# WINK: Wireless Inference of Numerical Keystrokes via Zero-Training Spatiotemporal Analysis

Edwin Yang
The University of Oklahoma
Norman, OK, USA
edwiny@ou.edu

Qiuye He
The University of Oklahoma
Norman, OK, USA
qiuye.he@ou.edu

Song Fang
The University of Oklahoma
Norman, OK, USA
songf@ou.edu

## ABSTRACT

Sensitive numbers play an unparalleled role in identification and authentication. Recent research has revealed plenty of side-channel attacks to infer keystrokes, which require either a training phase or a dictionary to build the relationship between an observed signal disturbance and a keystroke. However, training-based methods are unpractical as the training data about the victim are hard to obtain, while dictionary-based methods cannot infer numbers, which are not combined according to linguistic rules like letters are. We observe 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 called *WINK* that combines the spatial and time domain information into a spatiotemporal feature of keystroke-disturbed wireless signals. With this spatiotemporal feature, *WINK* can infer typed numbers without the aid of any training. Experimental results on top of software-defined radio platforms show that *WINK* can vastly reduce 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.

## CCS CONCEPTS

• **Security and privacy → Mobile and wireless security**.

## KEYWORDS

Keystroke eavesdropping; SSN; PIN; spatiotemporal correlation

**ACM Reference Format:**
Edwin Yang, Qiuye He, and Song Fang. 2022. 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 '22), November 7–11, 2022, Los Angeles, CA, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3548606.3559339

## 1 INTRODUCTION

In the digital era, identifying numbers permeate every aspect of our daily life, 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 [28], access to Medicaid or unemployment insurance benefits [55], or fraudulent tax return filing and return claims [44]. Due to the large unemployment benefit expansion during the COVID-19 pandemic, scammers have filed huge numbers of fraudulent unemployment claims with stolen SSNs [5].

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 [23]. 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 [27], 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 [45]. Recent research focuses on developing non-invasive keystroke inference attacks [3, 13, 21, 33, 36, 42, 43, 51, 53, 59], which are more surreptitious as they only require passive monitoring of corresponding physical disturbances (e.g., brainwave signals [43], vibrations [42], acoustic emanations [13, 51], motion [59], inter-keystroke timing [3, 53] and wireless signals [1, 2, 21, 33, 36, 62]) 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 [13, 21, 59] 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., [1, 33, 36, 62]) 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.,



*CCS '22, November 7–11, 2022, Los Angeles, CA, USA*

ACM ISBN 978-1-4503-9450-5/22/11.
https://doi.org/10.1145/3548606.3560652

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 paper, 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 on 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 [3, 53] 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 continuously 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.

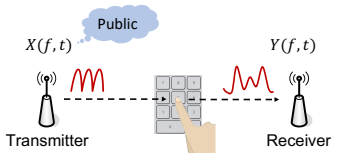



**Figure 1: General wireless-based keystroke inference setup.**

As wireless signals are ubiquitous, invisible, and able to propagate under non-line-of-sight (NLOS) conditions, we utilize wireless signal as the target disturbance to capture keystrokes. Particularly, a pre-trained model is not suitable 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 [1, 11, 33, 62] all utilize frequent training to cope with the environmental changes, instead of relying on a one-time pre-trained model. Also, all these work 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. [62] also performs multiuser 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.

Our contributions are summarized as follows.

- Unlike previous extensive research in inferring keystrokes using labor-intensive training or contextual information, this paper 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.

Our collected wireless channel data and code for processing the data are publicly available online[1].

## 2 PRELIMINARIES

**Wireless-based Keystroke Inference:** There are emerging research efforts in wireless-based keystroke inference [1, 2, 21, 33]. The common underlying principle is that the 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.

[1] https://projectwink.info

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 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 [25]. 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 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 $H(f,t) = \frac{Y(f,t)}{X(f,t)}$.

*General Workflow:* Existing keystroke inference methods using CSI [1, 2, 33] usually rely on three phases, including signal preprocessing, 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.

**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 [52], 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 [47] 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 ADVERSARY MODEL

In general, an attacker can control a wireless transmitter (TX) and receiver (RX) pair to launch the attack, as shown in Figure 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 [33]).

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 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 [35]. We discuss the limitations of the proposed technique in Section 7.1.

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

**Figure 2: Sketches of typical typing scenarios.**

## 4 ATTACK DESIGN

### 4.1 Attack Overview

To launch *WINK*, the attacker first estimates CSI with received signals (as described in Section 2) 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 neighboring 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.

### 4.2 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.

*4.2.1 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 [1, 21, 33]. 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 be classified into separate sets.

*4.2.2 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-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 to obtain candidates for the typed number or exhaust all sets.

*4.2.3 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 to the phase of spatiotemporal correlation, aiming to build the correlation between a sequence of single-keystroke waveforms and a digit sequence. Figure 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.

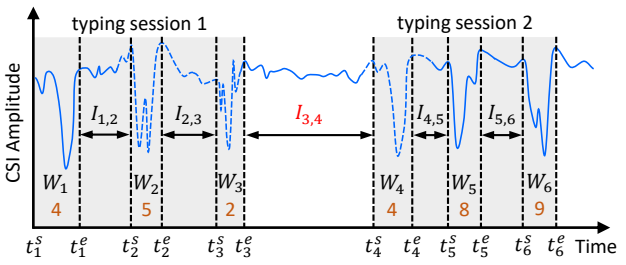



**Figure 3: Two typing sessions for inputting the number '452489' with a long delay after the first 3 digits. $W_i$ ($i \in \{1, \cdots, 6\}$) is the $i^{\text{th}}$ single-keystroke waveform, and $I_{i,i+1}$ is the flight interval between the $i^{\text{th}}$ and $(i+1)^{\text{th}}$ keystrokes.**

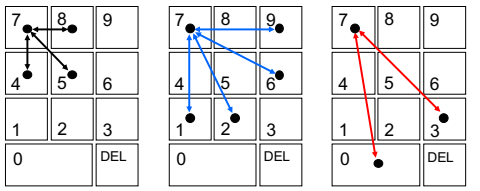
## 4.3 Spatiotemporal Correlation

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. After that, we consider recovering the digits typed within a period consisting of multiple CSI sessions.

*4.3.1 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 $\eta$, as the ratio between $K$ to $K_{max}$, i.e., $\eta = K/K_{max}$ ($\eta \in (0, 1]$ as $K \leq K_{max}$). When $\eta$ 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 $\eta$.

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.




(a) $IKD \in \{1, \sqrt{2}\}$ (b) $IKD \in \{2, \sqrt{5}\}$ (c) $IKD \in \{2\sqrt{2}, \frac{\sqrt{37}}{2}\}$

**Figure 4: Distances between digit '7' and other keys.**

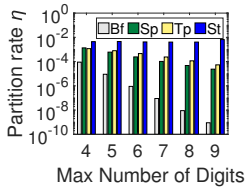



**Figure 5: Partition rates under different cases.**

**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

$$V : \mathbf{x} = [x_1, x_2, \cdots, x_n] \mapsto \mathbf{y} = [y_1, y_2, \cdots, y_n]. \quad (1)$$

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 waveforms as classified during spatial classification (Section 4.2.1); 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 vector to both CSI sessions and digit sequences, we can then extract their spatial and temporal features.

**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 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 [26, 46]. 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 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 $513/K_{max} = 4.62 \times 10^{-4}$.

**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 $\mathbf{s} = [s_1, s_2, \cdots, s_n]$ 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 5 presents partition rates when we search candidates using traditional brute-force guessing (Bf), spatial feature (Sp), temporal feature (Tp), and spatiotemporal feature (St), with $L_{max}$ varying from 4 to 9. We see an interesting phenomenon: with $L_{max}$ increasing, the partition rates for Bf, Sp, and Tp all decrease, indicating that the numeric inference difficulty increases, while the partition rate for St 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.

*4.3.2 Iterative Joint Decoding.* We consider the general case when observing $N$ CSI sessions, denoted by $[C_1, C_2, \cdots, C_N]$. 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.

Towards 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 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 $i^{\text{th}}$ ($i \in \{1, \cdots, 6\}$) observed single-keystroke waveform and $t_{j,j+1}$ is the calculated inter-keystroke flight interval between the $j^{\text{th}}$ 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.

## 4.4 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 [22]), the device would automatically initiate authentication. However, on certain occasions, we may need to use non-numeric keys, including the OK/Enter key, 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 which appear $\sigma$ 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 $\sigma$ 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.

## 4.5 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 obtained candidates have the correct typed number.

## 5 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 [17]. 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 6, 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* [16], 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

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

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 entropies are 0, leading to $G_l = \frac{N-1}{N} \approx 1$, where $N \gg 1$).

### 5.1 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 [1, 21, 33], including noise removal, dimension reduction, and keystroke waveform extraction. To sanitize the CSI data, a weighted moving average filter [32, 60] is applied. Next, we utilize Principal Component Analysis (PCA) [49] 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 6 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

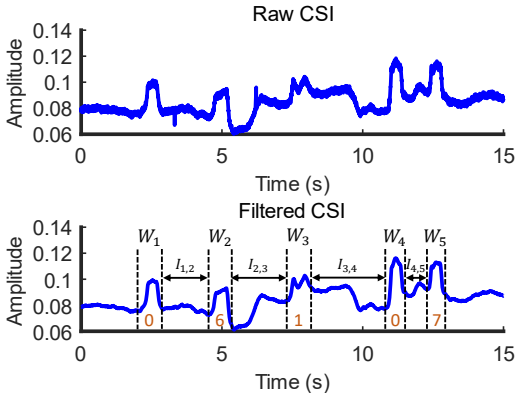



**Figure 6: CSI for the PIN "06107".**

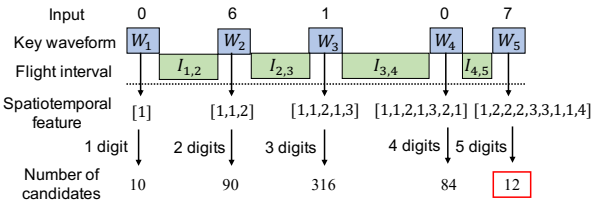



**Figure 7: Variation of the number of inferred candidates.**

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 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 7 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.

### 5.2 PIN Inference

To balance security and usability, most authentication systems allow PINs with 4 to 6 digits [29]. To approximate user choices of PINs, we extract leaked real-world PINs with 4 to 6 digits from the *RockYou* database [14], which is widely used in PIN security research (e.g., [6, 58]). 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 length, we sort the PINs in ascending order of the entropies and index them from 1 to 100 in increments of 1. Figure 8 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

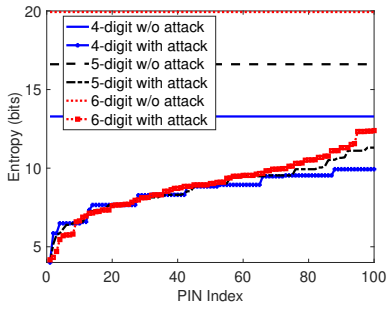



**Figure 8: Entropy distribution for PINs with different lengths.**

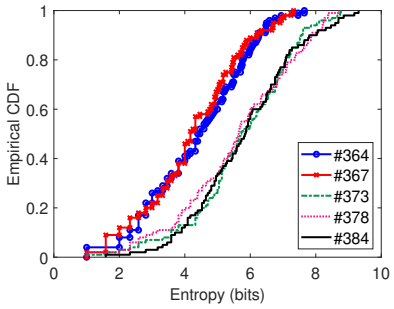



**Figure 9: CDFs of SSN entropy in a same state (Michigan).**

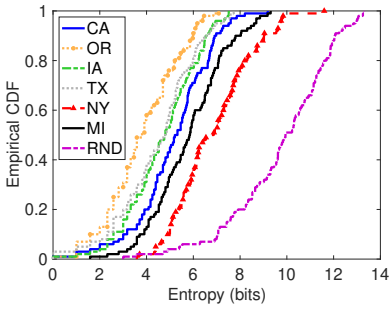



**Figure 10: CDFs of SSN entropy for different states.**

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 $G_6 = 0.57$. 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.

**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 [41] 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 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.

**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.

## 5.3 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.

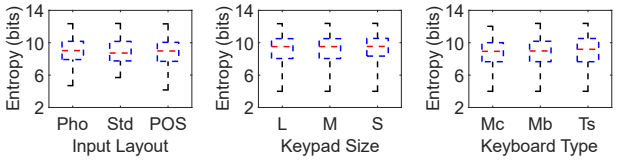
**Without SSN Randomization:** We select two states from each of the west, middle, and east of the United States, and obtain California (CA), Oregon (OR), Iowa (IA), Texas (TX), New York (NY), and Michigan (MI). Different states have different 3-digit area codes and a state may have one or multiple area codes. Such information is public [30]. 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.

*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 9 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, 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 $25 \times 10^6$ 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.

*SSNs across Different States*: We randomly generate 100 SSNs allocated for every state, and then let the user type each SSN separately

**Table 1: Degrees of security inequality for SSNs with a same area code and SSNs issued within a same state.**

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




(a) Impact of input layout (b) Impact of keypad size (c) Impact of keyboard type

**Figure 11: Impact of keypad features (layout/size/type).**

on a standard number pad. Figure 10 plots the CDFs of corresponding SSN entropies. We see that our attack consistently decreases SSN entropies, and different states have different entropy ranges. Specifically, 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 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 29.7 bits. Figure 10 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.

## 5.4 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.

*5.4.1 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 [19] and a directional

**Table 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 |

LP0965 antenna [18] 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 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.

*5.4.2 Impact of Keyboard Layout.* We test three popular layouts of numeric keypads with similar sizes, including (a) iOS passcode keypad (77.8×158 mm displayed on iPhone 11 Pro Max), (b) 3×4 number pad (75×94 mm) on the far right of a standard keyboard (DELL KB216T [15]), and (3) 3×4 POS keypad (53.5×80 mm for the model YD541). We refer to them as "Pho", "Std", and "POS", respectively. Figure 11 (a) 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 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 "Pho", "Std", 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.

*5.4.3 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

**Table 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 |

Mini (5.4-inch display), and iPhone SE (4.7-inch display), which we refer to as "L", "M", and "S". Figure 11 (b) 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 larger keypad size makes CSI waveforms for pressing different digits and switching between neighboring digits more distinguishable, thus yielding less 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.

*5.4.4 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 [24]), rubber-dome membrane (DELL KB216T), and touch-screen (on Lenovo Tab 4 10 Plus Tablet [31]). We denote them with "Mc", "Mb", and "Ts". Figure 11 (c) 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 "Mc", "Mb", and "Ts" are 0.59, 0.59, and 0.60, respectively, confirming the security inequality for all different keyboards.

*5.4.5 Impact of Typing Scenarios in Mobile Devices.* Users may interact with mobile devices in different ways. According to a questionnaire of 1,022 subjects [7], 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., 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 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

**Table 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 |

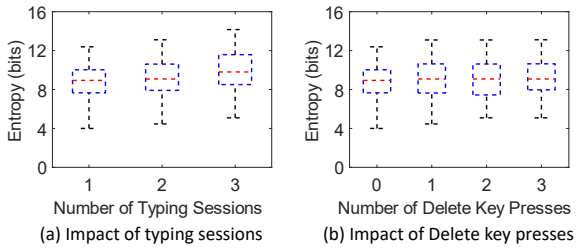
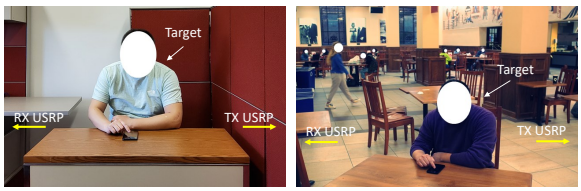



**Figure 12: Impact of typing sessions or Delete key presses.**

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 different PINs.

*5.4.6 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 $\theta$, which is defined as the angle between the keypad plane and the horizontal plane [4]. 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 [20]. We enable the user to type on a YD541 POS keypad and vary $\theta$ 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 4 presents the PIN entropies and Gini coefficients for different slope angles. We can see our attack decreases the PIN entropy consistently for different $\theta$ (note that the PIN entropy is 20.0 bits without our attack). Also, when $\theta$ equals 0° or 90°, the corresponding mean PIN entropy is slightly smaller than that for the case when $\theta$ 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 $\theta$.

*5.4.7 Impact of Multiple Typing Sessions.* As discussed in Section 4.2, 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 12 (a) 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




(a) Private office. (b) Public cafeteria.

**Figure 13: Experimental scenarios.**

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.

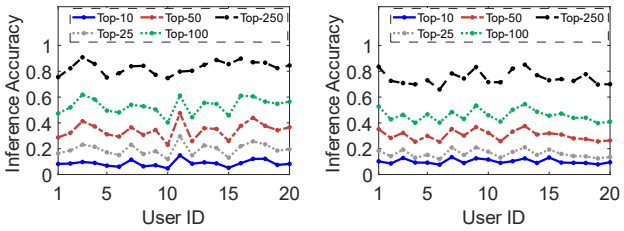
*5.4.8 Impact of Typing Non-numeric Keys.* As aforementioned, users may occasionally need to type some non-numeric keystrokes during inputting a number (e.g., for indicating the end of input or correcting input). No matter whether the OK/Enter key is used or not in the end to finish the typing session, *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 12 (b) presents the PIN entropies for different number of Delete key presses. We see that *WINK* decreases the PIN entries at a similar level regardless of the number 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.

## 6 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*.[2] We consider two general typing scenarios, as shown in Figure 13: (a) a private office room, and (b) 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.

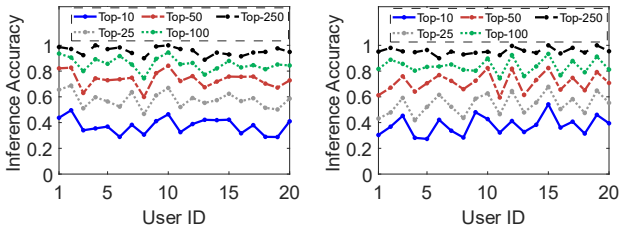
Each participant was instructed to do the following tasks:

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




(a) Office environment (b) Cafeteria environment

**Figure 14: Average top-$k$ accuracy for PIN inference.**




(a) Office environment (b) Cafeteria environment

**Figure 15: Average top-$k$ accuracy for inferring SSNs.**

- *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.

For ethical consideration, 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* $\alpha$, 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 $\alpha = 1$; 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.

**PIN Inference Results:** Figure 14 presents the PIN inference performance. 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 under the office and cafeteria environments, respectively.

**SSN Inference Results:** Figure 15 presents the SSN inference performance. 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%,

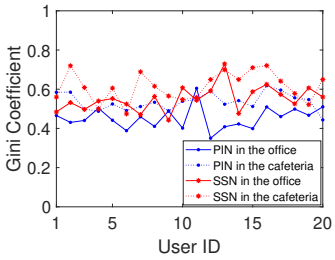



**Figure 16: Gini coefficients for chosen PINs and SSNs.**

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%.

**Security Inequity:** Figure 16 presents the corresponding Gini coefficients. We observe that all Gini coefficients are 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 9-digit SSNs are slightly larger than that for the 6-digit PINs for most users. This appears because the 3-digit area codes of some selected SSNs may disclose enough information that it becomes significantly easy to break those SSNs.

# 7 DISCUSSIONS

## 7.1 Limitations

**Typing with Two Hands or Multiple Fingers:** Our attack targets the most common scenario when the user types with one 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.

**Detecting Start of Number Entering:** In our scheme, 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 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 an extremum. This scheme is costly. External triggers such as a video feed could be added to our technique to improve the efficiency of finding the start of number entering. We leave such integration to our future work.

**Numbers in User Study:** Due to ethical reasons, we cannot have participants type PINs/SSNs they actively use. The numbers the participants select may not have good randomness, and are not participants' muscle-memorized own ones, possibly leading to slower and more cautious typing. Thus, the corresponding experimental results may not be a perfect representation of the real-world scenario. To mitigate such effects, we let participants spend more time memorizing their chosen numbers before testing.

## 7.2 Countermeasures

*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-keystrokes 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 bring inconvenience for many users who get used to input numbers using muscle memory and without any visual assistance.

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.

Alternatively, we can also directly stop the attacker from obtaining clear CSI data streams leveraging *jamming*, so that no valid spatiotemporal feature can be extracted for inferring the typed number. 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., [33]) and returns to the inactive mode when the typing ends. Compared with the previous two defenses, the jamming-based one does not require the user to change the typing process while it requires extra cost for jamming hardware.

# 8 RELATED WORK

Prior side-channel keystroke inference attacks can be mainly divided into the following categories.

**Video-based:** An attacker can stealthily record the typing process and recover keystrokes through computer vision techniques, e.g., tracking hand movement [50] or touching fingertip [61], and analyzing backside motion of input devices [54]. Recent research even shows that keystrokes can be disclosed via a video capturing eye movement [12] or a video call [48]. However, the accuracy of video-based techniques highly depends on the camera's field of view covering the victim, and the light condition in the environment.

**Sensor-based:** An attacker may use diverse onboard sensors to infer keystrokes, e.g., motion sensors (e.g., accelerometers and gyroscopes) and microphones. Most work (e.g., [8, 9, 13, 39, 40, 42, 56, 57]) require a supervised training process to build the correlation between each keystroke and corresponding sensing signal. For example, the accelerometer on a smartphone can capture vibrations caused by keypresses on a nearby physical keyboard [42] or onscreen keyboard [8], and such vibrational signals can be used for keystroke inferences. Also, recent research efforts (e.g., [39, 40, 56, 57, 59]) 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 for keystroke inference. However, to collect the sensor data, the attacker has to trick the victim to install malware onto the victim's smartwatch. [13] does not require to pre-hack the user's device, and collects acoustic emanations of 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. By incorporating the statistical constraints of the English language, [64] utilizes unsupervised training instead.

Besides, an attacker may leverage Time Difference of Arrival (TDoA) values to localize keystrokes (e.g., [37, 63]). However, such techniques have three disadvantages. First, they require multiple microphones with accurate time synchronization. Second, the victim needs to put her phone very close to the target keyboard. Lastly, the adversary must also pre-infect the victim's phone with malware, so that the intercepted acoustic signals can be sent back to her.

**Inter-keystroke timing based:** Inter-keystroke timing may leak information about the key sequences being typed. In existing work [3, 10, 53], a training process is necessary to build 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. Besides, our attack deduces inter-keystroke timing from the intercepted wireless signals. Such a method is more flexible and practical. For example, [53] learns inter-keystroke timing from the arrival times of SSH packets, and thus it needs to wait until the victim launches an SSH session. [3] uses cameras to record the typing process and extracts inter-keystroke timing from feedback on screens in the form of characters (e.g., $*$ or •), making it difficult to attack keyboards with no clear graphic feedback (e.g., a dim phone). [10] extracts inter-keystroke timing from the feedback sound when users type, and thus it must pre-install malware on the victim's device to record the acoustic signals.

Besides, [38] proposes user-independent inter-keystroke timing based attacks on PINs. It requires training to build a human cognitive model, which can be trained with data not coming from the victim. The cost of this relaxation is that [38] only works for skilled typists who share the universal typing behavioral phenomena. *WINK* requires no training and has no such restriction. Besides, *WINK* is non-invasive, while [38] requires pre-infecting the victim's device with recording malware.

**Wireless-based:** Recently, many studies have shown the success of leveraging wireless signals to infer keystrokes (e.g., [1, 21, 33, 36]). Compared with other side-channel keystroke inference attacks, wireless-based techniques have three advantages. First, wireless signals are ubiquitous and invisible, causing wireless-based attacks easy to set up and suspicious. 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 of the keystroke inference system. However, most of those wireless-based keystroke inference techniques ([1, 33, 36, 62]) still require a training process to pre-label the observed wireless signal sample with the corresponding keystroke. [21] removes the training process by exploring the context correlation which is strictly constrained by spelling and grammar of the English language, and thus it cannot be used for inferring numbers, in which digits are usually randomly combined. Also, [21] derives the *inter-element relationship matrix* to represent the structure of each English word or a CSI time-series data stream, while *WINK* defines the *structural vector* to extract the spatial features of a sequence of digits or single-keystroke waveforms. Inter-element relationship matrix and structure vector are different data structures (i.e., binary matrix vs. integer vector). Each entry in the matrix just indicates the relationship between two waveforms or characters, while each entry in the vector represents which cluster the corresponding waveform belongs to.

Another recent work [34] 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 [34] to fail in such scenarios.

## 9 CONCLUSION

We propose *WINK*, 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.

## ACKNOWLEDGMENTS

We would like to appreciate our shepherd, Dr. Jo Van Bulck, and the anonymous reviewers for their insightful comments and feedback. This work was supported in part by NSF under Grant No.1948547.

# REFERENCES


[1] Kamran Ali, Alex X. Liu, Wei Wang, and Muhammad Shahzad. 2015. Keystroke Recognition Using WiFi Signals. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking* (Paris, France) *(MobiCom '15)*. ACM, New York, NY, USA, 90–102.

[2] Kamran Ali, Alex X. Liu, Wei Wang, and Muhammad Shahzad. 2017. Recognizing Keystrokes Using WiFi Devices. *IEEE Journal on Selected Areas in Communications* 35, 5 (2017), 1175–1190.

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

[4] Ahmed Basager, Quintin Williamsand Hereford Johnson, and Prasanna Mahajan. 2020. A User-Centered Ergonomic Keyboard Design to Mitigate Work-Related Musculoskeletal Disorders. *International Journal of Ergonomics* 10, 1, 27–42.

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

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

[7] Christina Bröhl, Alexander Mertens, and Martina Ziefle. 2017. 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*. Springer, 3–16.

[8] Liang Cai and Hao Chen. 2011. TouchLogger: Inferring Keystrokes on Touch Screen from Smartphone Motion. In *Proceedings of the 6th USENIX Conference on Hot Topics in Security (HotSec'11)*. USENIX Association, USA, 9.

[9] Liang Cai and Hao Chen. 2012. On the practicality of motion based keystroke inference attack. In *International Conference on Trust and Trustworthy Computing*. Springer, 273–290.

[10] Matteo Cardaioli, Mauro Conti, Kiran Balagani, and Paolo Gasti. 2020. 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 '20)*. Springer-Verlag Italia, 720–735.

[11] Bo Chen, Vivek Yenamandra, and Kannan Srinivasan. 2015. Tracking Keystrokes Using Wireless Signals. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services* (Florence, Italy) *(MobiSys '15)*. Association for Computing Machinery, New York, NY, USA, 31–44.

[12] Yimin Chen, Tao Li, Rui Zhang, Yanchao Zhang, and Terri Hedgpeth. 2018. EyeTell: Video-Assisted Touchscreen Keystroke Inference from Eye Movements. In *2018 IEEE Symposium on Security and Privacy (SP)*. 144–160.

[13] Alberto Compagno, Mauro Conti, Daniele Lain, and Gene Tsudik. 2017. Don't Skype & Type! Acoustic Eavesdropping in Voice-Over-IP. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security* (Abu Dhabi, United Arab Emirates) *(ASIA CCS '17)*. 703–715.

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

[15] Dell. 2021. Dell Multimedia Keyboard-KB216. https://www.dell.com/en-us/shop/dell-multimedia-keyboard-kb216-us-international-qwerty-black/apd/580-admt/pc-accessories.

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

[17] Ettus Research. 2021. CBX 1200-6000 MHz Rx/Tx (40 MHz). https://www.ettus.com/all-products/cbx/.

[18] Ettus Research. 2021. LP0965 Antenna. https://www.ettus.com/all-products/lp0965/.

[19] Ettus Research. 2021. VERT2450 Antenna. https://www.ettus.com/all-products/vert2450/.

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

[21] Song Fang, Ian Markwood, Yao Liu, Shangqing Zhao, Zhuo Lu, and Haojin Zhu. 2018. 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 '18)*. ACM, Toronto, Canada, 1747–1760.

[22] Cyrus Farivar. 2015. Apple to require 6-digit passcodes on newer iPhones, iPads under iOS 9. https://www.lenovo.com/us/en/tablets/android-tablets/tab-4-series/Lenovo-TB-X704/p/ZZITZTATB1X.

[23] Fidelity Investments Inc. 2021. Fidelity's Automated Service Telephone. https://www.fidelity.com/customer-service/phone-numbers/fast/overview.

[24] Gigabyte. 2021. FORCE K83. https://www.gigabyte.com/Keyboard/FORCE-K83#kf.

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

[26] HFES 100 Committee and others. 2007. ANSI/HFES 100–2007 human factors engineering of computer workstations.

[27] Thorsten Holz, Markus Engelberth, and Felix Freiling. 2009. Learning More about the Underground Economy: A Case-Study of Keyloggers and Dropzones. In *Computer Security – ESORICS 2009*, Michael Backes and Peng Ning (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–18.

[28] Identity Theft Resource Center. 2016. *New Account Fraud—A Growing Trend in Identity Theft*. Retrieved 2021 from https://www.idtheftcenter.org/images/page-docs/NewAccountFraud.pdf

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

[30] IRS. 2021. Social Security Number Allocations. https://www.ssa.gov/employer/stateweb.htm.

[31] Lenovo. 2021. Tab 4 10 Plus. https://arstechnica.com/gadgets/2015/06/apple-to-require-6-digit-passcodes-on-newer-iphones-ipads-under-ios-9/.

[32] Hong Li, Wei Yang, Jianxin Wang, Yang Xu, and Liusheng Huang. 2016. WiFinger: Talk to Your Smart Devices with Finger-Grained Gesture. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (Heidelberg, Germany) *(UbiComp '16)*. 250–261.

[33] Mengyuan Li, Yan Meng, Junyi Liu, Haojin Zhu, Xiaohui Liang, Yao Liu, and Na Ruan. 2016. 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* (Vienna, Austria) *(CCS '16)*. 1068–1079.

[34] Zhengxiong Li, Fenglong Ma, Aditya Singh Rathore, Zhuolin Yang, Baicheng Chen, Lu Su, and Wenyao Xu. 2020. WaveSpy: Remote and Through-wall Screen Attack via mmWave Sensing. In *2020 IEEE Symposium on Security and Privacy (SP)*. 217–232.

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

[36] Kang Ling, Yuntang Liu, Ke Sun, Wei Wang, Lei Xie, and Qing Gu. 2020. SpiderMon: Towards Using Cell Towers as Illuminating Sources for Keystroke Monitoring. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM 2020)*.

[37] Jian Liu, Yan Wang, Gorkem Kar, Yingying Chen, Jie Yang, and Marco Gruteser. 2015. Snooping Keystrokes with Mm-Level Audio Ranging on a Single Phone. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking* (Paris, France) *(MobiCom '15)*. Association for Computing Machinery, New York, NY, USA, 142–154.

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

[39] Xiangyu Liu, Zhe Zhou, Wenrui Diao, Zhou Li, and Kehuan Zhang. 2015. When Good Becomes Evil: Keystroke Inference with Smartwatch. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security* (Denver, Colorado, USA) *(CCS '15)*. 1273–1285.

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

[41] Philipp Markert, Daniel V. Bailey, Maximilian Golla, Markus Dürmuth, and Adam J. Aviv. 2020. This PIN Can Be Easily Guessed: Analyzing the Security of Smartphone Unlock PINs. In *2020 IEEE Symposium on Security and Privacy (SP)*. 286–303.

[42] Philip Marquardt, Arunabh Verma, Henry Carter, and Patrick Traynor. 2011. (Sp)IPhone: Decoding Vibrations from Nearby Keyboards Using Mobile Phone Accelerometers. In *Proceedings of the 18th ACM Conference on Computer and Communications Security* (Chicago, Illinois, USA) *(CCS '11)*. 551–562.

[43] Ajaya Neupane, Md. Lutfor Rahman, and Nitesh Saxena. 2017. PEEP: Passively Eavesdropping Private Input via Brainwave Signals. In *Financial Cryptography and Data Security*, Aggelos Kiayias (Ed.). Springer International Publishing, Cham, 227–246.

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

[45] Stefano Ortolani, Cristiano Giuffrida, and Bruno Crispo. 2010. Bait Your Hook: A Novel Detection Technique for Keyloggers. In *Recent Advances in Intrusion Detection*, Somesh Jha, Robin Sommer, and Christian Kreibich (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 198–217.

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

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

[48] Mohd Sabra, Anindya Maiti, and Murtuza Jadliwala. 2021. Zoom on the Keystrokes: Exploiting Video Calls for Keystroke Inference Attacks. In *Network and Distributed Systems Security (NDSS) Symposium*.

[49] Jonathon Shlens. 2014. A Tutorial on Principal Component Analysis. *CoRR* abs/1404.1100 (2014). http://arxiv.org/abs/1404.1100

[50] Diksha Shukla, Rajesh Kumar, Abdul Serwadda, and Vir V. Phoha. 2014. Beware, Your Hands Reveal Your Secrets!. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS '14)*. ACM, 904–917.

[51] David Slater, Scott Novotney, Jessica Moore, Sean Morgan, and Scott Tenaglia. 2019. Robust Keystroke Transcription from the Acoustic Side-Channel. In *Proceedings of the 35th Annual Computer Security Applications Conference* (San Juan, Puerto Rico) *(ACSAC '19)*. 776–787.

[52] Social Security Administration. 2021. Social Security Number Randomization. https://www.ssa.gov/employer/randomization.html.

[53] Dawn Xiaodong Song, David Wagner, and Xuqing Tian. 2001. Timing Analysis of Keystrokes and Timing Attacks on SSH. In *Proceedings of the 10th Conference on USENIX Security Symposium - Volume 10 (SSYM'01)*. USENIX Association, Washington, D.C., Article 25.

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

[55] U.S. Attorney's Office. 2020. 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.

[56] Chen Wang, Xiaonan Guo, Yan Wang, Yingying Chen, and Bo Liu. 2016. Friend or Foe? Your Wearable Devices Reveal Your Personal PIN. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security* (Xi'an, China) *(ASIA CCS '16)*. 189–200.

[57] Chen Wang, Jian Liu, Xiaonan Guo, Yan Wang, and Yingying Chen. 2019. WristSpy: Snooping Passcodes in Mobile Payment Using Wrist-worn Wearables. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*. 2071–2079.

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

[59] He Wang, Ted Tsung-Te Lai, and Romit Roy Choudhury. 2015. MoLe: Motion Leaks through Smartwatch Sensors. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking* (Paris, France) *(MobiCom '15)*. 155–166.

[60] Yuxi Wang, Kaishun Wu, and Lionel M. Ni. 2017. WiFall: Device-Free Fall Detection by Wireless Networks. *IEEE Transactions on Mobile Computing* 16, 2 (2017), 581–594.

[61] Qinggang Yue, Zhen Ling, Xinwen Fu, Benyuan Liu, Kui Ren, and Wei Zhao. 2014. Blind Recognition of Touched Keys on Mobile Devices. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (Scottsdale, Arizona, USA) *(CCS '14)*. 1403–1414.

[62] Zijian Zhang, Nurilla Avazov, Jiamou Liu, Bakh Khoussainov, Xin Li, Keke Gai, and Liehuang Zhu. 2020. WiPOS: A POS Terminal Password Inference System Based on Wireless Signals. *IEEE Internet of Things Journal* 7, 8 (2020), 7506–7516.

[63] Tong Zhu, Qiang Ma, Shanfeng Zhang, and Yunhao Liu. 2014. Context-Free Attacks Using Keyboard Acoustic Emanations. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS '14)*. Association for Computing Machinery, New York, NY, USA, 453–464.

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