No Training Hurdles: Fast Training-Agnostic Attacks to Infer Your Typing

Song Fang*
University of Oklahoma
songf@ou.edu

Ian Markwood
University of South Florida
imarkwood@mail.usf.edu

Yao Liu
University of South Florida
yliu@cse.usf.edu

Shangqing Zhao
University of South Florida
shangqing@mail.usf.edu

Zhuo Lu
University of South Florida
zhuolu@usf.edu

Haojin Zhu
Shanghai Jiaotong University
zhu-hj@cs.sjtu.edu.cn

ABSTRACT

Traditional methods to eavesdrop keystrokes leverage some malware installed in a target computer to record the keystrokes for an adversary. Existing research work has identified a new class of attacks that can eavesdrop the keystrokes in a non-invasive way without infecting the target computer to install a malware. The common idea is that pressing a key of a keyboard can cause a unique and subtle environmental change, which can be captured and analyzed by the eavesdropper to learn the keystrokes. For these attacks, however, a training phase must be accomplished to establish the relationship between an observed environmental change and the action of pressing a specific key. This significantly limits the impact and practicality of these attacks.

In this paper, we discover that it is possible to design keystroke eavesdropping attacks without requiring the training phase. We create this attack based on the channel state information extracted from wireless signal. To eavesdrop keystrokes, we establish a mapping between typing each letter and its respective environmental change by exploiting the correlation among observed changes and known structures of dictionary words. We implement this attack on software-defined radio platforms and conduct a suite of experiments to validate the impact of this attack. We point out that this paper does not propose to use wireless signal for inferring keystrokes, since such work already exists. Instead, the main goal of this paper is to propose new techniques to remove the training process, which can make existing work unpractical.

CCS CONCEPTS

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

KEYWORDS

keystroke; correlation; eavesdropping attack

[*] This work was done at the University of South Florida. The author is now affiliated with the University of Oklahoma.

CCS '18, October 15–19, 2018, Toronto, ON, Canada
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5693-0/18/10…$15.00
https://doi.org/10.1145/3243734.3243755

ACM Reference Format:
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 *2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18), October 15–19, 2018, Toronto, ON, Canada.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3243734.3243755

1 INTRODUCTION

Sensitive information such as classified documents, trade secrets, or private emails are typeset and input into a computer for storage or transmission almost exclusively via a keyboard. Emerging research work has identified a new class of attacks that can eavesdrop the keystrokes in a non-invasive way [6, 7, 11, 12, 18, 21, 26, 28, 32, 37, 40, 42]. These new attacks eliminate the requirement to infect the target computer with a keylogger or other malware to violate the user's privacy. Their common underlying principle is that pressing a key on a keyboard causes subtle environmental impacts unique to that key, which can be observed and correlated for all keys. For example, an eavesdropper can set up a malicious WiFi router to receive the wireless signal emitted by a target laptop. A user pressing a key causes a unique disturbance on the received signal, and the eavesdropper can analyze these disturbances to learn which key is pressed. In general, these non-invasive keystroke eavesdropping attacks can be classified into three categories, vibration based attacks [21, 26], acoustic signal based attacks [7, 12, 37, 42], and wireless signal based attacks [6, 11, 18].

These attacks also share a common weakness, that of requiring a training phase to be at all effective. This establishes the relationships between observed environmental disturbances and specific key presses. During the attack phase, unknown disturbances are compared with those recorded in the training phase to determine which key was most likely pressed. However, the training significantly limits the impact of these attacks. Most existing works [6, 7, 11, 18, 21, 26, 28, 32, 37, 40] assume the attacker has some way to perform the training in a practical situation, but none have provided technical details justifying their logistical feasibility. [12] proposes a practical way to collect keystrokes for training by Voice-over-IP (VoIP) software (e.g., Skype), while this technique targets the scenario when the attacker is able to talk with the target user via VoIP calls.

Requiring training imposes a large practical hurdle for the attacker - most users are in full physical control of their keyboards, whether they are part of a laptop set in arbitrary locations or on a roll-out keyboard tray (a common feature of desks). Anytime a laptop is moved or a keyboard tray is pushed in or pulled out slightly, any previous training efforts are invalidated. A user may also change typing behaviors (heaviness of hand, etc.) during use of the computer. Hence, training must be conducted frequently to cope with all these changes. Because training requires knowledge of what key is pressed to construct a mapping, and therefore requires access to the system for some time, it is impossible to retrain once the user has control of the system, and it is highly difficult to train on systems controlled physically by the user (which are most).

In this paper, we make non-invasive keystroke eavesdropping practical under realistic circumstances, by removing the training requirement entirely. Not only does this make these attacks actually possible, but it also makes them far less invasive still, because physical access to the system is never required.

Intuitively, statistical methods provide a way to remove the training phase. Frequency analysis [16] is a typical unsupervised learning method based on the statistical observation that certain letters normally occur with varying frequencies in a given language. In English, the letter 'e' is the most often used. An input text of sufficiently large size will have a distribution of letter frequencies close to the typical distribution of English letters [3]. Since an environmental disturbance is associated with a key, by analyzing the frequencies of observed disturbances, the attacker can possibly determine the associated keys. Intuitively, the most frequently observed disturbance is likely to be caused by typing the letter 'e'.

However, statistical methods determine the typical distribution of English letters by ingesting a large amount of text, while the distribution within a small sample text may not be quite the same. The discrepancy between sample and typical distributions is unpredictable, so correlating observed environmental disturbances and keystrokes requires collecting statistics over a long time period, during which the environmental disturbances (e.g., wireless signal properties) for different keystrokes must remain static as well as distinct from one another. In practice, these disturbances (especially wireless signals) may change over the time due to environmental changes and mobility, preventing the attacker from collecting sufficient reliable statistics for accurate keystroke inference.

We point out that this paper does not propose to use wireless signal for keystroke inference, since existing work [6, 11, 18] has been already proposed to infer keystrokes by using wireless signal. All existing work requires a training process, which imposes a large practical hurdle for the attacker. This paper aims to remove the strong dependency of existing work on the training process to make the keystroke inference attack a practical threat.

The challenges with using statistical methods motivate us to develop an effective approach for non-invasive keyboard eavesdropping within a shorter time window. We analyze the self-contained structures of words, which can be immediately observed by typing a single word, rather than probabilistic statistics among words, which require many words to establish. In particular, we notice that the repetition or uniqueness of characters in a word shows through the structure of repeated or unique environmental disturbances collected in the process of eavesdropping. For example, assume that a user types "sense", and accordingly the attacker observes five environmental disturbances. The first and fourth observed disturbances are similar to each other, because they are caused by the action of pressing the same key "s". Similarly, the second and last disturbances appear alike, because they are caused by pressing the same key "e". This structural information enables the attacker to quickly identify the typed word, as only one word "sense" from the 1,500 most frequently used words [13] matches this structure. Thus, the search space quickly shrinks from 1,500 to only 1 word, enabling a much faster establishment of a mapping between disturbances and characters typed. This observation also requires no prior interaction with the user's system and thus facilitates fast and accurate training-agnostic keyboard eavesdropping.

To exploit this observation, we must compare the correlations among letters of words with those among observed disturbances. This requires a self-contained feature that can quantify such correlations and be compared against others. We identify and describe herein such a feature, having three necessary characteristics. First, it achieves high uniqueness to provide fast distinction among differently structured words. Second, it can be extracted both from words and sets of observed environmental disturbances, so the two can be compared. Lastly, as more words are typed, their corresponding structures can be captured and integrated with previous information to refine and shrink the search space.

Using this feature, we create approaches to compare sets of observed disturbances to possible candidate words. Our technique has mechanisms to adapt to and retain high accuracy in the presence of natural noise and sudden environmental changes, which may cause similar disturbances to appear different or vice versa. It is similarly able to continue inferring letters in the presence of non-alphabetical characters such as punctuation, navigation arrows, delete and backspace keys, etc.

Our attack analyzes disturbances in a wireless signal, which can penetrate through obstacles, so it does not require line-of-sight between the attacker and the victim. External wireless devices controlled by the attacker are used to collect the signal disturbances, so there is no need for exploits to install malware on the target computer. The attack is especially suitable for the wireless scenario, since the wireless channel is time-varying and it can quickly determine the disturbance-key relationship. Within a short time window, the attacker can apply this relationship to infer the remaining keystrokes, including typed words not in the dictionary.

We implement this attack on Universal Software Radio Peripherals (USRPs) X300 platform. The experiment results show that for a sample input of 150 words, the proposed attack can recognize an average 95.3% of these words, whereas frequency analysis can only recognize less than 2.4%. We also note that the attacker only needs 1-2 minutes to collect 50 words to identify the disturbance-key relationship that allows a word recovery rate of 94.3%. The attacker is also able to reach a word recovery ratio of 86% in the presence of a classification error rate as high as 20%. Furthermore, we show that the attacker can effectively decrease the entropy of a 9-character password from 54.8 bits to as low as 5.4 bits, vastly reducing the maximum brute-force attempts required for breaking the key from 31.08 quadrillion to just 42.

We also emphasize while the proposed attack targets English, it can be extended to other languages, because similar to English, the letters of any language are correlated and combine in some ways to form words. Thus, as long as these word structures are identified, the proposed attack can be easily customized for a target language to map correlations among observed disturbances to those among letters of words. In this paper, as a proof-of-concept, we focus on English, since it is widely used.

The rest of the paper is organized as follows. Section 2 describes background information. Section 3 explains the proposed attack and Section 4 presents experiment results. Possible defense methods are discussed in Section 5. Sections 6 and 7 lastly describe related work and conclude this paper.

2 PRELIMINARIES

Because wireless signals can penetrate through obstacles [4, 5, 27], we monitor this environment for our training-agnostic attack to remove the line-of-sight requirement. Without loss of generality, in this paper, we choose the *channel state information* (CSI) to capture the wireless signal disturbance caused by keystrokes. In the following, we impart preliminary knowledge about CSI and the general method used by existing work employing CSI to launch the keystroke eavesdropping attack.

2.1 Channel State Information

As discussed earlier, finger movement can induce disturbances into the surrounding wireless signal. The disturbances can be quantified by the CSI measurement [15], which describes how the wireless channel impacts the radio signal that propagates through the channel (e.g., amplitude attenuation and phase shift).

The orthogonal frequency-division multiplexing (OFDM) technique is widely used in modern wireless communication systems (e.g., 802.11a/g/n/ac/ad). OFDM utilizes multiple subcarrier frequencies to encode a packet, and the *channel frequency responses* measured from the subcarriers form the CSI of OFDM. The channel frequency response at time t is denoted by $H(f,t)$, where f represents a particular subcarrier frequency, and it is usually estimated by using a pseudo noise sequence that is publicly known [15]. Specifically, a transmitter sends a pseudo noise sequence over the wireless channel, and the receiver estimates the channel frequency response from the received, distorted copy and the publicly known original sequence. Let $X(f,t)$ denote the transmitted pseudo noise sequence. Based on the received signal $Y(f,t)$, $H(f,t)$ can be calculated by $H(f,t) = \frac{Y(f,t)}{X(f,t)}$. Existing work utilizes the amplitude of CSI to extract keystroke waveforms [6, 18]. In this paper, we also explore the amplitude of CSI and refer to this as just "CSI" in the following.

2.2 Existing Work on CSI-based Keystroke Inference

Researchers have proposed to utilize CSI to recognize subtle human activities, including mouth movements [34] and keystrokes [6, 18]. Existing techniques ([6, 18]) on CSI-based keystroke inference assume that the attacker typically sets up a wireless transmitter and receiver in the close proximity of the target keyboard. If the keyboard is part of a computer like a laptop that can connect to wireless networks, the computer itself transmits the wireless signal whenever it needs to exchange information with the WiFi router, and thus it can play the role of the transmitter for the attacker. The receiver can then be a malicious 802.11 access point that provides free WiFi service to attract victim computers to connect to it. In a general case, the attacker can also create a custom transmitter and receiver using software-defined radio platforms such as USRPs. The transmitter transmits the wireless signal to create a radio environment, and the receiver receives the signal from the wireless channel and computes the CSI.

These techniques normally use three steps to infer keystrokes, namely, pre-processing, training, and testing. Pre-processing removes noise from the CSI, reduces computational complexity for the keystroke inference, and segments the time series of the CSI into individual samples that correspond to keystrokes. The training phase records each keystroke and the corresponding CSI so that a training model for classification can be built. In the testing phase, an observed CSI for an unknown keystroke is matched within the training model to determine which keystroke it corresponds to. The training-agnostic attack described in this paper uses the same pre-processing step as these existing techniques.

3 ATTACK DESIGN

Existing work requires a training process to construct the relationship between observed CSI and keystrokes. We propose to remove the requirement of the training phase by quantifying the self-contained structures of words to recognize keystrokes without training. We next detail the necessary technical components we have developed.

3.1 System Overview

We consider a general attack scenario, where the attacker uses a customized transmitter and receiver pair to launch this attack. The attacker can constantly transmit the wireless signal, or just whenever typing activity is detected. In the latter case, a WiFi packet analyzer can detect when a user starts to type [18]. We also assume that the typed content is in English, though the attack can target other languages just as easily.

The receiver needs to collect the CSI, so the attacker implements a channel estimation algorithm such as the one mentioned in Section 2.1 on a software-defined radio platform. The input of the algorithm is the wireless signal received over the wireless channel, and the output is the CSI. The channel estimation algorithm computes the CSI based on the received signal, which is a continuous wave. Thus, the CSI returned by the algorithm forms a time series, and this stream is divided by the *pre-processing* step into individual segments that correspond to the actions of pressing a key. In this paper, we refer to a segment as a *CSI sample.* After pre-processing, unlike the existing methods, the training-agnostic attack described in this paper takes three different important steps to infer keystrokes, namely CSI word group generation, dictionary demodulation, and alphabet matching.

CSI word group generation partitions the CSI samples into groups corresponding to each typed word. The attacker will explore the correlation among and order of unique letters in each word to infer keystrokes, and thus needs to separate the stream into words. This step performs this task by identifying the CSI samples caused by pressing the space key, since words are almost always separated by a space. Dictionary demodulation aligns the correlation of CSI samples to that of letters in a word, so as to find the corresponding word for a CSI word group. Based on the demodulation result, potential mappings are formed between CSI samples and keystrokes, with

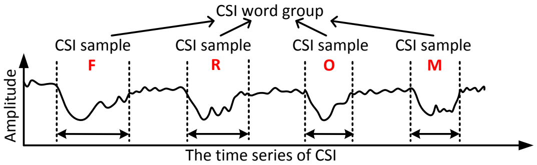


Figure 1: The CSI word group for the word "from".

which the attacker can infer the remaining typed words, including those not appearing in the dictionary.

3.2 CSI word group generation

CSI word group generation involves classification, sorting, and word segmentation.

3.2.1 Classification. Dynamic Time Warping is a classical technique to measure the similarity between two temporal sequences [29], and it has been widely used to identify the spatial similarity between the signal profiles of two wireless links [6, 17, 18, 36]. Thus, to quantify the similarity between two CSI samples, we utilize the Dynamic Time Warping technique to calculate the distance between them. A small distance indicates that both CSI samples are similar and accordingly that they originate from the same key. Conversely, a large distance indicates that they deviate from each other, and that they are caused by two different keys. We assume that the victim user presses a single key at a time, since this is the common typing behavior for most keyboard users.

3.2.2 Sorting. Since the space character is almost always used to connect consecutive words, it normally appears more frequently than any other characters in a long text. We thus expect that the CSI sample caused by the space key also appears more frequently than other CSI samples. The classification outcome includes multiple sets, each consisting of similar CSI samples. We sort the sets according to size and associate the space key with the largest set, so that all observed CSI samples in this set are assumed to be caused by pressing the spacebar. If this association is incorrect, we will ultimately not be able to recover meaningful English words. In that case, we continue on, associating the space key to the second largest set and reattempting the same recovery process. We try these sets from largest to smallest cardinality until we successfully recover meaningful English words or exhaust all sets.

3.2.3 Word Segmentation. Once the set of CSI samples associated with the space key is identified, we can start the word segmentation process to find the CSI samples comprising each word of the typed content. Everything between two successive CSI samples from the space-associated set are grouped together. In the following, we refer to such a group as a *CSI word group*, and this does not include the spaces at either end. CSI word groups will be used as the input of the dictionary demodulation method to eventually establish the complete mapping between the CSI samples and keystrokes. Figure 1 is an example of the CSI word group for the word "from" which consists of samples that are caused by typing letters 'f', 'r', 'o', and 'm'.

3.3 Dictionary Demodulation

Dictionary demodulation converts CSI word groups to corresponding English words. We begin by developing a feature to apply to these CSI word groups suitable for narrowing down the search space of possible candidates. Then we show how to apply this feature to words and sentences and handle errors.

3.3.1 Feature Selection. Ideally, a feature extracted from each CSI word group would enable us to uniquely determine the corresponding word. If the dictionary has n words, a perfect feature would classify the n words into n groups, each having one member only, such that an input CSI word group can uniquely match to a word based on this feature. Our strategy is thus to find a feature that can divide all words in the dictionary into as many sets as possible, to achieve high distinguishability.

Due to the lack of training, we have to identify a feature from only the self-contained relationships among the letters of a word (the CSI samples of a CSI word group). Without knowing the exact letters in a word, but having a CSI sample for each letter, we can determine the number of constituent letters and whether or not any letters in the word are repeated. These two pieces of information yield two features to partition words, and we utilize a top 1,500 most frequently used word list [13] as the dictionary to calculate the number of sets divided by each. To quantify the distinguishability of a feature, we define a new metric, called the *uniqueness rate*, as the ratio T_p/T, where T is the number of considered words, and T_p represents the number of sets obtained by dividing T words according the selected feature. The uniqueness rate should be maximized for the best partitioning of the words. We next evaluate the uniqueness rates for our two features:

Length: We empirically find that all words in this dictionary are 1-14 characters long. If we choose length as the only feature, we can divide all words into 14 sets, the members of each set having the same length. Only two words (i.e., 'administration' and 'responsibility') in the dictionary are of length 14; therefore a CSI word group of length 14 has only two candidates. On average, however, each set has $1,500/14 \approx 107$ words. This means that an input CSI word group will have an average of 107 possible candidate words based on the length feature. The uniqueness rate is then $14/1,500 \approx 0.009$.

CSI Sample Repetition: We also count the number of distinct letters that repeat. We denote the repetition information of a word as S_r, and we set $S_r = 0$ if no repetition is found. Otherwise, we denote S_r by $(t_1, \cdots, t_r)$, where r is the number of distinct letters that repeat, and t_i $(i \in \{1, \cdots, r\})$ denotes how many times the corresponding letter repeats. For example, the repetition information for the word "level" should be (2, 2), because 2 different letters ('l' and 'e') repeat, and both letters repeat twice respectively. Considering a word of length L, we can quantify the repetition information using (L, S_r). Using this repetition information, we can then divide all 1,500 words into a total of 63 sets, such that members of each set share the same value of (L, S_r). On average, each set has $1,500/63 \approx 24$ words, so an input CSI word group will be mapped to one of 24 words based on this feature. The uniqueness rate is then $63/1,500 \approx 0.042$.

The repetition feature has better distinguishability than the length feature, because its larger uniqueness rate yields a smaller average set cardinality, and hence a reduced search space to map

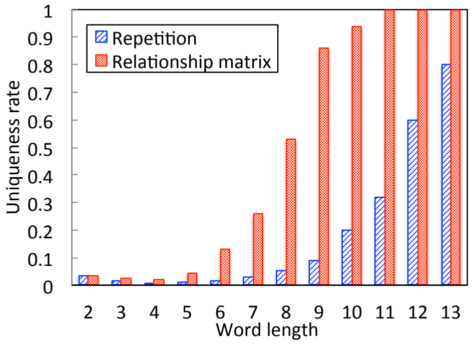


Figure 2: Uniqueness rate for words of different length.

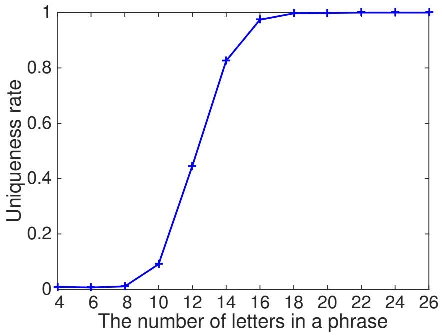


Figure 3: Uniqueness rate for joint words.

an input CSI word group to a word. The repetition feature only provides the result of repeated letters in a word, however, and does not consider the position information of these letters. We expect that the uniqueness rate can be further increased if we construct a feature that not only employs the word length and repetition information, but also distinguishes the positions of repeated letters from non-repeated letters.

3.3.2 Inter-Element Relationship Matrix. We define a new data structure to represent the structure of every word/CSI word group. Specifically, we denote a word or a CSI word group by a vector $[x_1, \ldots, x_n]$ of n elements, each of which is a letter (CSI sample). We then define its *inter-element relationship matrix* as

$$M : [x_1, \ldots, x_n] \mapsto \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & \cdots & r_{1,n} \\ r_{2,1} & r_{2,2} & r_{2,3} & \cdots & r_{2,n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ r_{n,1} & r_{n,2} & r_{n,3} & \cdots & r_{n,n} \end{bmatrix}.$$

To construct M, for a CSI word group $[x_1, \ldots, x_n]$, we set $r_{i,j} = 1$ if x_i and x_j are similar CSI samples as classified during the CSI word group generation step (Section 3.2). Otherwise, we set $r_{i,j} = 0$. Note the diagonal elements are always 1 and the matrix is symmetric.

We build the inter-element relationship matrix for each word and ultimately partition the 1,500 most commonly used words into 337 sets. The words in a particular set having the same inter-element relationship matrix. On average, each set has about $1,500/337 \approx 4$ words which are possible candidates for the CSI word group having that inter-element relationship matrix. The corresponding uniqueness rate is $337/1,500 \approx 0.225$, which is much larger than those of the previously discussed features.

Empirically, we find that the uniqueness rates for words of different lengths are not evenly distributed, and this fact actually enables our scheme. Figure 2 presents the uniqueness rates for the inter-element relationship matrix as well as the repetition feature for comparison, respective to word length. The relationship matrix clearly performs much better than the repetition feature in all cases, but very evident also is that as words become larger, they become more uniquely structured, leading to high uniqueness rates for the relationship matrix feature. For example, the uniqueness rate for a 3 letter word is 0.025, while that for a word of 10 letters is 0.940.

Indeed, a phrase comprised of multiple words can be considered as one "long word" for the purpose of generating an inter-element relationship matrix, though the dictionary must also expand to contain these combinations. Assuming a phrase formed by N words, the new dictionary will include $T_1 T_2 \cdots T_N$ phrases, where T_i ($1 \le i \le N$) is the size of the set of candidate words having length equal to the i-th CSI word group. Figure 3 illustrates how the uniqueness rate benefits from the combination of each pair of two words from the dictionary of 1,500 most used words. The words in each pair range from 2 to 13 characters in length, for a possible total of 4-26 characters. The uniqueness rate jumps as the length of these word pairs increases, and after 18 total characters, the pair of words has a fully unique structure. This indicates that within a few words it should always be possible to narrow down to the specific content the victim is typing, giving rise to our joint demodulation method.

3.3.3 Joint Demodulation Example. Before describing the general joint demodulation technique, we first show a simple clarifying example to illustrate how to demodulate the CSI word groups. Assume that a simple dictionary $\mathbf{W}$ = {'among', 'apple', 'are', 'hat', 'honey', 'hope', 'old', 'offer', 'pen'}. Further assume that the user types in two words "apple" and "pen". We denote the CSI word groups corresponding to these typed words by $c_1||c_2||c_3||c_4||c_5$ and $c_6||c_7||c_8$, respectively, where c_i is the i-th observed CSI sample after identification and removal of spaces.

Due to the previously discussed consistency between CSI samples for the same character, samples c_2 and c_3 within the first CSI word group are similar. The inter-element relationship matrix R_1 is correspondingly

$$\begin{array}{c|ccccc} & c_1 & c_2 & c_3 & c_4 & c_5 \\ \hline c_1 & 1 & 0 & 0 & 0 & 0 \\ c_2 & 0 & 1 & 1 & 0 & 0 \\ c_3 & 0 & 1 & 1 & 0 & 0 \\ c_4 & 0 & 0 & 0 & 1 & 0 \\ c_5 & 0 & 0 & 0 & 0 & 1 \end{array}.$$

Matrix R_2 is constructed in the same way for the second word. Combining both words and considering all similar CSI samples forms the joint sequence $c_1||c_2||c_3||c_4||c_5||c_6||c_7||c_8$, and likewise the new inter-element relationship matrix R_{new} can be obtained. To search the dictionary $\mathbf{W}$ for these CSI word groups, we pre-compute the inter-element relationship matrix for each word in $\mathbf{W}$. We compare each with R_1 and find the words "apple" and "offer" survive this test. Each is compared with R_2, and "hat", "old", "are", and "pen" survive the test for the second word group. The 8 resulting candidates for the two-word sequence are thus "apple||are", "apple||hat", "apple||old", "apple||pen", "offer||are", "offer||hat", "offer||old", and "offer||pen". We use $\mathbf{T}$ to denote the set of these candidates. Again, we generate the relationship matrix for each new candidate in **T** and compare it with R_{new}. Only "apple||pen" survives this test and must be the final result.

3.3.4 *General Joint Demodulation Method.*

In the following, we extend the joint demodulation method utilized in this example to m CSI word groups. After CSI word group generation, assume that the attacker obtains from the eavesdropped typing m CSI word groups denoted by $\mathbf{S} = \{S_1, S_2, \ldots, S_m\}$. We further use $W_1, W_2, \ldots, W_q$ to denote the q words in the dictionary **W**. Our goal is to find a phrase of m words that corresponds to the m CSI word groups. Clearly, while each individual CSI word group could have several candidate dictionary words with matching structure, each candidate will impose a mapping of some CSI samples and letters on some successive words, and several of these possible mappings will result in successive words that are not real, so the below technique works to rule out these impossible mappings. The full method includes two steps: 1) demodulation of each single CSI word group; and 2) joint demodulation of multiple CSI word groups.

Step 1: This step finds initial candidate words for each CSI word group or determines if a word cannot be immediately demodulated and must be returned to later. We first create the inter-element relationship matrices for $W_1, W_2, \ldots, W_q$ in our dictionary **W**. We next iterate over each $S_i \in \mathbf{S}$, creating its inter-element relationship matrix and considering the subset $\mathbf{W}'$ of **W** whose entries are of the same length as S_i. We compare the relationship matrix of S_i to that of each $W_j \in \mathbf{W}'$ and mark that W_j as a candidate if the two matrices are equal. If no candidates match, the word must not appear in the collection of English words comprising our dictionary, so we add S_i to the "undemodulated set" **U**.

Step 2: This step works to build up a mapping between CSI samples and letters that works for multiple CSI word groups simultaneously, successively ruling out the many candidates established by the first step, until (ideally) only one candidate remains for each word and the message is uncovered. Conceptually, we iterate over the word groups not in the undemodulated set **U**,

(a) concatenating each with all those previous,
(b) applying each possible mapping thus far constructed,
(c) ruling out all candidates that cannot coexist with any mappings,
(d) and adding any new CSI sample/character mapping information from the remaining candidates.

Specifically, we name T_i the concatenation of the first $i-1$ CSI word groups $\{S_1, \ldots, S_{i-1}\}, 1 < i \leq m$, excluding any $S_k \in \mathbf{U}$. In other words, while considering S_i, we concatenate all the previous CSI word groups which have candidates into T_i. Candidates for T_i, or groups of valid words satisfying the structures of the CSI samples comprising T_i, are denoted by $\mathbf{T}_{i_C} = \{T_{i_1}, T_{i_2}, \ldots, T_{i_p}\}$. Further, candidates for S_i, as determined by Step 1, are denoted by $\mathbf{S}_{i_C} = \{S_{i_1}, S_{i_2}, \ldots, S_{i_q}\}$. With $T_i||S_i$ signifying the concatenation of T_i and S_i, we calculate the inter-element relationship matrix for $T_i||S_i$, as well as that for every $T_{i_j}||S_{i_k}, T_{i_j} \in \mathbf{T}_{i_C}, S_{i_k} \in \mathbf{S}_{i_C}$. We note that this is $p \times q$ matrices to be compared and that this series of comparisons happens at each iteration; we analyze the time complexity in Section 4.3, and our experiments show the number of comparisons converges quickly over successive iterations. Then, if the relationship matrix for one such $T_{i_j}||S_{i_k}$ matches that for $T_i||S_i$, we know that the CSI sample/character mapping of the candidate S_{i_k} will work in concordance with the mapping established for T_{i_j} while maintaining the structure stipulated by $T_i||S_i$. Each such $T_{i_j}||S_{i_k}$ is therefore a new candidate for T_{i+1}.

Algorithm 1 Joint Demodulation

```
procedure JOINT_DEMOD($S_i$, $\mathbf{T}_{i_C}$, $S_{i_C}$, $\mathbf{U}$)
    $\mathbf{T}_{(i+1)_C} \leftarrow \emptyset$ $(i > 0)$
    for $T_{i_j}$ in $\mathbf{T}_{i_C}$ do
        for $S_{i_k}$ in $\mathbf{S}_{i_C}$ do
            if $M(T_i||S_i)$=$M(T_{i_j}||S_{i_k})$ then
                $\mathbf{T}_{(i+1)_C} \leftarrow \mathbf{T}_{(i+1)_C} \cup T_{i_j}||S_{i_k}$
            end if
        end for
    end for
    if $\mathbf{T}_{(i+1)_C} = \emptyset$ then                ▷ no candidates, skip $S_i$
        $\mathbf{U} \leftarrow \mathbf{U} \cup S_i$
        $\mathbf{T}_{(i+1)_C} \leftarrow \mathbf{T}_{i_C}$
    end if
    return $\mathbf{T}_{(i+1)_C}$, $\mathbf{U}$
end procedure
```

In the event that no $T_{i_j}||S_{i_k}$ has a relationship matrix matching that for $T_i||S_i$, this means that no CSI sample/character mappings satisfying the structure of T_i result in valid words within our dictionary when applied to S_i. Such S_i are placed in **U** and execution skips to S_{i+1}. Pseudocode for this step is shown in Algorithm 1. In this manner, we iterate over i and gradually build up T_i until all distinct CSI samples are mapped to characters in the alphabet. At this time, the mapping can be applied to the remaining word groups, including those in **U**, for which no matches were found in the dictionary used. An example of this final alphabet matching is visible in Figure 4.

3.3.5 *Error tolerance.*

Wireless channel noise may cause CSI classification errors, such that a recorded CSI sample for a character typed might not appear like others for that character or may appear like a different character. Otherwise, CSI samples may be classified correctly but a typo by the user may mean a word is misspelled and will not appear in the dictionary. This can cause a concatenated set of CSI word groups to have an incorrect inter-element relationship matrix, which may match with invalid words or have no candidates at all. The latter is the ideal case as the word group having the CSI sample in question will simply be added to the undemodulated set and skipped. However, if invalid words are incorporated into the candidates for joint demodulation, incorrect relationship matrices will continue to be used as the joint demodulation progresses, and the content recovery will fail. We have observed in experiments that even if a wrong matrix matches to other word sequences, cascading discovery failures inevitably happen for successive words.

The attacker may employ this observation to work around the presence of typos or CSI classification errors. If a CSI word group is successfully demodulated but continuous recovery failures occur thereafter, this word can be added to the undemodulated set and skipped in favor of proceeding with the next word. Further word groups are thus less likely to be processed with an incorrect portion of the relationship matrix, and a correct mapping is more probable. Algorithm 2 shows how our system checks for cascading errors at

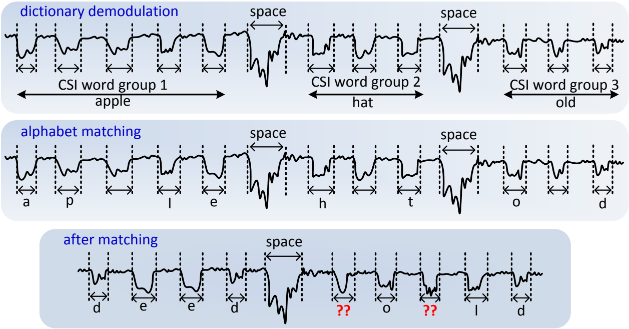


Figure 4: Assume a simple dictionary of three words "apple", "hat", and "old", typed in that order by the user. The alphabet of this dictionary consists of 8 letters "a","p","l","e","h","t","o", and "d". Dictionary demodulation maps each letter in this alphabet to the corresponding CSI sample, and any further CSI word groups may simply have this mapping applied to them. After matching, suppose the user then types the word "deed", the attacker can directly demodulate the observed CSI word group, which did not appear in the dictionary. Next, assume instead the second typed word is "would". Since "w" and "u" do not appear in the alphabet of this simple dictionary, the attacker cannot decode them but can continue decoding the other letters "o", "l", and "d".

Algorithm 2 Error Handling

```
[$\mathrm{T}_{(i+1)_C}$,U] = JOINT_DEMOD($S_i$,$\mathrm{T}_{i_C}$,$S_{i_C}$,U)
if $\mathrm{T}_{(i+1)_C} \neq \mathrm{T}_{i_C}$ then                ▷ demodulation success
    $F \leftarrow$ allowable failure threshold
    $flag \leftarrow$ true
    for $j \in \{i+1, \cdots, i+F\}$ do
        [$\mathrm{T}_{(j+1)_C}$,U]=JOINT_DEMOD($S_j$,$\mathrm{T}_{j_C}$,$S_{j_C}$,U)
        if $\mathrm{T}_{(j+1)_C} \neq \mathrm{T}_{j_C}$ then        ▷ demodulation success
            $flag \leftarrow$ false; break                 ▷ reset failure count
        end if
    end for
    if $flag$ then                             ▷ reached failure threshold
        $\mathrm{U} \leftarrow \mathrm{U} \cup S_i$                              ▷ skip $S_i$
        $\mathrm{T}_{(i+1)_C} \leftarrow \mathrm{T}_{i_C}$
    end if
end if
```

each step i based on the demodulation result for S_i. Finally, when the mapping is complete and applied to the CSI word groups in the undemodulated set, any errors in CSI classification or typos will persist, but not further damage the results. The attacker can use some common knowledge to work out these errors and any other ambiguities.

In the event the cascading errors do not seem to be avoidable, this is evidence that the wireless channel has changed, because as previously mentioned the wireless channel is time-varying. In this case, the dictionary demodulation may be begun anew, so that the attack can adapt to the changes.

3.3.6 *Impact of Non-Alphabetical Characters.*

Users mostly type alphabetical characters and spaces, but also occasionally use numbers and punctuation, which obviously cannot be matched by examining word structures. If these appear during alphabet mapping construction, they will cause cascading demodulation errors, be added to the undemodulated set, and be skipped, similar to typos or CSI classification errors as just discussed. If the mapping has already been constructed, the CSI samples for these numbers or punctuation will not appear in the mapping and will be left as unknown. In both cases, the attacker can use some common knowledge to infer or narrow down candidates for these characters.

For example, users press the backspace key to remove multiple characters before the cursor and then continue typing. For a CSI word group that is recovered as "abab××out", the attacker may recognize that the unidentified character "×" corresponds to the backspace key and that the word should be "about". In another case, a user may press the left arrow key to move the cursor backward, insert some text, and then press the right arrow key to return the cursor to the original position. Hence, the left and right arrow keys often appear in pairs and are each pressed multiple times. In this way, an attacker may infer the word "about" from a CSI word group recovered as "aut◃◃bo▹▹", with unidentified samples "◃" and "▹" corresponding to left and right arrow keys, respectively.

4 EXPERIMENT RESULTS

We implement the training-agnostic keystroke eavesdropping attack using USRPs. The prototype attack system includes a wireless transmitter and a receiver. Each node is a USRP X300 with 40 MHz bandwidth CBX daughterboards [14]. The channel estimation algorithm runs at the receiver to extract the CSI for key inference.

The target user operates a desktop computer with a Dell SK-8115 USB wired standard keyboard. The transmitter and the receiver are placed at opposite positions relative to the keyboard. We place the transmitter at a distance of 3 meters away from the keyboard, and the receiver under the 2 cm-thick desk, at a distance of 50 cm away from the keyboard. Also, there is a 4 cm-thick wooden barrier between the transmitter and the keyboard. Thus, both the transmitter and the receiver are not within line-of-sight of the target user. We also form a dictionary using the top 1,500 most frequently used English words [13].

4.1 Example Recovery Process

In this section, we will demonstrate the process of recovering a sample user's typed text to illustrate the attack and the sort of performance that may be expected.

CSI Sample Extraction: To extract the CSI samples from the CSI time series, we utilize the same pre-processing step as these existing techniques [6, 18]. Correspondingly, we implement the pre-processing through three phases, which are noise removal, Principle Component Analysis (PCA) [30], and segmentation. First, we experimentally observe the frequency of the CSI influenced by keystrokes always lies within a low frequency range of 2 to 30 Hz. Thus, we utilize a *Butterworth* low-pass filter [24] to mitigate the impact of ambient noise, which normally has a higher frequency. Initially, the receiver obtains CSI from all subcarriers. We then apply the PCA technique to decrease computational complexity by

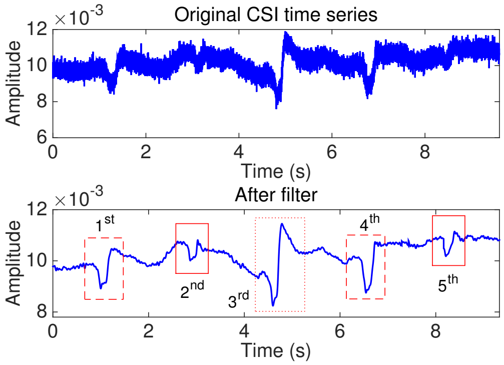


Figure 5: The CSI word group for the word “sense”.

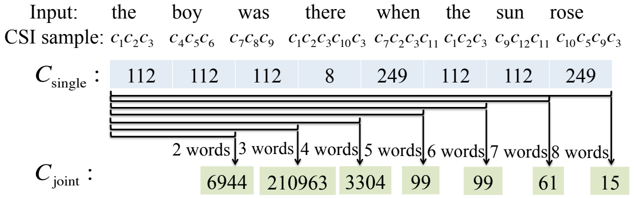

Input: the boy was there when the sun rose
CSI sample: $c_1c_2c_3$ $c_4c_5c_6$ $c_7c_8c_9$ $c_1c_2c_3c_{10}c_3$ $c_7c_2c_3c_{11}$ $c_1c_2c_3$ $c_9c_{12}c_{11}$ $c_{10}c_5c_9c_3$
C_{single} : 112 112 112 8 249 112 112 249
2 words
3 words
4 words
5 words
6 words
7 words
8 words
C_{joint} : 6944 210963 3304 99 99 61 15

Figure 6: The evolution of the amount of candidates returned as words are processed.

converting the received CSI into a set of orthogonal components, called principle components [30], which most represent the effects of the keystrokes. The segmentation phase separates the full CSI time series into the individual CSI samples corresponding to single unknown keystrokes. After the receiver assigns the space character to the most frequently appearing CSI sample group, the remaining samples are grouped into CSI word groups. Figure 5 shows the CSI word group for the word “sense”. The full data contains five CSI samples caused by pressing the keys ‘s’, ‘e’, ‘n’, ‘s’, and ‘e’ as visible on the figure. By using Dynamic Time Warping, we can classify the five samples into three sets, including the pair of the first and fourth samples, the pair of the second and fifth samples, and the third sample alone.

Next, we illustrate how the collected CSI word groups can be narrowed down to the typed content. We choose the Harvard sentences [23] to be typed in for our experiments; these are phonetically balanced sentences commonly used for testing speech recognition techniques. For this example recovery, we randomly select five sentences from these representative English sentences, with a total of 41 words. While we process the collected CSI, we record C_{single}, which is the number of words that have the same inter-element relationship matrix as the current CSI word group under processing, and C_{joint}, which is the number of candidates returned by the joint demodulation algorithm for each CSI word group.

Figure 6 shows C_{single} and C_{joint} during the processing of this sentence. To facilitate understanding, we also mark the CSI sample sets on this figure. For example, f_1, f_2, and f_3 represent the CSI sample sets caused by typing the letters ‘t’, ‘h’, ‘e’, respectively. We can see that C_{single} is 112 for three-letter words, and consequently C_{joint} increases dramatically from 112 to 6,944 and then to 210,963 as the second and third CSI word groups are added, as these word groups share no common CSI samples. However, as more CSI word groups are added, the joint demodulation algorithm finds more common CSI samples, which shrinks the search space. C_{joint} drops sharply from 210,963 to 3,304 after the fourth CSI word group is processed, and further reduces to 15 as the remaining CSI word groups are processed.

Input paragraph: *The boy was there when the sun rose. A rod is used to catch pink salmon. The source of the huge river is the clear spring. Kick the ball straight and follow through. Help the woman get back to her feet.*

Step 1 Searching results:

The boy/box was there when the sun rose. A *** is used to catch **** *****. The source of the huge river is the clear spring. **** the ball straight and follow through. Help the woman get back to her ****.

Step 2 Recovering words not in the dictionary:

(1) rod; (2) pink; (3) salmon; (4) Kick; (5) feet.

Figure 7: Example paragraph recovery.

After the demodulation phase, two candidates are returned as shown in Figure 7. The two differ by only one word; the second word is either “boy" or “box". Even for the wrong candidate, 97.6% of the words are successfully recovered, and all characters except one. The example paragraph also contains five words (“rod”, “pink”, “salmon”, “kick”, and “feet”) that are not in the dictionary. These are still successfully recovered, however, because their constituent CSI samples also appear in other words, and their sample/letter mappings have already been determined by the matching phase.

4.2 Eavesdropping Accuracy

We now quantify the general performance of our attack. We define the word recovery ratio as the ratio of successfully recovered words to the total number of input words. We employ this metric to ascertain the accuracy of our attack using 100 online articles randomly selected from CNN, New York Times, and Voice of America. For comparison purposes, we also apply the traditional frequency analysis technique to the segmented CSI samples.

4.2.1 Single Article Recovery. We first type a piece of CNN news [2] into a computer, and collect the CSI while typing. We then extract the CSI samples and run the demodulation algorithm. Suppose the demodulation algorithm returns N candidates for the typed content. We use WRR_i ($i \in \{1, \cdots, N\}$) to denote the word recovery ratio for the i^{th} candidate. We consider the overall word recovery ratio WRR of the proposed attack to be calculated as the average of these word recovery ratios for each candidate: $WRR = \sum_{i=1}^{N} \frac{WRR_i}{N}$.

Figure 8 shows the overall word recovery ratio as a function of the number of typed words. We can observe for the first couple of typed words, the ratio is less than 0.17, because these words are not in the dictionary or the joint demodulation algorithm returns wrong candidates. As more words are typed in, the ratio increases significantly and fluctuates, since newly typed words may or may not be identified correctly in the various candidates. After a sufficient number of words are typed, the mapping between CSI samples and the alphabetic letters converges to only one candidate. As a result, the word recovery ratio stabilizes at a high value. As shown

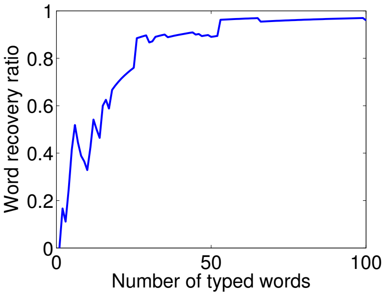
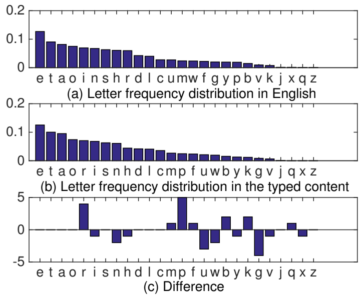


Figure 8: WRR vs. word count

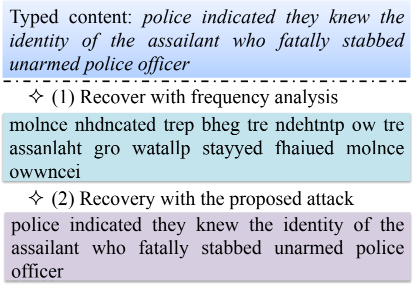


(a) Letter frequency distribution in English

(b) Letter frequency distribution in the typed content

(c) Difference

Figure 9: Comparing distributions.



Typed content: *police indicated they knew the identity of the assailant who fatally stabbed unarmed police officer*

✧ (1) Recover with frequency analysis

molnce nhdncated trep bheg tre ndehtntp ow tre assanlaht gro watallp stayyed fhaiued molnce owwncei

✧ (2) Recovery with the proposed attack

police indicated they knew the identity of the assailant who fatally stabbed unarmed police officer

Figure 10: Recovered words.

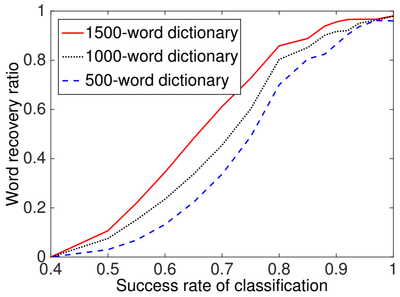


Figure 11: Word recovery ratios vs. CSI sample classification errors.

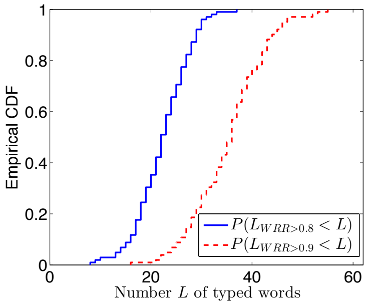


Figure 12: CDFs of $L_{WRR>0.8}$ and $L_{WRR>0.9}$.

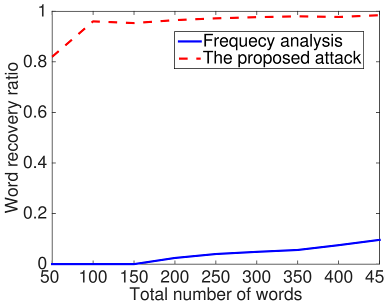


Figure 13: Frequency analysis vs. the proposed attack.

in Figure 8, when more than 52 words have been typed, the overall word recovery ratio remains above 0.96.

For meaningful results, we apply the frequency analysis recovery technique to compare with our method. Figure 9(a) shows the typical distribution of frequencies of English letters [16], while Figure 9(b) shows the distribution of letters in the typed text. Because the typed text is short and not representative of the whole English language, the sample distribution is not perfectly equal to the typical distribution. This difference is highlighted in Figure 9(c) and causes the word recovery ratio for the frequency analysis to be as low as 0.07. Figure 10 shows parts of the recovery results using the frequency analysis and our method. The content recovered using the frequency analysis is meaningless, whereas our new attack successfully recovers the typed words.

Impact of CSI sample classification errors and dictionary size: As discussed in Section 3.3.5, errors in grouping CSI samples during the pre-processing step may occasionally lead to a failure in demodulating a CSI word group when the pattern of the word is not correctly detected. To test the impact of this on the overall word recovery ratio, we artificially introduce errors into the groupings and attempt the demodulation algorithm using the intentionally incorrect data. Specifically, we vary the number of correctly grouped CSI samples from 40% to 100% in intervals of 5%, and measure the resulting overall word recovery ratio. We also examine the effects of using dictionaries of three different sizes, including the 500, 1000, and 1500 most frequently used words.

We repeat this experiment 10 times and present the average results in Figure 11. Intuitively, more correctly classified CSI samples result in higher word recovery ratios, as do larger dictionaries. Nonetheless, we also note that only 80% of CSI samples need be correctly classified for the overall word recovery ratios to achieve 0.86, 0.81, and 0.7 for the various dictionary sizes.

4.2.2 Average Article Recovery. We repeat the above experiment for 100 online articles. Intuitively based on the discussed observations, the proposed attack should achieve a high word recovery ratio for a long text. Considering a desired overall word recovery ratio of 0.8 or 0.9, let $L_{WRR>0.8}$ and $L_{WRR>0.9}$ denote the required number of typed words from each article to satisfy those ratios, respectively. Figure 12 shows the empirical cumulative distribution functions (CDFs) of $L_{WRR>0.8}$ and $L_{WRR>0.9}$, indicating conclusively longer input text results in higher word recovery ratios. Specifically, for more than 82.4% of articles, the achieved word recovery ratio is greater than 0.8 and 0.9 when the number of these words is greater than 27 and 42, respectively.

Figure 13 compares the efficacy of our attack and the frequency analysis technique. Our attack can achieve a 0.82 word recovery ratio after 50 typed words, whereas the frequency analysis requires typing 150 words before any can be successfully recovered. Indeed, the highest ratio achieved by the frequency analysis in these online articles is around 0.1, after 450 words, while in stark contrast our attack stabilizes around 0.95 after 150 words.

4.3 Time Complexity Analysis

As our attack requires no training, its processing time is naturally of interest. The comparison of inter-element relationship matrices is the dominant part of the demodulation phase of our dictionary demodulation algorithm, so we count the required comparisons to calculate complexity. Dictionary sizes have a bearing on this count, as a larger dictionary results in more candidate words found for a

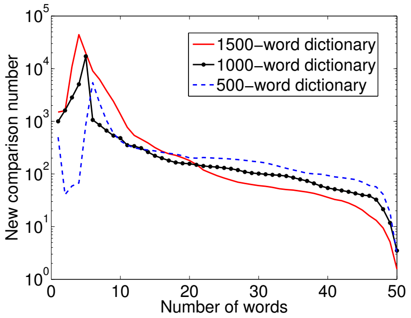


Figure 14: New comparisons vs. the number of processed words.

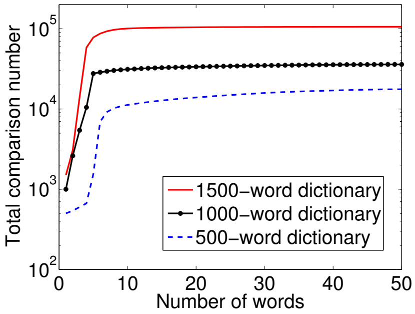


Figure 15: Total comparisons vs. the number of processed words.

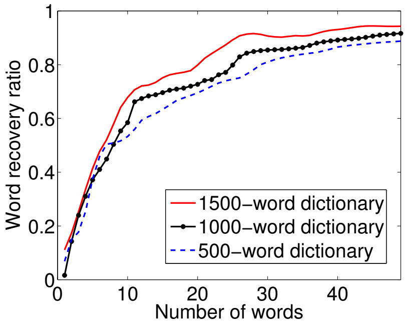


Figure 16: Recovery of volunteer-typed "secrets".

CSI word group and thus a larger search space. Consequently we examine the time complexity while using three different dictionaries, which contain the top 500, 1000, and 1500 most frequently used words [13].

During the 100 experiments discussed in Section 4.2, we count the comparisons needed to generate candidates for the typed content each time a new CSI word group is added to the dictionary demodulation process. Figure 14 shows the average comparison number for each newly typed word, on a log scale. This number greatly increases for the first few typed words but promptly decreases to a low value below 10 as more words are typed. This was seen for a single sentence in Figure 6 and holds true for these 100 trials as well. The addition of more unique letters results in a vastly enlarged search space, while the later inclusion of more repeated letters imposes a structure to the words and quickly reduces the search space again. For a 1500-word dictionary, the average number of required comparisons for the 4^{th} word is 4.4×10^4, while it becomes 1.6 for the 50^{th} word.

Interestingly, the search space for a larger dictionary shrinks faster than that of a smaller dictionary as more words are typed, despite being larger after the first few words. For example, for the 45^{th} word, the average numbers of required comparisons for the 1500-, 1000-, and 500-word dictionaries are 20.6, 39.7, and 70.1, respectively. At first, a larger dictionary will find more matches for the word structures searched, but this quickly narrows down as repeated letters are added. Conversely, a smaller dictionary has a lower probability of finding candidate words for a particular structure, leading to skipped words, and therefore requiring more typed words before repeated letters can appear and reduce the search space.

Figure 15 shows the cumulative average comparison numbers (i.e., the total time complexities) as more words are typed, for the three dictionary sizes. The time is clearly spent mostly on inferring the first couple of words, after which the total time complexity stabilizes. This trend is the same for all dictionaries, though larger dictionaries see distinctly more total comparisons and consequently higher time complexity. Larger dictionaries also stabilize faster, however; the 1500-, 1000-, and 500-word dictionaries stabilize at 8, 11, and 15 typed words, respectively.

4.4 An Example of the Attack

We recruited 10 volunteers and asked each of them to type a paragraph of "secret" content for us to attempt to eavesdrop. For ethical reasons, we did not ask them to type actual secrets or content that they would wish to keep private, but simply to type comprehensible English content which we did not provide them.

While each volunteer typed, the receiver continuously collected CSI data, which was then processed and the eavesdropping result presented to the volunteer. Volunteers compared the recovered content with their typed content to quantify the word recovery rate of the proposed attack. Figure 16 shows the resulting average word recovery ratios as each word is typed and with the use of the three different dictionary sizes. In practice, the proposed attack achieves a word recovery ratio of more than 0.8 after 28 words are typed, regardless of dictionary size. Additionally, a larger dictionary yields a higher word recovery ratio. With more than 40 typed words and a dictionary of 1,500 words, the ratio exceeds 0.94. This demonstrates that the proposed attack can recover typed secrets effectively and efficiently in a real-world setting.

4.5 Password Entropy Reduction

Modern password requirements include letters, numbers, and special characters. The strength of a password lies in its resistance to brute-force guessing attacks, and is a function of length, complexity, and unpredictability. Our unobtrusive keyboard eavesdropping attack focuses on inferring letters, but it can still greatly decrease password strength. As users normally type both passwords and comprehensive English content during typical computer usage, we can apply the alphabet matching afforded by the latter to infer significant portions of the former.

Typical users usually pick fewer non-letter characters in their password in order to make it easy to remember, leaving the password more vulnerable to these attacks. We did preliminary experiments to evaluate the entropy reduction impact using the password list, which contains 342,508 passwords leaked from Yahoo! Voices [1]. Figure 17 shows the average ratios of letter characters in passwords with different lengths. We observe that the ratio of letters in a password with a length ranging from 6 to 12 lies between 0.65 to 0.73, and also with the key length increasing, the ratio of letters slightly increases. Analyzing the leaked passwords, we found that 98.42% of passwords are 12 characters or fewer, and people utilize an average of 8.72 letters for a 12-character password. This means that the difficulty for guessing a 12-character random password is reduced to that for guessing an extremely weak password of 3-4 characters. Furthermore, the attacker knows these 3-4 characters are not English letters.

In this section, we quantify the damage our attack can inflict on password entropy, the typical measure of password strength. The entropy of a password X is defined as $H(X) = -\sum_{i=1}^{n} P(x_i) \cdot \log_2 P(x_i)$, where x_i $(i \in \{1, 2, \cdots, n\})$ is one of n possible values of the password X and $P(x_i)$ represents the probability that $X = x_i$ holds. Considering a keyboard housing N characters, a password with length l selected at random has N^l possible values and $l \cdot \log_2 N$ bits of entropy. Suppose this password has l' letter characters and $l - l'$ non-letter characters. The keyboard with N characters necessarily contains 26 letters and $N - 26$ non-alphabetical characters. Having successfully established a full CSI sample/letter mapping and applying this mapping to the CSI samples comprising the password, its entropy becomes $(l - l') \log_2(N - 26)$ bits. As users typically choose passwords mostly comprised of letters for ease of memory, the proposed attack should reduce the entropy significantly.

In our experiment, we then randomly select 1000 9-character passwords from the Yahoo! Voices dataset. 32 non-alphanumerical characters are allowed in passwords, yielding 42 non-alphabetical characters when factoring in numbers. However, we find that an average 6.38 of 9 characters were letters, meaning their discovery will vastly reduce entropy. Each of these 1000 passwords was added to the end of the text typed by volunteers in the previous experiment, and the resulting CSI sample/letter map was applied to each. We compare the inferred password information to the original password, to identify the correctly recovered characters and calculate the difference in password entropy.

Figure 18 plots the empirical probability mass functions (PMFs) of the password entropies before and after the proposed attack is applied, respectively. A randomly selected 9-character password with the assumed keyboard layout provides 54.8 bits of entropy and requires a maximum of 31.08 quadrillion brute force attempts. After applying our attack, the password entropy can be decreased to within a range of 5.4 to 27.0 bits, such that breaking a 9-character password is reduced to guessing 1-5 non-letter characters. The maximum number of brute-force attack attempts targeting a password with an entropy of 5.4 bits is just 42. In fact, 89.0% of the randomly selected passwords have less than 16.2 bits of entropy after our attack, which means that at most 74,000 brute-force attack attempts are required for the vast majority of these passwords. Evidently, the security of these passwords is decreased by several orders of magnitude courtesy of the proposed attack.

5 DEFENSE DISCUSSION

The proposed attack explores the inter-element relationship matrix to eavesdrop keystrokes. An intuitive defense solution is thus to disrupt the attacker from obtaining the correct relationship. The user may manually encrypt the words to be typed by using some traditional substitution and permutation ciphers. However, this approach is impractical, because it requires the user to calculate and type in the ciphertext, an unintelligible string of random appearance which would take much more time to type and incur numerous input errors. The encryption also brings extreme computational burden to the user.

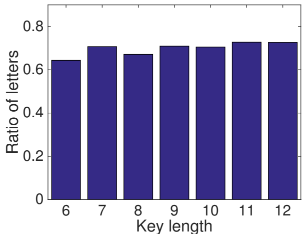


Figure 17: Ratio of letters vs. key length.

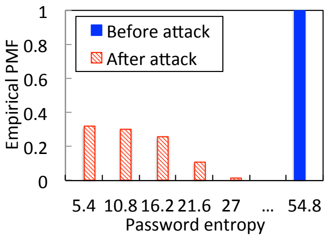


Figure 18: The PMFs of password entropies.

In a more expedient fashion, the user may disrupt the inter-element relationship among letters by randomly inserting a large number of uncommonly used characters (e.g., \, <, >, and &) while typing. Specifically, if the user inserts uncommon characters before the first word, the matrix of the first observed CSI word group will either match an incorrect word or not match with any word in the dictionary, so the demodulation algorithm will return incorrect or no candidates. In the former case, the attacker can still correctly demodulate the following word if it shares no letters with the previous. If no candidates are returned, the attacker will discard the first observed CSI word group and start the demodulation algorithm at the second observed CSI word group. Clearly in both cases, to confuse subsequent words, the user must continue inserting uncommon characters in each word.

To further mislead the attacker, the user can also construct sequences of uncommon characters with the same inter-element relationship matrices as various words in the dictionary. The user can type several of these "fake words" before inputting the meaningful content, and continue typing fake words periodically. The fake words can not only feed the attacker with wrong mappings but also mislead the attacker with incorrect eavesdropping results. To prevent the fake words from interfering with the meaningful content, the user may employ a computer program that automatically searches for and removes the uncommon characters or fake words from the input text.

Beyond disrupting the inter-element relationship matrix, we can also prevent the attacker from receiving useful CSI. Users can employ a wireless jamming device that constantly transmits noise signals to the wireless channel to interfere with the attacker's transmissions. Thus the attacker will not be able to collect accurate CSI, which is required for all wireless-based keyboard eavesdropping attacks, including the proposed one. However, the user will have to set up an external wireless device and turn it on whenever needing to type. This method has more hardware demands than the previous, but it does not require inserting many additional characters.

6 RELATED WORK

Existing non-invasive attacks to infer keystrokes fall into the following categories:

Vibration based attacks: Typing on a keyboard can cause vibrations on the surface where the keyboard rests, with subtle differences depending on keys typed [21, 26]. The accelerometer of a nearby phone or tablet on the same surface can capture the vibrations. With training, an attacker can establish the relationship between the keystroke and the acceleration disturbance caused by the vibration. In the detection phase, the attacker can then recover the typed content by applying this relationship.

Cai *et al.* discovered typing different keys on a soft keyboard on a smartphone may also cause different vibrations [10]. Hence, an invasive key inference attack can be constructed if the attacker can install a key logging malware on the target phone. Recent research has further discovered keystrokes can be inferred by exploiting the accelerometer of a compromised smart smartwatch that a victim wears on the wrist [20, 35]. Under the assumption that the attacker and the user may cause similar impact on accelerometer measurement, the wrist movement trajectory may be used to reconstruct the finger movement trajectory to determine what keys are pressed. [20] novelly combines both accelerometer data and acoustic emanations acquired with a smartwatch to attack keyboards. These methods using smartwatches [20, 35], however, require some kind of malware to be installed on the wrist-worn device to read the sensor data and report them to the adversary.

Acoustic signal based attacks: It has been observed typing on a keyboard can produce sounds unique to each key. Researchers extract features from these sounds and then train a classifier to reconstruct the keystrokes [7, 12, 37, 42]. Some work also exists which relaxes the requirement for training. For example, [42] uses an statistical unsupervised training method to design a supervised classifier. However, the proposed method is faster than the method in [42] for establishing the mapping between features and keystrokes. The fundamental reason for this advantage is that the proposed method is based on word structures whereas the method in [42] is based on statistics. Specifically, the method in [42] uses the Hidden Markov Model (HMM) for key recognition. HMM requires creating a statistical Markov state transition matrix. Creating an accurate transition matrix requires a large number of samples (features). For example, [42] mentions that the attacker needs to collect 50 features for good performance. This means that the state transition matrix is 50-by-50, with each of these 2500 elements representing a state-to-state transition probability. Such a transition matrix can be estimated with reasonable accuracy only after the number of letters a user types is 2500 or larger. Therefore, as discussed in [42], the HMM method requires collecting 10 minutes worth of keystrokes (around 340 words) for a word recovery rate of 87.6%. This minimized training method may not function for wireless based attacks, as due to the time-varying nature of the wireless channel, a training time of 10 minutes may be too long to generate a useful mapping between observed CSI samples and letters. Unlike [42], frequency analysis, and all other statistical methods, the proposed method explores the self-contained structures of words, which can be observed for each word immediately as it is typed, rather than probabilistic statistics among words, which require many words to establish. Thus, the proposed attack only needs 50 words within 1-2 minutes for a word recovery rate of 94.3%.

An adversary may use a triangulation localization technique to localize the sound source and accordingly infer which keys are typed [19, 41]. This approach, however, requires the adversary to have sophisticated equipment that can precisely measure the sound propagation distance from the key to equipment, and also requires line-of-sight between the keyboard and equipment. Both of these requirements hinder attack plausibility and application. Berger *et al.* infer keystrokes with the observation that similar sounds are highly likely to come from keys positioned close to each other on the keyboard [9]. This technique aims to reconstruct a single long (7-13 characters) word that must appear in the dictionary, whereas the goal of the proposed attack is to reconstruct the entire typed content regardless of whether or not all its constituent words are in the dictionary.

Timing based attacks: Keystroke timing patterns can be another source to infer keystrokes [28, 32, 40]. For example, [32] infers keystroke sequences by using the inter-keystroke timing information collected from the arrival times of the SSH packets. [28] proposes to infer keystrokes by utilizing the keystroke timing information gathered via cache-based load measurements on an idle machine. [40] infers keystroke sequences by utilizing the shared information on a multi-user system. However, except for the ways to obtain the keystroke timing information (e.g., launching a SSH session [32], installing a malicious virtual machine on the target physical machine [28], owning a user account on a multi-core system [40]), these timing-based attacks all require a training process to statistically generate the attack models.

Wireless signal based attacks: There are emerging research efforts performing keystroke eavesdropping attacks using wireless signals due to the ubiquitous deployment of wireless infrastructures, the radio signal nature of invisibility, and the elimination of the line-of-sight requirement. In particular, [11] infers keystrokes by examining the amplitude and phase changes of the wireless signal, and [6, 18] utilize the channel condition extracted from the observed wireless signal to distinguish keystrokes. All these works still require a training process to construct the relationship between the observed signal feature and the typing.

Camera-based attacks: A traditional and intuitive method to infer keystrokes is to use cameras to record the typing process and then identify keystrokes by analyzing the recorded video. Researchers have found that video recording of hand movement [8, 31, 39], tablet backside motion [33], or the shadow around fingertips [38], is also able to aid the keystroke inference. However, when the movement of interest does not happen in the presence of a camera, keystroke activities cannot be detected.

Cryptanalysis based attacks: Cryptanalysis is a technique of discovering secrets. Cryptanalysis attacks can be in the form of known-plaintext or ciphertext-only attacks. If we consider the CSI sample as the ciphertext and the original typed content as the plaintext, the training-based keystroke inference attacks [6, 18] are indeed known-plaintext attacks, because the attacker must know some plaintext (i.e., typed content) and the corresponding ciphertext (i.e., CSI) for training. The proposed method does not require the training data and thus it is a ciphertext-only attack. Existing ciphertext-only attacks that attempt to decode the ciphertext of natural language are largely based on the statistical information about the ciphertext [22, 25]. For example, [25] regards the author of an instant message conversation as the plaintext and applies character frequency analysis to instant messages for authorship identification and validation. [22] recovers the plaintext by using a statistical language model and a dynamic programming algorithm.

Nevertheless, collecting statistical information implies that the attacker needs to acquire a large amount of ciphertext. This may not be suitable for the wireless based keystroke inference, because collecting the wireless statistics does require a long time period of observation. As mentioned earlier, this can prevent the attacker from collecting sufficient reliable statistics for accurate keystroke inference. The proposed method is based on the self-contained feature of words instead and thus does not require the long-time observation about wireless statistics.

7 CONCLUSION

We identify a new type of keystroke eavesdropping attack. Compared with all previously discovered attacks, the attack reported in this paper can bypass (1) the requirement of the training phase, which is impractical for most attack scenarios, (2) the requirement to deceive the user or bypass the user's anti-virus and firewall software to install malware on the target device, and (3) the requirement of line-of-sight between the attacker's device and the keyboard. This attack is constructed based on the CSI extracted from the wireless signal. An essential component to this attack is a joint demodulation algorithm, which we create to establish the mapping between each letter and the corresponding CSI sample without training. We implement this attack on USRP X300 platform running GNURadio, and conduct experiments to validate this attack. The experiment results show that the word recovery ratio of the proposed attack is 0.95 for an input of 150 words, whereas that of the traditional frequency analysis method is less than 0.02.

ACKNOWLEDGEMENT

The authors would like to thank the anonymous reviewers for the insightful comments and feedback. The authors at the University of South Florida were supported in part by NSF under grants CNS-1527144, CNS-1553304, and CNS-1717969. The author at the Shanghai Jiaotong University was supported by NSFC under grant 61672350.

REFERENCES

[1] 2017. 2012 Yahoo! Voices hack. https://en.wikipedia.org/wiki/2012_Yahoo!_Voices_hack.

[2] 2017. London attack: Assailant shot dead after 4 killed near Parliament. http://www.cnn.com/2017/03/22/europe/uk-parliament-firearms-incident/index.html.

[3] 2017. Statistical Distributions of English Text. http://www.data-compression.com/english.html.

[4] Fadel Adib, Chen-Yu Hsu, Hongzi Mao, Dina Katabi, and Frédo Durand. 2015. Capturing the Human Figure Through a Wall. *ACM Trans. Graph.* 34, 6, Article 219 (Oct. 2015), 13 pages.

[5] Fadel Adib and Dina Katabi. 2013. See Through Walls with WiFi!. In *Proceedings of the 2013 ACM Conference on SIGCOMM (SIGCOMM '13).* ACM, Hong Kong, China, 75–86.

[6] 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 (MobiCom '15).* ACM, Paris, France, 90–102.

[7] Dmitri Asonov and Rakesh Agrawal. 2004. Keyboard acoustic emanations. In *Proceedings of the IEEE Symposium on Security and Privacy.* IEEE Computer Society, 3–11.

[8] Davide Balzarotti, Marco Cova, and Giovanni Vigna. 2008. ClearShot: Eavesdropping on Keyboard Input from Video. In *Proceedings of the IEEE Symposium on Security and Privacy.* IEEE Computer Society, 170–183.

[9] Yigael Berger, Avishai Wool, and Arie Yeredor. 2006. Dictionary Attacks Using Keyboard Acoustic Emanations. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS '06).* ACM, Alexandria, Virginia, USA, 245–254.

[10] 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, San Francisco, CA.

[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 (MobiSys '15).* ACM, Florence, Italy, 31–44.

[12] 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 (ASIA CCS '17).* ACM, Abu Dhabi, United Arab Emirates, 703–715.

[13] Mark Davies. 2017. Word frequency data from the Corpus of Contemporary American English (COCA). http://www.wordfrequency.info/free.asp.

[14] Matt Ettus. 2005. *USRP user's and developer's guide.* Ettus Research LLC.

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

[16] Jonathan Katz and Yehuda Lindell. 2007. *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series).* Chapman & Hall/CRC.

[17] Swarun Kumar, Ezzeldin Hamed, Dina Katabi, and Li Erran Li. 2014. LTE Radio Analytics Made Easy and Accessible. In *Proceedings of the 2014 ACM Conference on SIGCOMM (SIGCOMM '14).* ACM, Chicago, Illinois, USA, 211–222.

[18] 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 23Nd ACM SIGSAC Conference on Computer and Communications Security (CCS '16).* ACM, Vienna, Austria, 1068–1079.

[19] 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 (MobiCom '15).* ACM, Paris, France, 142–154.

[20] 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 (CCS '15).* ACM, Denver, Colorado, USA, 1273–1285.

[21] 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 (CCS '11).* ACM, Chicago, Illinois, USA, 551–562.

[22] Joshua Mason, Kathryn Watkins, Jason Eisner, and Adam Stubblefield. 2006. A Natural Language Approach to Automated Cryptanalysis of Two-time Pads. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS '06).* ACM, Alexandria, Virginia, USA, 235–244.

[23] IEEE Subcommittee on Subjective Measurements. 1969. IEEE Recommended Practice for Speech Quality Measurements. *IEEE Transactions on Audio and Electroacoustics* 17, 3 (Sep 1969), 227–246.

[24] Alan V. Oppenheim, Alan S. Willsky, and S. Hamid Nawab. 1996. *Signals & Systems (2Nd Ed.).* Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[25] Angela Orebaugh. 2006. An Instant Messaging Intrusion Detection System Framework: Using character frequency analysis for authorship identification and validation. In *Proceedings 40th Annual 2006 International Carnahan Conference on Security Technology.* 160–172.

[26] Emmanuel Owusu, Jun Han, Sauvik Das, Adrian Perrig, and Joy Zhang. 2012. ACCessory: Password Inference Using Accelerometers on Smartphones. In *Proceedings of the Twelfth Workshop on Mobile Computing Systems and Applications (HotMobile '12).* ACM, San Diego, California, Article 9, 6 pages.

[27] Qifan Pu, Sidhant Gupta, Shyamnath Gollakota, and Shwetak Patel. 2013. Whole-home Gesture Recognition Using Wireless Signals. In *Proceedings of the 19th Annual International Conference on Mobile Computing and Networking (MobiCom '13).* ACM, New York, NY, USA, 27–38.

[28] Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. 2009. Hey, You, Get off of My Cloud: Exploring Information Leakage in Third-party Compute Clouds. In *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS '09).* ACM, Chicago, Illinois, USA, 199–212.

[29] Stan Salvador and Philip Chan. 2007. Toward Accurate Dynamic Time Warping in Linear Time and Space. *Intell. Data Anal.* 11, 5 (Oct. 2007), 561–580.

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

[31] 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, Scottsdale, Arizona, USA, 904–917.

[32] 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.

[33] Jingchao Sun, Xiaocong Jin, Yimin Chen, Jinxue Zhang, Rui Zhang, and Yanchao Zhang. 2016. VISIBLE: Video-Assisted Keystroke Inference from Tablet Backside Motion. In *Proceedings of the 23th Annual Network and Distributed System Security*

Conference (NDSS '16). The Internet Society, San Diego, California, USA.

[34] Guanhua Wang, Yongpan Zou, Zimu Zhou, Kaishun Wu, and Lionel M. Ni. 2014. We Can Hear You with Wi-Fi!. In *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking (MobiCom '14)*. ACM, Maui, Hawaii, USA, 593–604.

[35] 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 (MobiCom '15)*. ACM, Paris, France, 155–166.

[36] Jue Wang and Dina Katabi. 2013. Dude, Where's My Card?: RFID Positioning That Works with Multipath and Non-line of Sight. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM (SIGCOMM '13)*. ACM, Hong Kong, China, 51–62.

[37] Junjue Wang, Kaichen Zhao, Xinyu Zhang, and Chunyi Peng. 2014. Ubiquitous Keyboard for Small Mobile Devices: Harnessing Multipath Fading for Fine-grained Keystroke Localization. In *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '14)*. ACM, Bretton Woods, New Hampshire, USA, 14–27.

[38] 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 (CCS '14)*. ACM, Scottsdale, Arizona, USA, 1403–1414.

[39] Qinggang Yue, Zhen Ling, Wei Yu, Benyuan Liu, and Xinwen Fu. 2015. Blind Recognition of Text Input on Mobile Devices via Natural Language Processing. In *Proceedings of the 2015 Workshop on Privacy-Aware Mobile Computing (PAMCO '15)*. ACM, Hangzhou, China, 19–24.

[40] Kehuan Zhang and Xiaofeng Wang. 2009. Peeping Tom in the Neighborhood: Keystroke Eavesdropping on Multi-User Systems. In *Proceedings of the 18th Conference on USENIX Security Symposium (SSYM'09)*. USENIX Association, Montreal, Canada, 17–32.

[41] 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)*. ACM, Scottsdale, Arizona, USA, 453–464.

[42] Li Zhuang, Feng Zhou, and J. D. Tygar. 2005. Keyboard Acoustic Emanations Revisited. In *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS '05)*. ACM, Alexandria, VA, USA, 373–382.