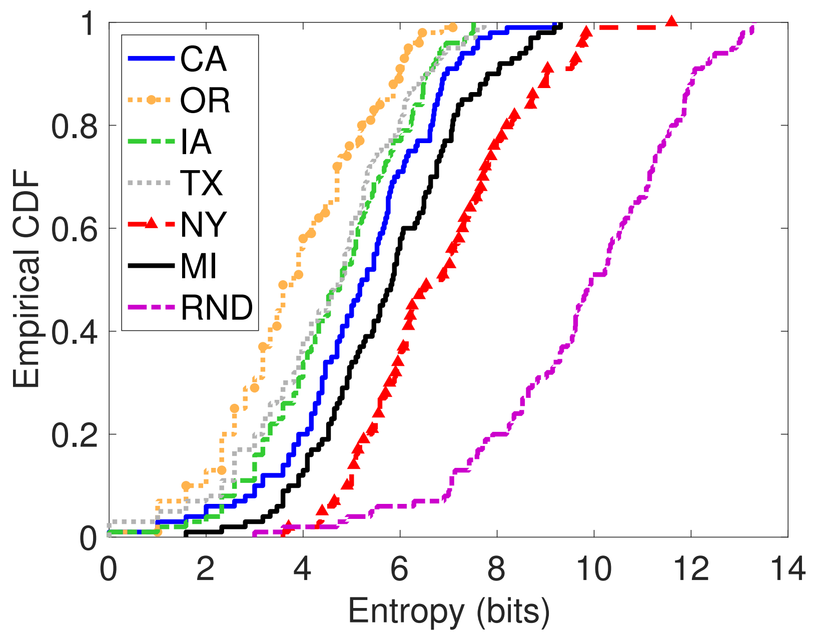


Figure 3.14: CDFs of SSN entropy for different states.

85% of the chosen SSNs in Oregon have an entropy of less than 5.6 bits, indicating that they can be inferred with an average of fewer than 25 attempts. However, such a ratio equals just 27% for New York.

We calculate Gini coefficients (G_{area}) for SSNs with the same area code, and also overall Gini coefficients (G_{state}) for SSNs issued within the same state. Table 3.1 presents the values of G_{area} and G_{state} for different states. We see that no matter from geographical region-wide or state-wide SSNs, they exhibit quite serious security inequality (with Gini coefficients ranging from 0.44 to 0.77). Also, for different area codes, "126" in NY and "561" in CA demonstrate the highest and lowest Gini coefficients (i.e., 0.63 and 0.44).

With SSN Randomization: In this case, an SSN can have any first 3-digit codes except 000, 666, and 900-999. With traditional brute-force attacks, $(10^9 - 102 \times 10^6)/2$ attempts are required on average, i.e., the SSN entropy equals

Table 3.2: PIN entropy under different environments.

Environment	Antenna	Entropy (bits)		
		Minimum	Maximum	Mean
Quiet	Directional	2.0	10.3	8.1
	Omni-directional	3.0	10.8	8.7
Noisy	Directional	5.1	10.6	9.0
	Omni-directional	5.1	12.5	9.6

29.7 bits. Figure 3.14 also presents the CDF of entropies for SSNs assigned via SSN randomization (RND) with the same experimental setting. We observe that SSN randomization increases the SSN entropies overall compared with the previous SSN assignment process, while our attack still greatly shrinks the search space compared with traditional brute-force attacks. Over 7% of SSNs can be inferred with an average of 50 attempts. The Gini coefficient for the selected SSNs equals 0.57, again indicating the severe inequality of SSN security even when SSN randomization is employed.

3.4.4 Robustness to Influential Factors

To evaluate the impact of each influential factor, we employ 100 randomly selected 6-digit PINs from the RockYou password dataset and ask the user to type them, once per PIN, under each situation.

3.4.4.1 Impact of Environment

We test *WINK* to infer PINs inputted on an iPhone 11 Pro Max under two different environments: (a) quiet one where there is no movement of other users, and (b) noisy one where other users walk around. Also, we compare the performance of an omni-directional VERT2450 antenna [24] and a directional LP0965 antenna [23] focusing the energy towards the target of interest. For a quiet environment, omni-directional and directional antennas present 96.7% and 98.2% of CSI fragmentation success rate (i.e., the ratio of successfully segmented single-keystroke waveforms to the total number of keystrokes performed by the user), respectively, and 92.2% and 97.2% for a noisy environment. Table 3.2 shows the obtained PIN entropies in different environments. We can see that *WINK* works under both environments regardless of the antenna type. It lowers PIN entropy by at least 7.5 bits compared to traditional brute-force attacks (in which a 6-digit PIN has 20.0 bits of entropy). In a quiet environment, both antennas present similar entropy ranges due to low CSI interference, while in a noisy environment, the directional antenna presents a slightly lower mean entropy than the omni-directional one, demonstrating the directional antenna effectively reduces the effects of the nearby human movement. Consequently, *WINK* employs directional antennas for better inference performance.

3.4.4.2 Impact of Keyboard Layout

We test three popular layouts of numeric keypads with similar sizes, including (1) iOS passcode keypad (77.8×158 mm displayed on iPhone 11 Pro Max), (2) 3×4 number pad (75×94 mm) on the far right of a standard keyboard (DELL KB216T [19]), and (3) 3×4 POS keypad (53.5×80 mm for the model YD541), which we refer to as "Phone", "Standard", and "POS", respectively. Figure 3.15 presents the corresponding PIN entropies. We can observe that regardless of input layout, our attack decreases the entropies of different PINs in varying degrees. For iOS passcode and POS keypads, their PIN entropy ranges (i.e., 4.7-12.4 bits and 4.1-12.3 bits) are quite close. This is due to the high similarity of their digit arrangement on keypads (with the 7-8-9 keys two rows above the 1-2-3 keys). As the number pad on a standard keyboard has a different key arrangement (with

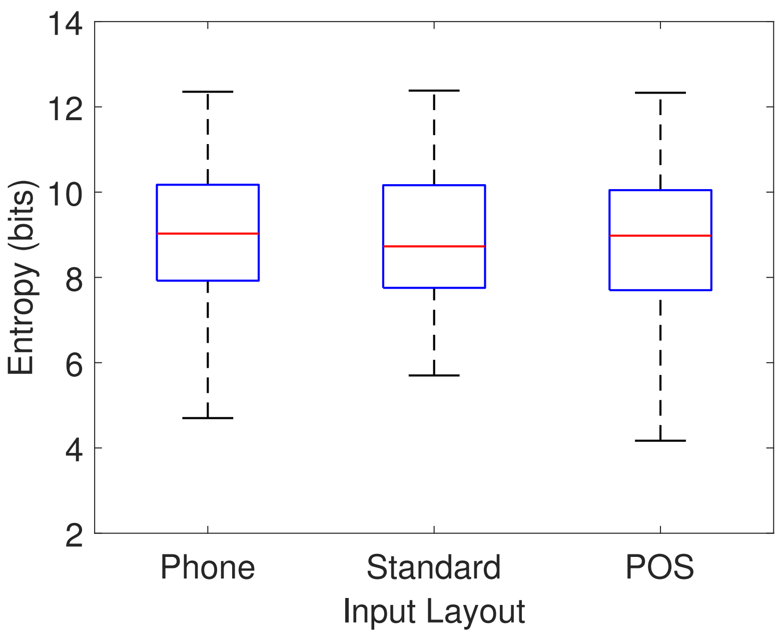


Figure 3.15: Impact of keypad layout.

the 1-2-3 keys on top and the 7-8-9 keys on the third row), its PIN entropy range (i.e., 5.7-12.3 bits) is slightly different. Also, for “Phone”, “Standard”, and “POS”, the Gini coefficients are 0.57, 0.58, and 0.57, respectively, implying a small difference in security inequality brought by input layout.

3.4.4.3 Impact of Keypad Size

Even for the same input layout, the keypad size may differ. We test three iOS devices with different keypad sizes, i.e., iPhone 11 Pro Max (6.5-inch display), iPhone 12 Mini (5.4-inch display), and iPhone SE (4.7-inch display), which we refer to as “Large”, “Medium”, and “Small”. Figure 3.16 shows the resultant PIN entropies. We see that for all keypad sizes, our attack makes breaking PINs much easier than traditional brute-force attacks. With the key size increasing, the mean PIN entropy slightly decreases. This appears due to the fact that a

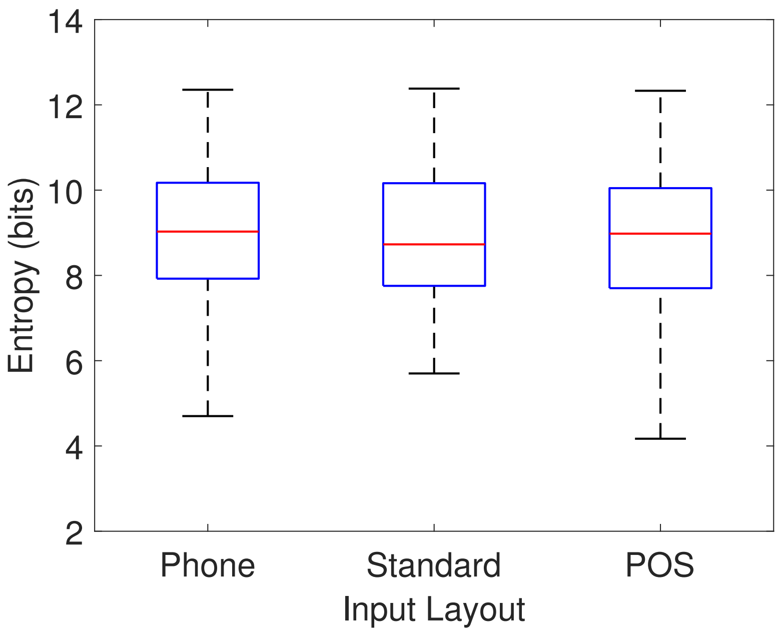


Figure 3.16: Impact of keypad size.

larger keypad size makes CSI waveforms for pressing different digits and switching between neighboring digits more distinguishable, thus yielding fewer spatiotemporal classification errors and richer spatiotemporal features. In terms of security inequality, the corresponding Gini coefficients are 0.50, 0.50, and 0.48 in the order of keypad size, implying that keypads with different sizes consistently have severe security inequality.

3.4.4.4 Impact of Keyboard Type

With different keyboard types, the amplitudes of hand movement vary, which may affect the accuracy of spatiotemporal classification. We choose three popular types of keyboards with the same keypad layout (3×4 number pad with size 75×94 mm): mechanical (Gigabyte Force K83 [36]), rubber-dome membrane (DELL KB216T), and touch-screen (on Lenovo Tab 4 10 Plus Tablet [54]). We

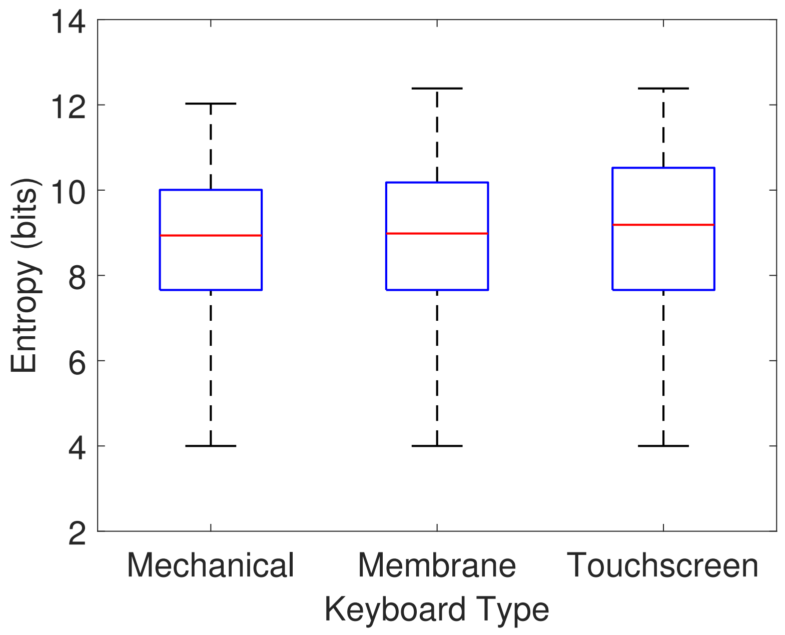


Figure 3.17: Impact of keypad type.

denote them with “Mechanical”, “Membrane”, and “Touchscreen”. Figure 3.17 shows the obtained PIN entropies. We observe similar entropy ranges for different keyboard types. The mechanical keyboard has the lowest mean entropy (8.8 bits) and the touch screen exhibits the highest one (9.0 bits). This is because the mechanical keyboard has the longest key travel distance and the membrane comes second. A longer key travel distance makes CSI waveforms associated with different digits more distinguishable, leading to more accurate spatiotemporal classification. Accordingly, the Gini coefficients for “Mechanical”, “Membrane”, and “Touchscreen” are 0.59, 0.59, and 0.60, respectively, confirming the security inequality for all different keyboards.

Table 3.3: Impact of different interaction scenarios.

Typing Hand	Scenario	G	PIN Entropy (bits)		
			Minimum	Maximum	Mean
Left	One-hand	0.50	3.8	12.3	9.4
	Same-hand	0.66	3.8	14.8	9.6
	Two-hand	0.79	4.0	17.2	9.8
Right	One-hand	0.58	3.8	14.8	9.5
	Same-hand	0.57	3.8	14.8	9.7
	Two-hand	0.77	3.8	17.2	10.1

3.4.4.5 Impact of Typing Scenarios in Mobile Devices

Users may interact with mobile devices in different ways. According to a questionnaire of 1,022 subjects [10], the following two interaction scenarios are the most popular: (1) same-hand: holding the device and typing with the thumb of the same hand; (2) two-hand: holding the devices with one hand and typing with a finger of the other hand. In another common case, referred to as a one-hand scenario, a user operates a mobile device placed face-up on a flat surface (e.g., a table). We focus on these three scenarios when the user inputs PINs on an iPhone 11 Pro Max. Additionally, a user may be left- or right-handed. Table 3.3 shows the PIN entropies and Gini coefficients for different scenarios. We have the following observations. First, handedness does not have an obvious impact on the attack performance, as corresponding PIN entropies and Gini coefficients are quite similar for left-hand and right-hand typing. Second, the PIN entropies for the three scenarios slightly vary. Overall, the one-hand scenario has the smallest mean PIN entropy, the same-hand scenario takes second place, and the two-hand scenario has the largest. This can be explained by the fact that the hand holding the phone may introduce extra movement during typing and such interference is most impactful in the two-hand scenario. Finally, all Gini coefficients are above 0.5, again demonstrating that our attack causes severe security inequality among

Table 3.4: Impact of different keypad's slope angles.

Slope Angle	G	PIN Entropy (bits)		
		Minimum	Maximum	Average
0°	0.50	2.0	11.5	8.8
30°	0.76	4.5	17.2	9.5
60°	0.84	3.8	17.2	9.2
90°	0.69	2.0	14.8	8.9

different PINs.

3.4.4.6 Impact of Keypad's Slope Angle

To reduce wrist extension and facilitate viewing of the keypad, some keypads may have a built-in or tilt-adjustable slope angle θ, which is defined as the angle between the keypad plane and the horizontal plane [6]. For example, keypads for most POS, ATMs, and petrol pumps are often installed with a slope angle between 0 and 90 degrees, determined by the keypad height above ground level [25]. We enable the user to type on a YD541 POS keypad and vary θ from 0° to 90°, with increments of 30°, where 0° denotes that the keypad is placed flat and 90° represents that the keyboard is parallel to the vertical wall. Table 3.4 presents the PIN entropies and Gini coefficients for different slope angles. We can see our attack decreases the PIN entropy consistently for different θ (note that the PIN entropy is 20.0 bits without our attack). Also, when θ equals 0° or 90°, the corresponding mean PIN entropy is slightly smaller than that for the case when θ is 30° or 60°. This appears as a tilted surface is more likely to introduce more failure of spatiotemporal classification. Lastly, all Gini coefficients are larger than 0.5, convincingly implying that different PINs may have different strengths against our attack irrespective of θ.

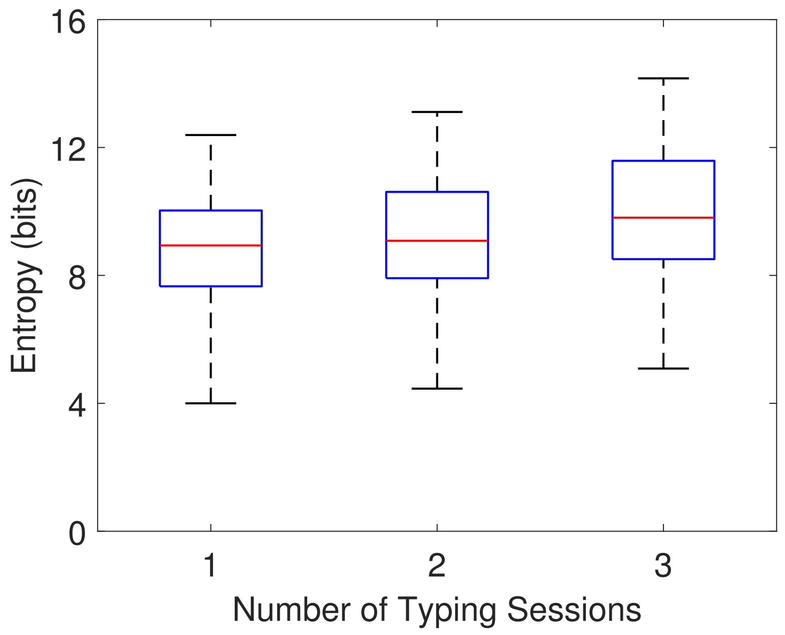


Figure 3.18: Impact of typing sessions.

3.4.4.7 Impact of Multiple Typing Sessions

As discussed in Section 3.3.2, users may perform intermittent typing and finish PIN input with several typing sessions. We let the user type each PIN on a keyboard number pad (Gigabyte Force K83) with varying typing sessions (1 to 3). Figure 3.18 presents the corresponding PIN entropies. We observe that *WINK* consistently reduces PIN entropy for all cases. Overall, such reduction performance is slightly decreased with more typing sessions. Specifically, the average PIN entropies for 1 to 3 typing sessions are 8.9, 9.2, and 9.9 bits, respectively. This is due to the fact that adding one typing session indicates that one inter-keystroke flight interval would not be used to shrink the candidates for the typed PIN. Besides, the Gini coefficients for 1 to 3 typing sessions are 0.59, 0.63, and 0.66, respectively, indicating the existence of security inequity among the PINs.

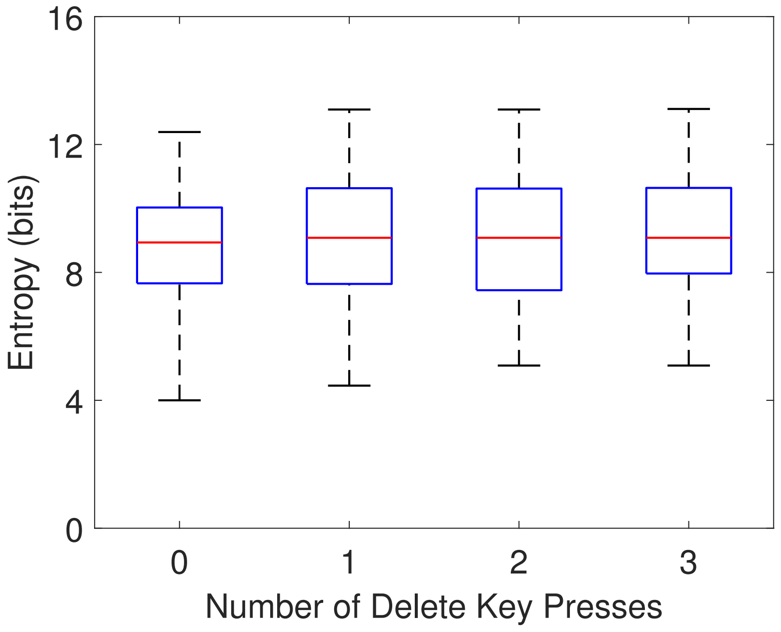


Figure 3.19: Impact of Delete key presses.

3.4.4.8 Impact of Typing Non-numeric Keys

As aforementioned, users may occasionally need to type some non-numeric keystrokes during inputting a number (e.g., for indicating the end of input or correcting input). No matter whether the OK/Enter key is used or not in the end to finish the typing session, *WINK* extracts the same spatiotemporal feature with the same CSI time series. Thus, its performance would not be affected. In this section, we focus on evaluating the impact of erasing mistyped digits with the Delete key. For each PIN, we let the user type the Delete key once to correct one digit, twice to correct two digits, and three times to correct three digits. The number pad on a Gigabyte Force K83 keyboard is utilized. For comparison, we also let the user type each PIN without using the Delete key. Figure 3.19 presents the PIN entropies for different numbers of Delete key presses. We see that *WINK* decreases the PIN entries at a similar level regardless of the number

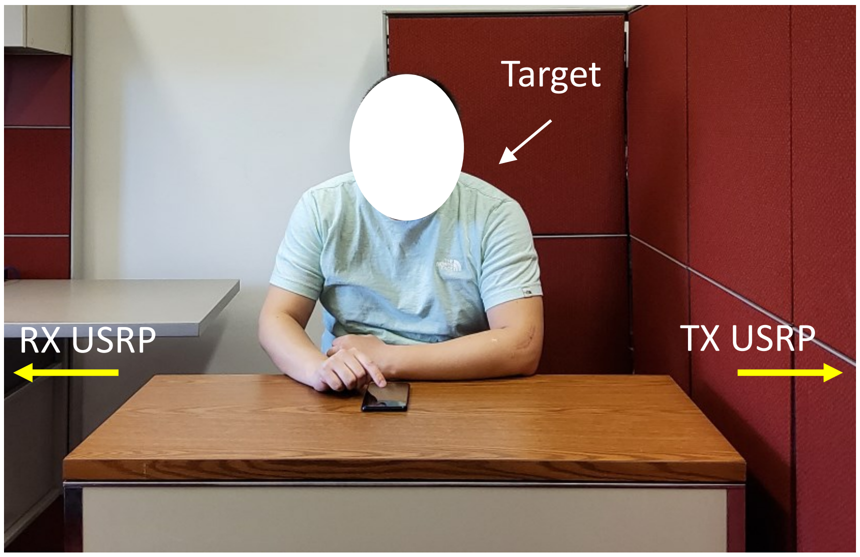


Figure 3.20: Experimental scenarios for office environment.

of Delete key presses. The mean PIN entropies for 1 to 3 Delete key presses are 9.10, 9.16, and 9.25 bits. Compared to the case when no Delete key is used (with the mean PIN entropy of 8.9 bits), Delete key usage brings slightly higher average PIN entropies. This is because the Delete key essentially disrupts the PIN entry into two typing sessions (before and after typing the Delete key). Furthermore, the Gini coefficients for 0 to 3 Delete key presses are 0.59, 0.64, 0.64, and 0.63, respectively, showing that typing Delete keys does not mitigate security inequity for different PINs.

3.5 Real-world User Study

We recruited 20 volunteers (U1-U20; aged 21-36 years old; 8 self-identified as females and 12 as males) to examine the practicality of *WINK*.[1] We consider two general typing scenarios, as shown in Figure 3.20: a private office room, and

[1]The study has been reviewed and approved by our institution's IRB.

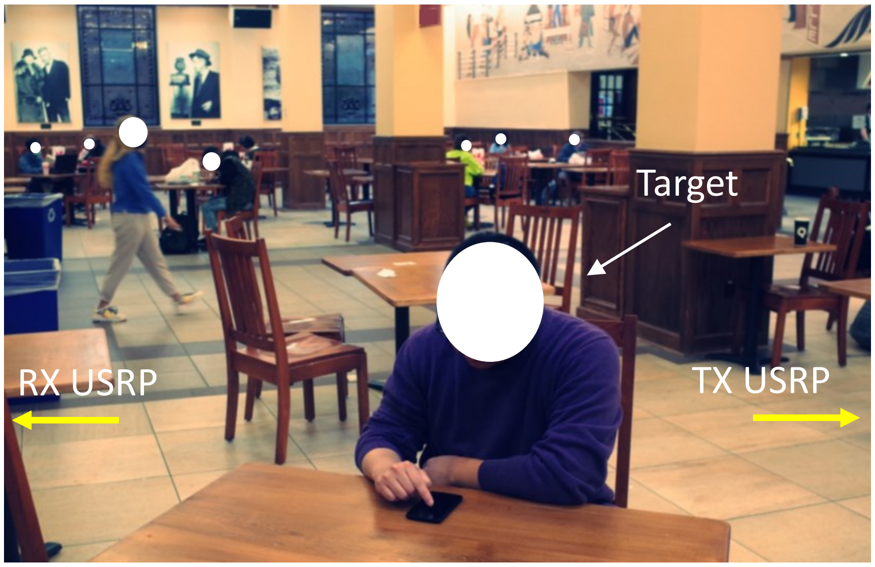


Figure 3.21: Experimental scenarios for public cafeteria environment.

Figure 3.21: a public campus cafeteria. The office room offers a quiet environment where there is no person walking around the user, while the cafeteria is noisy as people walk around or move chairs from time to time. In both scenarios, the target user sits at a table and types; the transmitter (Tx) and the receiver (Rx) are placed at opposite positions relative to the table. Each of Tx and Rx is a USRP X300 connected with a directional antenna – LP0965. Both Tx and Rx are put behind a 6 cm-thick wooden partition panel and are thus within non-line-of-sight (NLOS) of the target user. Their distances with the typing device (iPhone 11 Pro Max) are both 1.5 m.

Each participant was instructed to do the following tasks:

- *Unlocking with a 6-digit PIN*: iOS PIN blocklist is enforced to guarantee that no weak key is used.
- *Typing an SSN*: a valid SSN is formed by selecting a state, a corresponding 3-digit area code, and the rest 6 digits.

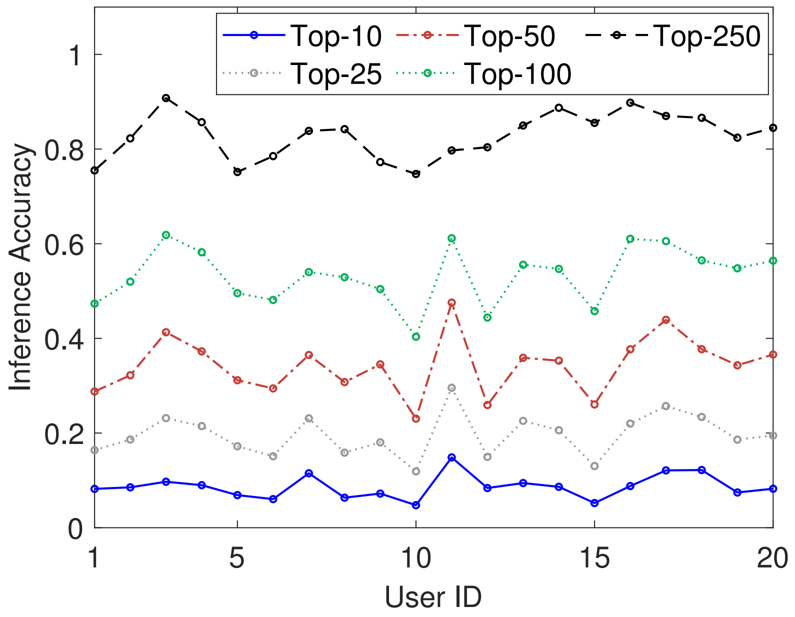


Figure 3.22: Average top-k accuracy for PIN inference in an office environment

For ethical considerations, we reminded users not to select their own in-use PINs for various applications or SSNs. Only after the user confirmed this, we started to launch our attack. Meanwhile, we allow the participants enough time to memorize/practice their selected numbers before testing. For every task, each participant performed 100 attempts with different numbers. We present the inference result to the participant, who determines whether the typed number is in the inferred number list. When the typed number is in the list, we then calculate the *top-k accuracy* α, defined as the probability that the top k guesses from the N candidates returned by our attack contain the typed number. If $k>N$, we have α=1; otherwise, $\alpha=\frac{{}^{1}C_{1}\cdot{}^{(N-1)}C_{(k-1)}}{{}^{N}C_{k}}=\frac{k}{N}$, where ${}^{N}C_{k}$ is the number of combinations by choosing k from N numbers.

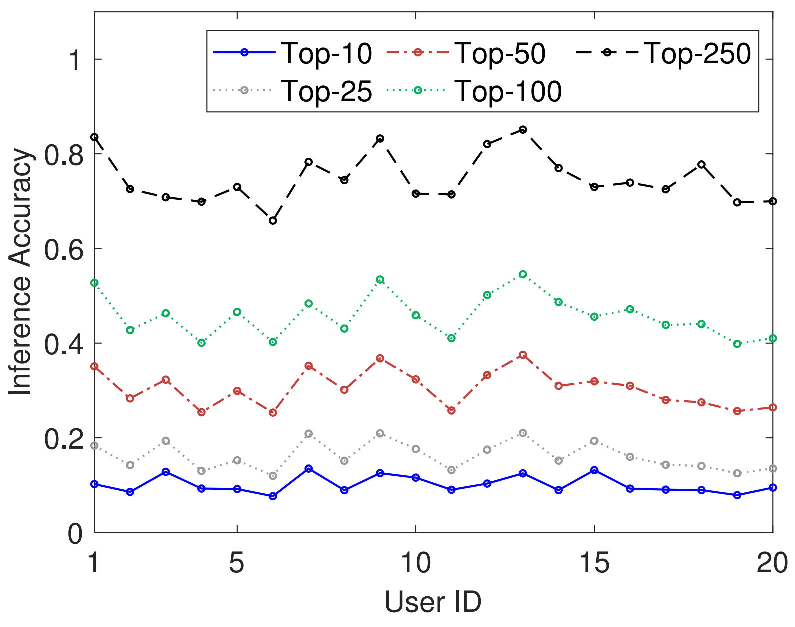


Figure 3.23: Average top-k accuracy for PIN inference in a cafeteria environment

3.5.1 PIN Inference Results

Figure 3.22 and 3.23 present the PIN inference performance in the two different environments. We observe that our attack consistently decreases the PIN strength for all users in both test environments. The top-25 accuracy in the office ranges from 13.0% to 29.6% while that value varies from 12.7% to 22.0% in the cafeteria. The slight accuracy decrease comes from the higher interference in the cafeteria. Also, the mean top-100 accuracy for all users equals above 50% (office: 52.3%; cafeteria: 50.6%), implying that more than half of the selected 6-digit PINs can be successfully inferred with up to 100 guesses. Besides, our attack achieves at least 74.7% and 66.0% top-250 accuracy for all users in the office and cafeteria environments, respectively.

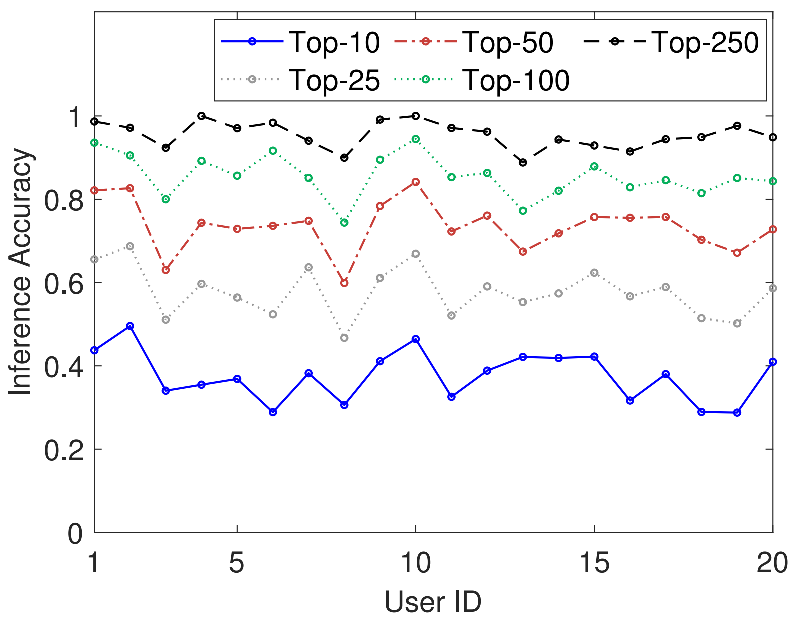


Figure 3.24: Average top-k accuracy for inferring SSNs in an office environment.

3.5.2 SSN Inference Results

Figure 3.24 and 3.25 present the SSN inference performance in the two different environments. We observe regardless of k, the corresponding top-k accuracy for 9-digit SSNs is always higher than that for 6-digit PINs in each scenario. This is because the knowledge of the 3-digit area code range provides extra information for shrinking the candidates. In the office, the top-25 accuracy is in the range of 46.7%-68.7%, implying that a substantial portion of typed SSNs is quite vulnerable to our attack; the average top-100 and top-250 accuracy across all users achieve 85.6% and 95.5%. Also, for some users (e.g., U4 and U10), the top-250 accuracy can reach 100%. In the cafeteria, the users obtain a top-25 accuracy in the range of 41.9% to 67.8%. Meanwhile, the average top-100 and top-250 accuracy across all users are 83.8% and 95.6%, respectively. Particularly, some users (e.g., U15 and U19) can also obtain a top-250 accuracy of 100%.

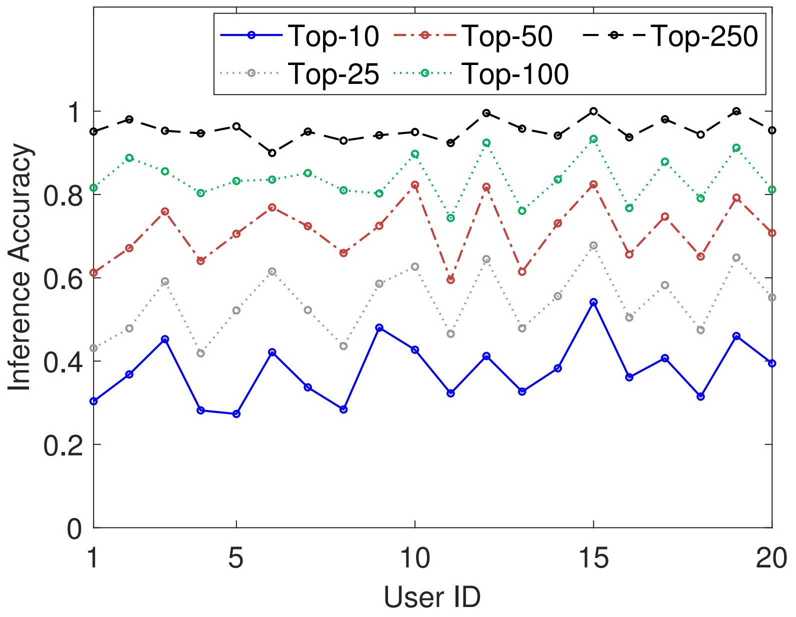


Figure 3.25: Average top-k accuracy for inferring SSNs in a cafeteria environment.

3.5.3 Security Inequity

Figure 3.26 shows the corresponding Gini coefficients. The result depicts all Gini coefficients above 0.36, confirming conclusively that our attack leads to severe security inequality among different PINs or SSNs. Also, in the same environment, the Gini coefficients for the SSNs are slightly larger than that for the PINs for most users. This is because the 3-digit area codes of some selected SSNs may disclose enough information for breaking those SSNs easier than the others.

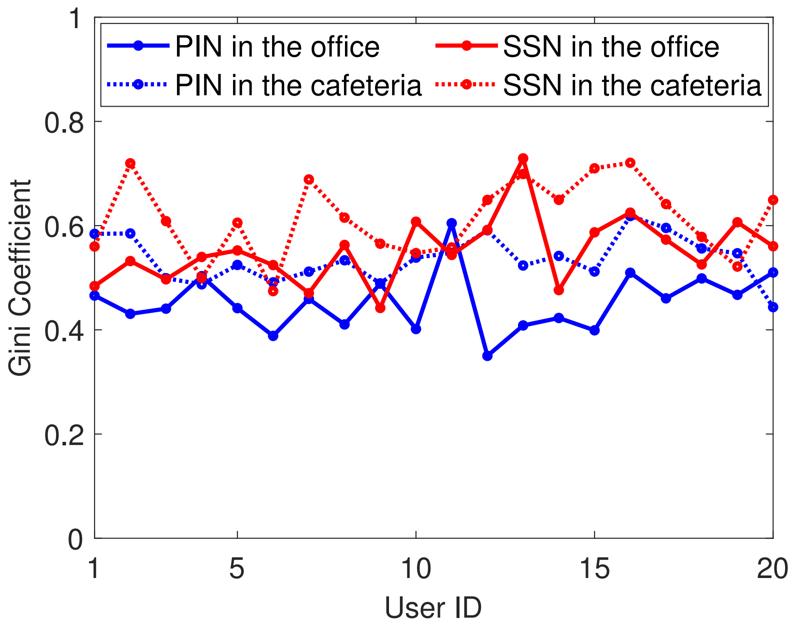


Figure 3.26: Gini coefficients for chosen PINs and SSNs.

Chapter 4

Countermeasures

In this chapter, we discuss possible countermeasures for resisting *WINK*. We also expect that additional countermeasures can be implemented based on our survey on wireless-based Human Profile Information (HPI) inference attacks [40].

4.1 Randomized Input Layouts

WINK exploits the spatiotemporal feature in CSI measurements to infer keystrokes. Intuitively, to defend against such attacks, we can stop the attacker from obtaining the correct spatiotemporal feature. Accordingly, one straightforward defense is to randomize the number pad every digit typing, such that the disclosed spatiotemporal feature would be obfuscated (e.g., the repetition of single-keystroke waveforms does not necessarily indicate the same keystrokes). This randomization can be implemented for touch-screen number pads, while it is not feasible for physical number pads. Meanwhile, it may be inconvenience for many users who get used to typing with muscle memory and without any visual assistance.

4.2 Typing Extra Digits

A practical way to confuse the attacker is *to type extra digits that are unknown to attackers*, as the attacker has to distinguish which part of the input corresponds to the target number. However, typing extra digits may disclose more information about the target number and thus cause a decrease in security, as repeatedly demonstrated in our experiments, where a longer digit sequence is not necessarily more secure than a relatively shorter one against our attack. To avoid being counter-productive, the extra digits and their positions in the whole digit sequence should be carefully designed.

Let N_0 denote the number of candidates for the target number when no defense is applied, and N represents the number of candidates for the typed whole digit sequence. If typing the chosen extra digits can guarantee that $N \geq N_0$, such extra digits can be then utilized to increase the security of the typed number. Extra digits can be put at the beginning or end of the whole typing session to make a user easily remember their positions. Note that the positions of these extra digits are pre-shared between the user and the target computer system so that the system can isolate the input of the target number from the extra digits. Suppose there are L extra digits. There are $(L+1)$ possibilities for the position of the target number within each candidate of the typed whole digit sequence. Thus, the attacker would obtain up to $N \cdot (L+1)$ possibilities for the target number. Though this method may increase the efforts of the attacker, it introduces an extra typing burden for typists and slows down the number input efficiency.

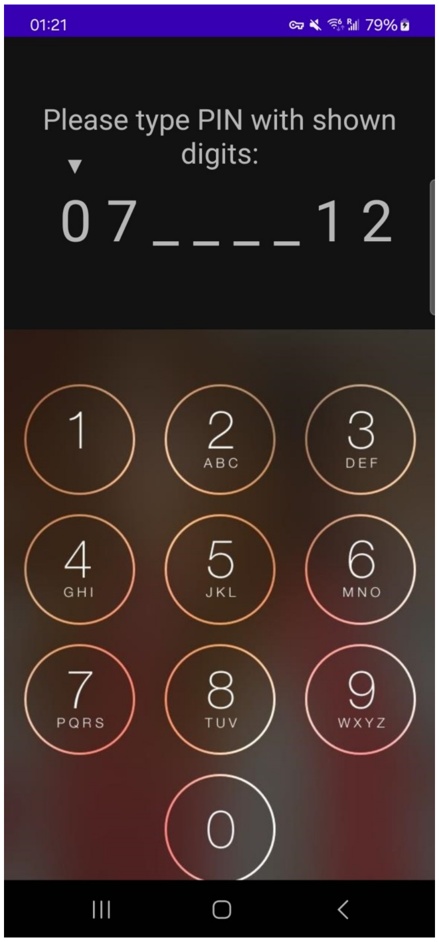


Figure 4.1: User interface of the smartphone application for the countermeasure experiments.

4.2.1 Experimental Results

To verify the effectiveness of the proposed countermeasure, we first implement a prototype as a smartphone application. As presented in Figure 4.1, two randomly selected digits are shown to the user before and after the actual PIN input, respectively. We note that the number of additional digits can be adjusted. In the quiet office environment, a user types 50 4-digit PINs separately, while an attacker launches *WINK*. For each PIN, the user makes 50 attempts, where the random digits are shown to the user for each attempt. For comparison, each

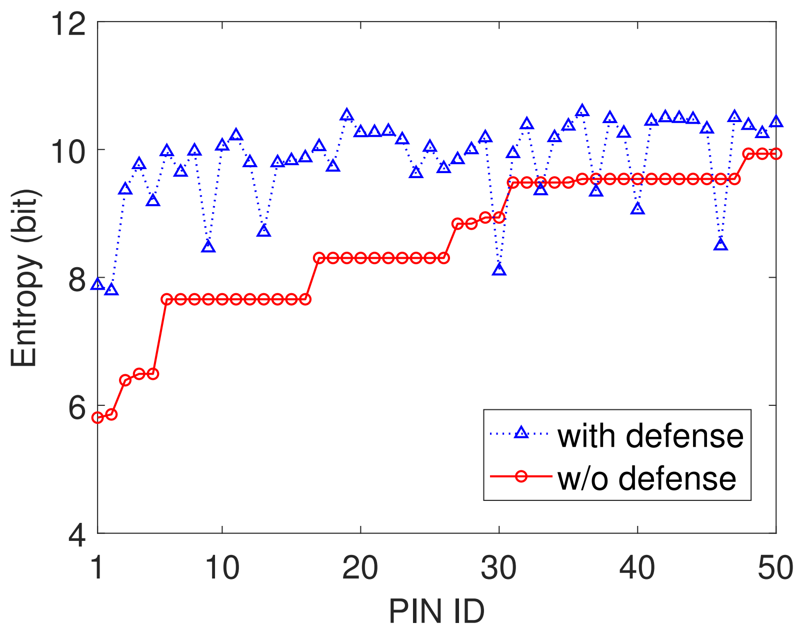


Figure 4.2: Impact of countermeasure on average PIN entropy.

PIN's entropy without the countermeasure, and average PIN entropy over 50 trials are measured.

Figure 4.2 demonstrates PIN entropy with and without countermeasure. For most PINs, we can see that the entropy values are improved from 0.3 to 3.27-bit. However, we can see that the entropy values for some PINs have decreased even further, even with the countermeasure. This is because the additional digits are randomly selected, where they do not guarantee improved PIN security. In other words, the extra digits typed by the user disclose richer spatiotemporal information. This observation also confirms the experiment results in Section 3.4.2, where longer PINs do not always provide better security.

We further verify this with a specific case. When the victim types a 4-digit PIN '0246', the attacker gets 58 candidates after launching *WINK*. With the proposed countermeasure, when the victim types **41**0246**22**, although the victim

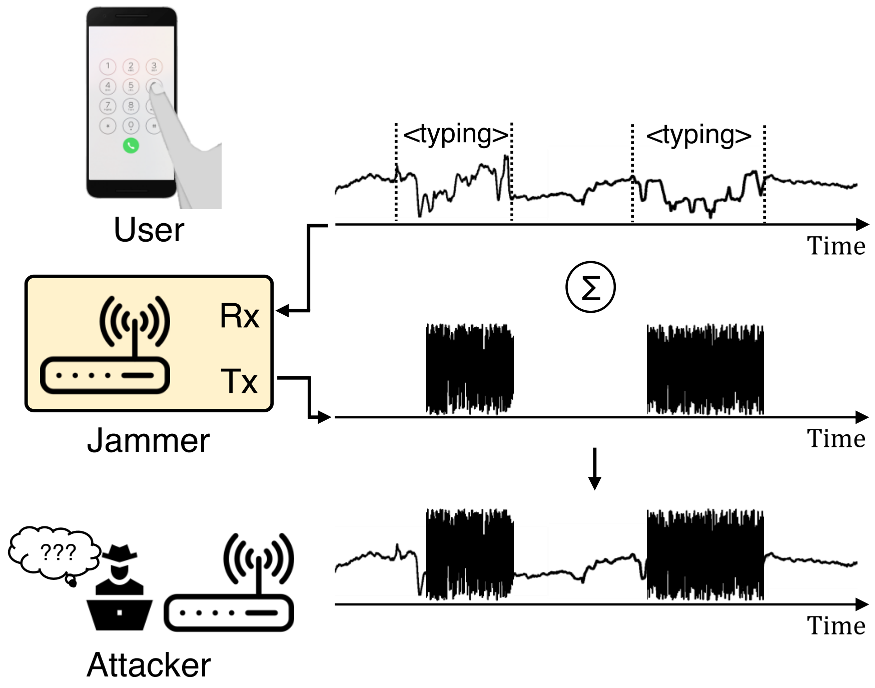


Figure 4.3: A reactive jamming device preventing an attacker from obtaining valid CSI.

typed 4 more digits, the attacker gets only 4 candidates, where the countermeasure compromises the PIN security. To prevent such cases, the additional digits need to be carefully selected. This can be achieved by simulating *WINK*, i.e., profiling the expected PIN entropy with possible additional digits, and utilizing the digits that guarantee the improved PIN security. This profiling can be done right after the user configures a new PIN. By excluding the additional digit combinations that compromise the PIN security, we expect the average PIN entropy will be further improved, compared with the current result shown in Figure 4.2.

4.3 Selective Jamming

Alternatively, we can also directly stop the attacker from obtaining clear CSI data streams leveraging selective jamming [27], so that no valid spatiotemporal feature can be extracted for inferring the typed number. The key idea of jamming is that signals of the jammer and the sender collide at the receiver, and the signal reception process is disrupted, due to the shared nature of the wireless medium.

We can set up a jammer that constantly transmits random signals over the wireless channel to prevent the attacker from sensing the variation of the signal transmitted by the transmitter. As a constant jammer that never stops is quite inefficient, we can employ a reactive jammer instead, which initializes the jamming once the typing is detected (e.g., [58,100]) and returns to the inactive mode when the typing ends.

Specifically, a selective jammer in [100] keeps an eavesdropper from observing the keystroke waveform when the victim performs text input. As the same technique can be applied to numerical keystrokes, we introduce it in this section. 4.3 depicts how the jammer works.

4.3.1 Determination of Starting and End Point of Jamming

A reactive jammer (i.e., defender) needs to take a reactive time to detect the typing and initialize the jamming. To detect the typing (i.e., the event of at least one keystroke), the defender needs to collect the waveform of the first keystroke in the typing session. Thus, the ending point of the first keystroke waveform would trigger the jamming. Each keystroke normally corresponds to a sharp fall and rise pattern in the CSI waveform, which in turn facilitates the detection of each keystroke duration.

To obfuscate received signals at the attacker, the jammer transmits high-power noise signals, which can become dominant at the attacker's side. The jammer then needs to return to the inactive mode once it identifies the end of the typing session. Due to the existence of the reaction time, the attacker is still able to obtain the first keystroke waveform for each typing session when the jammer is launched. However, with only one keystroke waveform, the attacker can only guess the first typed character and is unable to infer the whole typing content without knowing the inner structure of the typing content. Therefore, the proposed attack fails. This approach presents hardware demands, however, the jammer does not need to be a sophisticated high-end device, and it can be any low-cost wireless device (e.g., BladeRF [74] or nRF24L01+ [21]) that can perform basic wireless communication function (e.g., transmitting jamming signals).

4.3.2 Experimental Results

To evaluate the effectiveness of the active jamming technique, [100] utilizes a third USRP X300 as the jammer, which starts to transmit noise signals when it detects the type event, and stops when it detects that the typing session ends.

Figure 4.4 presents an example of the pre-processed CSI waveforms with and without the jammer, where the user types a word "apple". We can see that at the time of 1.8 seconds, the jammer initiates, and the jamming signals successfully obscure the keystroke associated patterns in the CSI waveform, demonstrating the effectiveness of the reactive jamming technique.

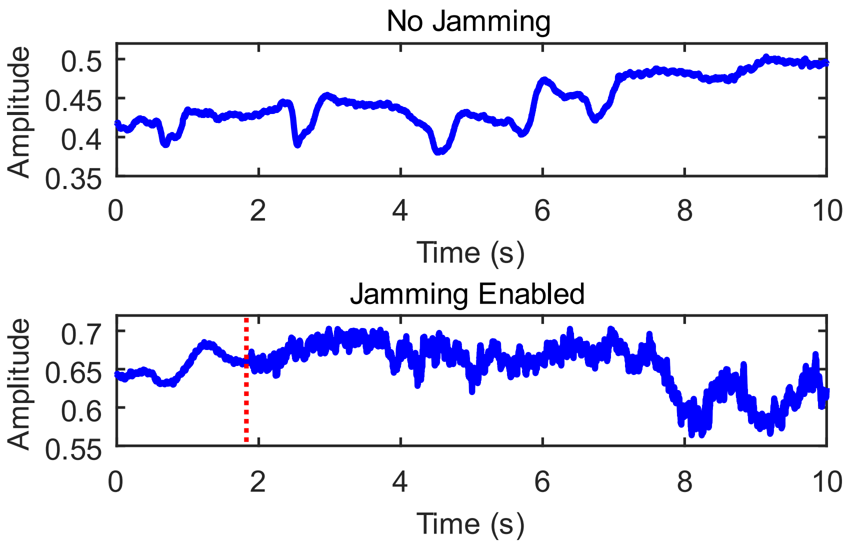


Figure 4.4: An example of observed CSI waveforms after pre-processing at the eavesdropper with and without jamming.

4.4 Injecting Fake Channel Estimate Information

As demonstrated above, deploying a jammer makes an attacker hard to obtain valid CSI observations. However, as the attacker can identify the presence of a jammer, it is also possible that she escalates the attack to another level (e.g., utilize a different side-channel) to infer the typed digit. In such a case, a smarter countermeasure would be deceiving the attacker, where we make her believe that the attacker is getting meaningful information from a side-channel by leaking fake information. There has been research effort to deceive an eavesdropper on wireless channels [33,38,39], and this idea has been implemented for wireless CSI side-channel attacks [41].

The deception strategy varies in two ways: (1) Hiding the impact of the keystroke while the victim is typing, or (2) Injecting fake keystroke impacts even if there are no keystroke events. In wireless channels, both strategies can be achievable by manipulating the public preamble $X(t)$ we discussed in Section 3.1.1.

Chapter 5

Future Work

This chapter discusses possible improvements of the proposed attack, and also directions of future work.

5.1 Improved Wireless-based Keystroke Inference Attack

5.1.1 Utilizing Additional CSI Features

As shown in Section 3.3, *WINK* relies on the observed single-keystroke waveforms and fight interval between the keystroke waveforms. We note that one possible way to further improve the performance of *WINK* is utilizing the CSI waveforms observed over flight intervals. During each flight interval, the corresponding CSI waveform is resulted by the user's hand movement. For example, on one hand, when the user types the same key twice, we can expect a static CSI waveform during the corresponding flight interval. On the other hand, when the user moves a hand from one key to a different key, the corresponding CSI waveform will not remain static. We can get an initial observation in Figure 3.8 in Section 3.4.1. We expect that extracting specific CSI patterns/features according to the victim's hand movement will further reduce the search space for inferring the secret numbers. To achieve this, additional feasibility experiments are required.

5.1.2 Typing with Two Hands or Multiple Fingers

Currently, *WINK* targets the most common scenario when the user types with the same finger of a hand. Occasionally, people may use both hands to type or change typing fingers while typing. In such cases, CSI waveforms associated with typing the same key may differ, and the correlation between flight interval and inter-key distance would be broken. As a result, our attack may no longer work for such scenarios. To overcome this limitation, investigating the relationship between human typing behaviors and keystroke inference is required.

5.1.3 Detecting Start and End of Number Entering

In *WINK*, the transmitter constantly emits signals while the receiver continuously estimates CSI with received signals. As keystroke-associated CSI waveforms often show distinguishable rising and falling trends, we use a sliding window method to identify the start and end of number input. Specifically, we search for noteworthy fluctuation (i.e., the difference between two neighboring local extrema) caused by a keystroke in the window, which slides along the CSI time series every extremum. This scheme is costly.

To detect the first keystroke event, we can first consider performing wireless traffic analysis [58]. Although the sensitive data in the traffic are secured by

HTTPS, the metadata in packet headers (e.g., IP address of the destination) are not encrypted. Therefore, by monitoring whether the victim is transmitting packets to a specific destination (e.g., online payment service), the attacker can identify the first keystroke. Also, motion-induced wireless traffic bursts may also help identify human activities (e.g., [42–44]), such as keystrokes. Lastly, external triggers such as a video feed can be utilized for searching the first keystroke event. Meanwhile, determining the end of typing is simple, as the user will stop typing once the number input is done. Once the CSI becomes static for a certain period, we identify the end of the last keystroke waveform as the end of the input.

5.1.4 Lowering Cost of Attack

As *WINK* utilizes CSI observations for inferring the keystrokes, we implemented it using the Software-Defined Radio (SDR) platform. However, utilizing low-cost wireless devices that are capable of collecting CSI can significantly reduce the budget for launching the attack. For example, Nexmon [81] allows cellphones and small single-board computers (SBCs) with specific WiFi chipsets to collect CSI. Another way to collect CSI is utilizing ESP32 micro-controller with the ESP32 CSI Toolkit [46]. ESP32 is a popularly utilized controller for various Internet-over-Thing (IoT) applications.

5.2 Different Applications of Side-channel

This dissertation primarily discusses *WINK*, i.e., a wireless side-channel attack. However, it is also an important fact that these side-channels can be utilized in a defensive manner. Some devices (i.e., Bluetooth headsets) we carry nowadays need to be paired with each other to establish a secure communication channel.

Usually, this pairing process is completed with human interactions (i.e., pushing a button, typing a PIN), which can be frustrating, especially with many devices.

To overcome this problem, context-based key-establishment schemes have been proposed. Due to the broadcast nature of the wireless medium, and the fact that the devices at the same location achieve highly similar wireless signal properties. However, [26,28,29] have shown that the wireless-based key-establishment techniques present vulnerability. There has been extensive research effort to develop improved wireless-based authentication schemes [30, 31]. However, these schemes may not work when the wireless signal is not available, or when the devices are co-located in the moving vehicle.

In [99] and [101], we propose context-based pairing schemes that utilize the Global Navigation Satellite System (GNSS) as security infrastructure. The key idea is the same as the one shared among the wireless signal-based authentication schemes [97]: the GPS-equipped devices located in the same vehicle can observe highly similar vehicle movement data, and we can utilize such data to establish a secret key.

When a user drives a vehicle, the corresponding GPS data exhibit randomness as the driver may alternatively step on the accelerator and brake pedals from time to time with varying force in order to adapt to the road traffic during driving. A vehicle provides a physically secure boundary as the devices co-located within the vehicle can observe common GPS data, as opposed to devices that do not experience the trip.

Meanwhile, a side-channel can be utilized as an input interface. In [45], we propose a practical PIN entry scheme that allows a user to enter a PIN through foot tapping on the ground. This scheme utilizes geophones to collect structural vibration signals caused by foot tapping. When a user generates the activation signals by performing a predetermined sequence of foot taps within the target area, the input sequence is initiated, and the user can use foot tapping to input a PIN. The system then demodulates the corresponding structural vibration signals into a PIN.

Furthermore, different types of side-channel can be exploited for implementing the attack. [107] proposes a voice command injection attack against Automated Speech Recognition (ASR) systems (e.g., Google Assistant). In this work, the injected commands are still played as audible sounds. However, we exploit the impact of fast speech to camouflage voice commands. This idea is based on the fact that both humans and ASR systems often misinterpret fast speech, and such misinterpretation can be exploited to launch hidden voice command attacks.

Chapter 6

Conclusion

This dissertation proposes a novel and practical numerical keystroke inference technique, with the following advantages over previous methods: (1) non-invasive, there is no need to pre-infect the victim's device with malware; (2) training-free, no training is required; (3) context-free, it does not rely on contextual information; and (4) non-line-of-sight (NLOS), the attacker's devices can be hidden from the target user. The novelty of *WINK* stems from identifying and constructing the spatiotemporal correlation between consecutive CSI measurements and typed digit sequences. Extensive evaluations show *WINK* significantly decreases the required attempts to infer numbers and such reduction for different numbers may vary substantially, causing severe security inequality among different PINs with the same length or SSNs. This dissertation also discusses possible countermeasures that can resist the proposed attack. Experimental results show that typing extra digits that do not disclose more spatiotemporal features can improve the security of the typed secret number.

Bibliography

[1] Social Security Administration. Social Security Number Allocations. https://web.archive.org/web/20190825104438/https://www.ssa.gov/employer/stateweb.htm, 2021.

[2] Md. Nafiul Alam Nipu, Souvik Talukder, Md. Saiful Islam, and Amitabha Chakrabarty. Human identification using WiFi signal. In *2018 Joint 7th International Conference on Informatics, Electronics & Vision (ICIEV) and 2018 2nd International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*, pages 300–304, 2018.

[3] Kamran Ali, Alex X. Liu, Wei Wang, and Muhammad Shahzad. Keystroke recognition using WiFi signals. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, MobiCom 2015, pages 90–102, New York, USA, 2015. ACM.

[4] Kamran Ali, Alex X. Liu, Wei Wang, and Muhammad Shahzad. Recognizing keystrokes using WiFi devices. *IEEE Journal on Selected Areas in Communications*, 35(5):1175–1190, 2017.

[5] Kiran Balagani, Matteo Cardaioli, Mauro Conti, Paolo Gasti, Martin Georgiev, Tristan Gurtler, Daniele Lain, Charissa Miller, Kendall Molas, Nikita Samarin, et al. PILOT: Password and PIN information leakage from obfuscated typing videos. *Journal of Computer Security*, 27(4):405–425, 2019.

[6] Ahmed Basager, Quintin Williams, Hereford Johnson, and Prasanna Mahajan. A user-centered ergonomic keyboard design to mitigate work-related musculoskeletal disorders. In *International Journal of Ergonomics*, volume 10, pages 27–42, 2020.

[7] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer, 2009.

[8] Darlynda Bogle. Unemployment Insurance Fraud and Social Security. https://blog.ssa.gov/unemployment-insurance-fraud-and-social-security/, 2021.

[9] Joseph Bonneau, Sören Preibusch, and Ross Anderson. A birthday present every eleven wallets? the security of customer-chosen banking PINs. In *International Conference on Financial Cryptography and Data Security*, pages 25–40. Springer, 2012.

[10] Christina Bröhl, Alexander Mertens, and Martina Ziefle. How do users interact with mobile devices? An analysis of handheld positions for different technology generations. In *International Conference on Human Aspects of IT for the Aged Population*, pages 3–16. Springer, 2017.

[11] United States Census Bureau. Income in the united states: 2022. https://www.census.gov/content/dam/Census/library/publications/2023/demo/p60-279.pdf, 2023.

[12] Liang Cai and Hao Chen. Touchlogger: Inferring keystrokes on touch screen from smartphone motion. In *Proceedings of the 6th USENIX Conference on Hot Topics in Security*, HotSec 2011, page 9, USA, 2011. USENIX Association.

[13] Matteo Cardaioli, Mauro Conti, Kiran Balagani, and Paolo Gasti. Your PIN sounds good! On the feasibility of pin inference through audio leakage. In *Proceedings of the 25th European Symposium on Research in Computer Security*, ESORICS 2020, pages 720–735. Springer-Verlag Italia, 2020.

[14] Bo Chen, Vivek Yenamandra, and Kannan Srinivasan. Tracking keystrokes using wireless signals. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys 2015, pages 31–44, New York, USA, 2015. Association for Computing Machinery.

[15] Yimin Chen, Tao Li, Rui Zhang, Yanchao Zhang, and Terri Hedgpeth. Eyetell: Video-assisted touchscreen keystroke inference from eye movements. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 144–160, 2018.

[16] Alberto Compagno, Mauro Conti, Daniele Lain, and Gene Tsudik. Don't Skype & Type! acoustic eavesdropping in Voice-Over-IP. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS 2017, pages 703–715, 2017.

[17] Patrick Cronin, Xing Gao, Chengmo Yang, and Haining Wang. Charger-Surfing: Exploiting a power line Side-Channel for smartphone information leakage. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 681–698. USENIX Association, August 2021.

[18] Nik Cubrilovic. RockYou Hack: From Bad To Worse. https://techcrunch.com/2009/12/14/rockyou-hack-security-myspace-facebook-passwords/, 2009.

[19] Dell. Dell Multimedia Keyboard-KB216. https://www.dell.com/en-us/shop/dell-multimedia-keyboard-kb216-us-international-qwerty-black/apd/580-admt/pc-accessories, 2024.

[20] Yadolah Dodge. *The concise encyclopedia of statistics*. Springer Science & Business Media, 2008.

[21] SparkFun Electronics. Sparkfun transceiver breakout - nrf24l01+ (rp-sma). https://www.sparkfun.com/products/705, 2022.

[22] Ettus Research. CBX 1200-6000 MHz Rx/Tx (40 MHz). https://www.ettus.com/all-products/cbx/, 2024.

[23] Ettus Research. LP0965 antenna. https://www.ettus.com/all-products/lp0965/, 2024.

[24] Ettus Research. VERT2450 antenna. https://www.ettus.com/all-products/vert2450/, 2024.

[25] European Payments Council. EPC343-08 v2.0 Privacy Shielding for PIN Entry. https://www.europeanpaymentscouncil.eu/document-library/guidance-documents/privacy-shielding-pin-entry, 2016.

[26] Song Fang, Yao Liu, and Peng Ning. Mimicry attacks against wireless link signature and new defense using time-synched link signature. *IEEE Transactions on Information Forensics and Security*, 11(7):1515–1527, 2016.

[27] Song Fang, Yao Liu, and Peng Ning. Wireless communications under broadband reactive jamming attacks. *IEEE Transactions on Dependable and Secure Computing*, 13(3):394–408, 2016.

[28] Song Fang, Yao Liu, Wenbo Shen, and Haojin Zhu. Where are you from? confusing location distinction using virtual multipath camouflage. In *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*, MobiCom 2014, pages 225–236, New York, USA, 2014. Association for Computing Machinery.

[29] Song Fang, Yao Liu, Wenbo Shen, Haojin Zhu, and Tao Wang. Virtual multipath attack and defense for location distinction in wireless networks. *IEEE Transactions on Mobile Computing*, 16(2):566–580, 2017.

[30] Song Fang, Ian Markwood, and Yao Liu. Manipulatable wireless key establishment. In *2017 IEEE Conference on Communications and Network Security (CNS)*, pages 1–9, 2017.

[31] Song Fang, Ian Markwood, and Yao Liu. Wireless-assisted key establishment leveraging channel manipulation. *IEEE Transactions on Mobile Computing*, 20(1):263–275, 2021.

[32] Song Fang, Ian Markwood, Yao Liu, Shangqing Zhao, Zhuo Lu, and Haojin Zhu. No Training Hurdles: Fast training-agnostic attacks to infer your typing. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS 2018, pages 1747–1760, Toronto, Canada, 2018. ACM.

[33] Song Fang, Tao Wang, Yao Liu, Shangqing Zhao, and Zhuo Lu. Entrapment for wireless eavesdroppers. In *Proceedings of the IEEE International Conference on Computer Communications*, INFOCOM 2019, pages 2530–2538. IEEE, 2019.

[34] Cyrus Farivar. Apple to require 6-digit passcodes on newer iPhones, iPads under iOS 9. https://arstechnica.com/gadgets/2015/06/apple-to-require-6-digit-passcodes-on-newer-iphones-ipads-under-ios-9/, 2015.

[35] Fidelity Investments Inc. Fidelity's automated service telephone. https://www.fidelity.com/customer-service/phone-numbers/fast/overview, 2024.

[36] Gigabyte. FORCE K83. https://www.gigabyte.com/Keyboard/FORCE-K83#kf, 2024.

[37] Andrea Goldsmith. *Wireless Communications.* Cambridge University Press, New York, USA, 2005.

[38] Qiuye He and Song Fang. Phantom-CSI attacks against wireless liveness detection. In *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses*, RAID 2023, pages 440–454, New York, USA, 2023. Association for Computing Machinery.

[39] Qiuye He, Song Fang, Tao Wang, Yao Liu, Shangqing Zhao, and Zhuo Lu. Proactive anti-eavesdropping with trap deployment in wireless networks. *IEEE Transactions on Dependable and Secure Computing*, 20(1):637–649, 2023.

[40] Qiuye He, Edwin Yang, and Song Fang. A survey on human profile information inference via wireless signals. *IEEE Communications Surveys & Tutorials*, pages 1–34, 2024.

[41] Qiuye He, Edwin Yang, Song Fang, and Shangqing Zhao. HoneyBreath: An ambush tactic against wireless breath inference. In *International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*, pages 203–226. Springer, 2022.

[42] Yan He, Qiuye He, Song Fang, and Yao Liu. MotionCompass: pinpointing wireless camera via motion-activated traffic. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys 2021, pages 215–227, New York, USA, 2021. Association for Computing Machinery.

[43] Yan He, Qiuye He, Song Fang, and Yao Liu. When free tier becomes free to enter: A non-intrusive way to identify security cameras with no cloud subscription. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, CCS 2023, pages 651–665, New York, USA, 2023. Association for Computing Machinery.

[44] Yan He, Qiuye He, Song Fang, and Yao Liu. Precise wireless camera localization leveraging traffic-aided spatial analysis. *IEEE Transactions on Mobile Computing*, 23(6):7256–7269, 2024.

[45] Yan He, Hanyan Zhang, Edwin Yang, and Song Fang. Virtual Step PIN Pad: Towards foot-input authentication using geophones. In *2020 IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pages 649–657, 2020.

[46] Steven M. Hernandez and Eyuphan Bulut. Lightweight and standalone iot based WiFi sensing for active repositioning and mobility. In *2020 IEEE 21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, pages 277–286, 2020.

[47] Thorsten Holz, Markus Engelberth, and Felix Freiling. Learning more about the underground economy: A case-study of keyloggers and dropzones. In *Computer Security–ESORICS 2009: 14th European Symposium on Research in Computer Security, Saint-Malo, France, September 21-23, 2009. Proceedings 14*, pages 1–18. Springer, 2009.

[48] Feng Hong, Xiang Wang, Yanni Yang, Yuan Zong, Yuliang Zhang, and Zhongwen Guo. WFID: Passive device-free human identification using WiFi signal. MOBIQUITOUS 2016, pages 47–56, New York, USA, 2016. Association for Computing Machinery.

[49] Jingyang Hu, Hongbo Wang, Tianyue Zheng, Jingzhi Hu, Zhe Chen, Hongbo Jiang, and Jun Luo. Password-Stealing without Hacking: Wi-Fi enabled practical keystroke eavesdropping. In *Proceedings of the 2023*

ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, pages 239–252, New York, USA, 2023. Association for Computing Machinery.

[50] Identity Theft Resource Center. New account fraud? a growing trend in identity theft. https://www.idtheftcenter.org/images/page-docs/NewAccountFraud.pdf, November 2016.

[51] American National Standards Institute. ANSI/HFES 100–2007 human factors engineering of computer workstations. *Hum Fact Ergon Soc*, 10:89, 2007.

[52] International Organization for Standardization. ISO 9564-1:2017 Financial services? Personal Identification Number (PIN) management and security. https://www.iso.org/standard/68669.html, 2017.

[53] Wenqiang Jin, Srinivasan Murali, Huadi Zhu, and Ming Li. Periscope: A keystroke inference attack using human coupled electromagnetic emanations. CCS 2021, pages 700–714, New York, NY, USA, 2021. Association for Computing Machinery.

[54] Lenovo. Tab 4 10 Plus. https://www.lenovo.com/us/en/tablets/android-tablets/tab-4-series/Lenovo-TB-X704/p/ZZITZTATB1X, 2024.

[55] Fan Li, Xiuxiu Wang, Huijie Chen, Kashif Sharif, and Yu Wang. ClickLeak: Keystroke leaks through multimodal sensors in cyber-physical social networks. *IEEE Access*, 5:27311–27321, 2017.

[56] Hong Li, Wei Yang, Jianxin Wang, Yang Xu, and Liusheng Huang. WiFinger: talk to your smart devices with finger-grained gesture. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp 2016, pages 250–261, New York, NY, USA, 2016. Association for Computing Machinery.

[57] Hong Li, Wei Yang, Jianxin Wang, Yang Xu, and Liusheng Huang. WiFinger: Talk to your smart devices with finger-grained gesture. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp 2016, pages 250–261, 2016.

[58] Mengyuan Li, Yan Meng, Junyi Liu, Haojin Zhu, Xiaohui Liang, Yao Liu, and Na 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*, CCS 2016, pages 1068–1079, 2016.

[59] Zhengxiong Li, Fenglong Ma, Aditya Singh Rathore, Zhuolin Yang, Baicheng Chen, Lu Su, and Wenyao Xu. WaveSpy: Remote and through-wall screen attack via mmWave sensing. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 217–232, 2020.

[60] Cheng-Jhe Lin and Changxu Wu. Factors affecting numerical typing performance of young adults in a hear-and-type task. *Ergonomics*, 54(12):1159–1174, 2011.

[61] Kang Ling, Yuntang Liu, Ke Sun, Wei Wang, Lei Xie, and Qing Gu. Spidermon: Towards using cell towers as illuminating sources for keystroke monitoring. In *Proceedings of the IEEE International Conference on Computer Communications*, INFOCOM 2020, 2020.

[62] Jian Liu, Yingying Chen, Yan Wang, Xu Chen, Jerry Cheng, and Jie Yang. Monitoring vital signs and postures during sleep using WiFi signals. *IEEE Internet of Things Journal*, 5(3):2071–2084, 2018.

[63] Jian Liu, Yan Wang, Gorkem Kar, Yingying Chen, Jie Yang, and Marco Gruteser. Snooping keystrokes with mm-level audio ranging on a single phone. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, MobiCom 2015, pages 142–154, New York, USA, 2015. Association for Computing Machinery.

[64] Xiangyu Liu, Zhe Zhou, Wenrui Diao, Zhou Li, and Kehuan Zhang. When good becomes evil: Keystroke inference with smartwatch. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS 2015, pages 1273–1285, 2015.

[65] Ximing Liu, Yingjiu Li, Robert H Deng, Bing Chang, and Shujun Li. When human cognitive modeling meets pins: User-independent inter-keystroke timing attacks. *Computers & Security*, 80:90–107, 2019.

[66] Chris Xiaoxuan Lu, Bowen Du, Hongkai Wen, Sen Wang, Andrew Markham, Ivan Martinovic, Yiran Shen, and Niki Trigoni. Snoopy: Sniffing your smartwatch passwords via deep sequence learning. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 1(4):152:1–152:29, January 2018.

[67] Anindya Maiti, Murtuza Jadliwala, Jibo He, and Igor Bilogrevic. Side-channel inference attacks on mobile keypads using smartwatches. *IEEE Transactions on Mobile Computing*, 17(9):2180–2194, 2018.

[68] Philipp Markert, Daniel V. Bailey, Maximilian Golla, Markus Durmuth, and Adam J. Aviv. This pin can be easily guessed: Analyzing the security

of smartphone unlock pins. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 286–303, 2020.

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

[70] Ajaya Neupane, Md Lutfor Rahman, and Nitesh Saxena. Peep: Passively eavesdropping private input via brainwave signals. In *Financial Cryptography and Data Security: 21st International Conference, FC 2017, Sliema, Malta, April 3-7, 2017, Revised Selected Papers 21*, pages 227–246. Springer, 2017.

[71] Graeme R Newman and Megan M McNally. Identity theft literature review. *Paper prepared for presentation and discussion at the National Institute of Justice Focus Group Meeting*, January 2005.

[72] Gabriele Oligeri, Savio Sciancalepore, Simone Raponi, and Roberto Di Pietro. BrokenStrokes: on the (in)security of wireless keyboards. In *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec 2020, pages 231–241, New York, USA, 2020. Association for Computing Machinery.

[73] Stefano Ortolani, Cristiano Giuffrida, and Bruno Crispo. Bait your hook: a novel detection technique for keyloggers. In *International workshop on recent advances in intrusion detection*, pages 198–217. Springer, 2010.

[74] Karel Pärlin, Muhammad Mahtab Alam, and Yannick Le Moullec. Jamming of UAV remote control systems using software defined radio. In *2018 International Conference on Military Communications and Information Systems (ICMCIS)*, pages 1–6, 2018.

[75] Anna Pereira, David L Lee, Harini Sadeeshkumar, Charles Laroche, Dan Odell, and David Rempel. The effect of keyboard key spacing on typing speed, error, usability, and biomechanics: Part 1. *Human factors*, 55(3):557–566, 2013.

[76] Carolyn Puckett. The story of the social security number. *Social Security Bulletin*, 69(2):55–74, 2009.

[77] Kun Qian, Chenshu Wu, Zheng Yang, Yunhao Liu, Fugui He, and Tianzhang Xing. Enabling contactless detection of moving humans with dynamic speeds using CSI. *ACM Trans. Embed. Comput. Syst.*, 17(2), jan 2018.

[78] Kun Qian, Chenshu Wu, Zheng Yang, Yunhao Liu, and Zimu Zhou. PADS: Passive detection of moving targets with dynamic speed using PHY layer information. In *2014 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, pages 1–8, 2014.

[79] Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS 2009, pages 199–212, 2009.

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

[81] Matthias Schulz, Daniel Wegemer, and Matthias Hollick. Nexmon: Build your own Wi-Fi testbeds with low-level MAC and PHY-access using firmware patches on off-the-shelf mobile devices. In *Proceedings of the 11th Workshop on Wireless Network Testbeds, Experimental Evaluation & CHaracterization*, WiNTECH 2017, pages 59–66, New York, USA, 2017. Association for Computing Machinery.

[82] Cong Shi, Jian Liu, Hongbo Liu, and Yingying Chen. Smart user authentication through actuation of daily activities leveraging WiFi-enabled IoT. In *Proceedings of the 18th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, Mobihoc 2017, New York, NY, USA, 2017. Association for Computing Machinery.

[83] Jonathon Shlens. A tutorial on principal component analysis. https://www.cs.cmu.edu/~elaw/papers/pca.pdf, 2014.

[84] Diksha Shukla, Rajesh Kumar, Abdul Serwadda, and Vir V. Phoha. Beware, your hands reveal your secrets! In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS 2014, pages 904–917. ACM, 2014.

[85] David Slater, Scott Novotney, Jessica Moore, Sean Morgan, and Scott Tenaglia. Robust keystroke transcription from the acoustic side-channel. In *Proceedings of the 35th Annual Computer Security Applications Conference*, ACSAC 2019, pages 776–787, 2019.

[86] Social Security Administration. Social Security Number Randomization. https://www.ssa.gov/employer/randomization.html, 2024.

[87] Dawn Xiaodong Song, David Wagner, and Xuqing Tian. Timing analysis of keystrokes and timing attacks on SSH. In *Proceedings of the 10th*

Conference on USENIX Security Symposium - Volume 10, SSYM 2001, Washington, D.C., 2001. USENIX Association.

[88] Jingchao Sun, Xiaocong Jin, Yimin Chen, Jinxue Zhang, Yanchao Zhang, and Rui Zhang. VISIBLE: Video-assisted keystroke inference from tablet backside motion. In *Network and Distributed Systems Security (NDSS) Symposium*. NDSS 2016, 2016.

[89] Sheng Tan and Jie Yang. WiFinger: leveraging commodity WiFi for fine-grained finger gesture recognition. In *Proceedings of the 17th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc 2016, pages 201–210, New York, NY, USA, 2016. Association for Computing Machinery.

[90] U.S. Attorney's Office. Dominican national sentenced for health care fraud, misuse of a social security number, ID theft. https://www.justice.gov/usao-ri/pr/dominican-national-sentenced-health-care-fraud-misuse-social-security-number-id-theft, 2020.

[91] Aditya Virmani and Muhammad Shahzad. Position and orientation agnostic gesture recognition using WiFi. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys 2017, pages 252–264, New York, USA, 2017. Association for Computing Machinery.

[92] Chen Wang, Xiaonan Guo, Yan Wang, Yingying Chen, and Bo Liu. Friend or Foe? Your wearable devices reveal your personal pin. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, ASIA CCS 2016, pages 189–200, 2016.

[93] Chen Wang, Jian Liu, Xiaonan Guo, Yan Wang, and Yingying Chen. WristSpy: Snooping passcodes in mobile payment using wrist-worn wearables. In *Proceedings of the IEEE International Conference on Computer Communications*, INFOCOM 2019, pages 2071–2079, 2019.

[94] Ding Wang, Qianchen Gu, Xinyi Huang, and Ping Wang. Understanding human-chosen pins: characteristics, distribution and security. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 372–385, 2017.

[95] Guanhua Wang, Yongpan Zou, Zimu Zhou, Kaishun Wu, and Lionel M. Ni. We can hear you with Wi-Fi! *IEEE Transactions on Mobile Computing*, 15(11):2907–2920, 2016.

[96] He Wang, Ted Tsung-Te Lai, and Romit Roy Choudhury. Mole: Motion leaks through smartwatch sensors. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, MobiCom 2015, pages 155–166, 2015.

[97] Tao Wang, Yao Liu, Tao Hou, Qingqi Pei, and Song Fang. Signal entanglement based pinpoint waveforming for location-restricted service access control. *IEEE Transactions on Dependable and Secure Computing*, 15(5):853–867, 2018.

[98] Yuxi Wang, Kaishun Wu, and Lionel M. Ni. WiFall: Device-free fall detection by wireless networks. *IEEE Transactions on Mobile Computing*, 16(2):581–594, 2017.

[99] Edwin Yang and Song Fang. GPSKey: GPS-based secret key establishment for intra-vehicle environment. In *Workshop on Automotive and Autonomous Vehicle Security (AutoSec)*, January 2022.

[100] Edwin Yang, Song Fang, Ian Markwood, Yao Liu, Shangqing Zhao, Zhuo Lu, and Haojin Zhu. Wireless training-free keystroke inference attack and defense. *IEEE/ACM Transactions on Networking*, 30(4):1733–1748, 2022.

[101] Edwin Yang, Song Fang, and Dakun Shen. DASK: Driving-assisted secret key establishment. In *2022 IEEE Conference on Communications and Network Security (CNS)*, pages 73–81, 2022.

[102] Edwin Yang, Qiuye He, and Song Fang. WINK: Wireless inference of numerical keystrokes via zero-training spatiotemporal analysis. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS 2022, pages 3033–3047, 2022.

[103] Qinggang Yue, Zhen Ling, Xinwen Fu, Benyuan Liu, Kui Ren, and Wei Zhao. Blind recognition of touched keys on mobile devices. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS 2014, pages 1403–1414, 2014.

[104] Yunze Zeng, Parth H. Pathak, and Prasant Mohapatra. WiWho: WiFi-based person identification in smart spaces. In *2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 1–12, 2016.

[105] Jin Zhang, Bo Wei, Wen Hu, and Salil S. Kanhere. WiFi-ID: Human identification using WiFi signal. In *2016 International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 75–82, 2016.

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

[107] Zhaohe (John) Zhang, Edwin Yang, and Song Fang. Commandergabble: A universal attack against asr systems leveraging fast speech. In *Proceedings of the 37th Annual Computer Security Applications Conference*, ACSAC 2021, pages 720–731, New York, USA, 2021. Association for Computing Machinery.

[108] Zijian Zhang, Nurilla Avazov, Jiamou Liu, Bakh Khoussainov, Xin Li, Keke Gai, and Liehuang Zhu. WiPOS: A POS terminal password inference system based on wireless signals. *IEEE Internet of Things Journal*, 7(8):7506–7516, 2020.

[109] Tong Zhu, Qiang Ma, Shanfeng Zhang, and Yunhao Liu. Context-free attacks using keyboard acoustic emanations. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS 2014, pages 453–464, New York, USA, 2014. Association for Computing Machinery.

[110] Li Zhuang, Feng Zhou, and J Doug Tygar. Keyboard acoustic emanations revisited. *ACM Transactions on Information and System Security (TISSEC)*, 13(1):1–26, 2009.