WritingHacker: Audio based Eavesdropping of Handwriting via Mobile Devices

Tuo Yu, Haiming Jin, Klara Nahrstedt
Department of Computer Science
University of Illinois at Urbana-Champaign, IL, USA
{tuoyu2, hjin8, klara}@illinois.edu

ABSTRACT

When filling out privacy-related forms in public places such as hospitals or clinics, people usually are not aware that the sound of their handwriting leaks personal information. In this paper, we explore the possibility of eavesdropping on handwriting via nearby mobile devices based on audio signal processing and machine learning. By presenting a proof-of-concept system, WritingHacker, we show the usage of mobile devices to collect the sound of victims' handwriting, and to extract handwriting-specific features for machine learning based analysis. WritingHacker focuses on the situation where the victim's handwriting follows certain print style. An attacker can keep a mobile device, such as a common smartphone, touching the desk used by the victim to record the audio signals of handwriting. Then the system can provide a word-level estimate for the content of the handwriting. To reduce the impacts of various writing habits and writing locations, the system utilizes the methods of letter clustering and dictionary filtering. Our prototype system's experimental results show that the accuracy of word recognition reaches around 50% - 60% under certain conditions, which reveals the danger of privacy leakage through the sound of handwriting.

ACM Classification Keywords

K.6.5. [Management of Computing and Information Systems]: Security and Protection – Invasive software

Author Keywords

Eavesdropping; Handwriting; Audio Signals.

INTRODUCTION

It has been proven that the sound of typing keyboards leaks information, which implies the possibility of acoustic side channel attacks [1-4]. However, it has not yet been recognized that the sound of handwriting also leaks information. People's writing habits follow some common patterns (e.g., stroke number), which make audio-based handwriting recognition possible. The rapid evolution of mobile terminals compounds the danger of audio-based eavesdropping of handwriting in public environments. For instance, when a person is filling out privacy-related forms on a desk in medical environments such as hospitals or clinics, an attacker can record the sound of handwriting with a smartphone, and recover the content of handwriting. To expose this danger, in this paper we investigate the possibility of eavesdropping on handwriting via nearby mobile devices based on audio signal processing and machine learning.

Although handwriting on desks has been considered as a new text entry method for small mobile devices, this technique is not applicable for the eavesdropping attack. The methods in [5] and [6] enable users to write on surfaces by using smartphones as pens or by wearing smartwatches on wrists. However, these methods assume that users have direct contact with some mobile devices, and thus cannot be used for attack because attackers usually have no access to victims' devices. In [7], the authors leverage the embedded microphone in commercial smartphones to record the sound of handwriting on surfaces and recognize the input text. However, once the system has been trained, its performance highly depends on the location of handwriting, which makes eavesdropping attack impossible. In this paper, we investigate the capacity of audio-based side channel attack under the condition that the attacker's mobile device has no direct contact with the victim, and the performance of the attack does not highly depend on the location of handwriting.

In this paper, we present WritingHacker, an audio-based eavesdropping proof-of-concept system which eavesdrops on handwriting via mobile devices. Our method is based on the basic assumption that the handwriting of victims is print-style, which means that we currently ignore the impact of joined-up writing. We consider this assumption reasonable because most of privacy-related forms require print-style writing.

The main idea of our attack method is that, by keeping a mobile device such as a smartphone touching the desk which is used by a victim, an attacker can record the sound of the victim's handwriting. After regaining the device, the attacker can reconstruct the words that the victim has written by extracting features from the sound sequence and recognizing the unique patterns of each letter.

The main challenge is that, in the scenarios of attack, labeled training data from victims are always unavailable. Moreover, the performance of existing audio-based handwriting recognition methods highly depends on the location of handwriting, which can hardly be controlled by the attacker. We show that, using WritingHacker, an attacker can leverage the common patterns of people's print-style handwriting, which are not af-

UbiComp '16, September 12-16, 2016, Heidelberg, Germany
 ISBN 978-1-4503-4461-6/16/09…$15.00
DOI: http://dx.doi.org/10.1145/2971648.2971681

fected by the aforementioned factors. For instance, different letters sometimes have different stroke numbers. Letter 'A' usually has three strokes, while letter 'C' has only one stroke, which makes it easy to distinguish between these two letters even if victims' training data are unavailable. Meanwhile, the stroke number for each letter is not impacted by handwriting locations. In WritingHacker, we classify all the letters into different clusters according to their stroke numbers, and implement a Support Vector Machine (SVM) for each cluster to distinguish the in-cluster letters. To train the SVMs, the attacker can collect training data from other people instead of the victims. Moreover, to provide the linguistic word-level correction and narrow down the word search range, we apply a dictionary filter, which is also based on letter clustering.

The main contributions of our paper are:

- To the best of our knowledge, we are the first to investigate the possibility of eavesdropping on handwriting based on audio signals.
- To solve the lack of training data from victims and reduce the impacts of diverse writing locations, we propose to use the stroke number as a common handwriting pattern, and apply the method of letter clustering.
- We implement a prototype system on Android-laptop platform and conduct evaluations. Experiments confirm that privacy leakage through eavesdropping on handwriting is highly probable under certain conditions such as print-style writing and low near-field noise.

WritingHacker is still a proof-of-concept system based on our basic assumption. However, it sheds light on a new threat to people's privacy caused by widespread mobile devices.

We organize the rest of our paper as follows. After introducing related work, we present an overview of the system architecture and the attack method. Then we successively introduce the methods and algorithms used by the system components. We include the implementation details and performance evaluation of WritingHacker. We discuss the test results and then conclude our paper.

RELATED WORK

Leakage of privacy via text entry methods: The topic about the leakage of privacy via text entry methods has become popular in recent years. Many existing eavesdropping attacks of text entries focus on keyboards, which are the most common input tools for electronic equipments. Besides the approaches based on video of typing sessions [8] or timing information of key presses [9], the other methods are usually based on audio or acceleration signal processing.

By extending the technique of human voice recognition, attackers can recognize the keys that a victim has pressed by eavesdropping on keystrokes. Asonov *et al.* firstly showed that keyboards are vulnerable to attacks based on differentiating the sound emanated by different keys [1]. In [2], Zhuang *et al.* presented an audio-based attack method which combines standard machine learning and speech recognition techniques. Berger *et al.* in [3] presented a dictionary attack which can reconstruct a single typed word from audio signals with a dictionary of words. For the acceleration-based eavesdropping, Marquardt *et al.* revealed that an application with access to accelerometer readings on a smartphone can recover text entered on a nearby keyboard [10]. However, the audio/acceleration signals of handwriting are much more diverse compared with those of keystrokes. For instance, the keyboard audio signal for each keystroke usually contains a press region and a release region [4]. Thus it is easier to select appropriate features for machine learning algorithms to compare different keystrokes. For the case of handwriting, the acoustic signals of different strokes vary with people. Thus the previous approaches cannot be applied to our scenario.

The universalization of smart mobile devices is also increasing the risk of privacy leakage. For example, Wang *et al.* has shown the danger that the motion sensors in smartwatches can leak information about what the user is typing [11]. However, this approach requires the direct contact between victims and mobile devices, which is not practical in the scenario of eavesdropping attack.

In this paper, WritingHacker eavesdrops on handwriting based on audio signal processing, because compared with acceleration signals, audio signals are less influenced by the propagation distance in solid media such as desks.

Handwriting Recognition: The techniques for automatic handwriting recognition (HWR) can be classified as online and offline case [12]. Offline recognition usually processes images of handwriting only, whereas online recognition can further utilize rich sensor data such as stylus positions and temporal information during the writing process [12, 13].

In this paper we focus on audio signal based attack, and thus do not apply the image-based offline recognition techniques such as Optical Character Recognition (OCR) [14]. For the online case, multiple sensors can be used to improve recognition accuracy. In [15], Nakai *et al.* used pen pressure as a feature based on the observation that the pressure represents pen ups and downs as well as the temporal pattern of handwriting in a continuous manner. In [16-18], motion sensors embedded in pens or Micro Electro Mechanical Systems were also utilized to sense the motion produced by written characters. Different from these approaches, in our attack method, we utilize the widely installed microphones in mobile devices to recognize handwriting.

Handwriting on desks as an input method: Since handwriting is a convenient input method other than keystroking, there have been some works focusing on enabling handwriting on desks to be a new text entry method for small mobile devices. The system named GyroPen in [5] enables users to use their smartphones as pens to write on desks by using embedded gyroscopes and accelerometers. The system in [6] enables a user to write on surfaces with fingers. A smartwatch worn on the user's wrist records the accelerometer signals of user's hand, which can be utilized to recognize the user's handwriting with the algorithm of gesture recognition. However, these approaches are not suitable for attack because they require users to have direct contact with mobile devices.

Using the system from [7], after executing a training phase, a user can write with a finger on surfaces, and the embedded mi-

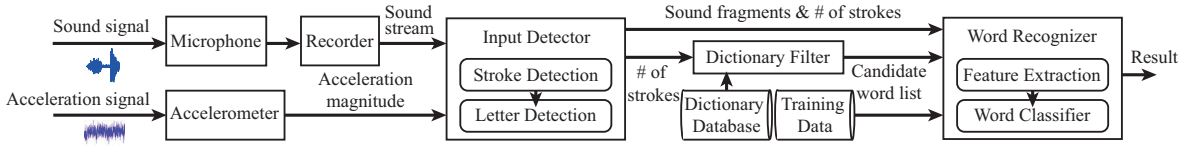


Figure 1. System Architecture of WritingHacker.

crophone in a smartphone records the sound of handwriting. However, once the location of the smartphone is changed, the training phase must be repeated. Since the training data, collected from victims, are usually unavailable in practice, this system cannot be used for eavesdropping attack.

In this paper, we focus on a more practical case where the attacker' mobile device has no direct contact with the victim, and the attacker has no control on the distance or relative angle between the device and the victim's handwriting.

SYSTEM AND ATTACK OVERVIEW

In this section we have an overview of the system architecture and the workflow of our attack method. Based on the observation that people often write separate words instead of sentences when filling out forms, currently we mainly focus on the word-level recognition.

System Architecture

As shown in Figure 1, our system mainly consists of the following components.

Recorder and accelerometer: The recorder records the sound signal of handwriting with the embedded microphone and sends the recorded signal to the input detector in stream. The output of the accelerometer is used to combat near-field noise.

Input detector: This component receives the sound stream and extracts the sound fragment for each letter. The subcomponent of stroke detection detects each stroke of handwriting, and the letter detection subcomponent detects and extracts sound fragments, each of which contains the audio signal for a single letter. The sound fragments together with the number of strokes for each letter are sent to the word recognizer, while the number of strokes is also passed to the dictionary filter to narrow word search range.

Dictionary filter: The input to this component includes the number of strokes for each letter in each word. The filter reads the dictionary database and provides a list of candidate words.

Word recognizer: This component calculates the similarity between the recorded sound fragments and each word in the candidate list. The best match is the recognition result. The feature extraction subcomponent extracts the features for each letter, and the word classifier recognizes each word using machine learning algorithms.

Attack Method

Our attack method consists of the following phases.

Phase 1 (Eavesdropping): To commit the handwriting eavesdropping without being noticed, an attacker can keep the mobile device touching the desk where a victim is writing on a piece of paper, which does not violate non-direct-contact. The recorder records the sound of the victim's handwriting via the microphone. The component of input detector separates the sound sequence and stores the sound fragment of each letter. Then the attacker gets the device back.

Phase 2 (Training Data Collection): The attacker needs to hire some other people instead of the victim to collect labeled training data. The training data should include the sound fragment and the number of strokes for each character (e.g. 'A'-'Z') from different people's handwriting. We assume that it is easy for the attacker to collect enough training data. The data are used to train the word classifier. Note that Phase 2 can be executed ahead of Phase 1. However, to achieve better performance, the attacker can execute Phase 1 first, and during Phase 2 the attacker can use the same type of pen and material of desk as those used by the victim in Phase 1.

Phase 3 (Recognition): The stored sound fragments are processed by dictionary filter and word recognizer, and then the attacker gets the estimate for the victim's handwriting. Note that phase 2 and phase 3 are offline phases.

In this paper, we take the capital letters 'A'-'Z' as examples to show how WritingHacker works. However, our technique can be easily applied to other characters.

Next we introduce the algorithms of the three main components (input detector, dictionary filter and word recognizer) in detail respectively.

INPUT DETECTION

The method used by the input detector is based on signal processing in time domain, which consists of two steps, namely stroke detection and letter detection. First, we introduce the definition of a stroke and the concept of letter clusters.

Definition of Stroke and Letter Cluster

Stroke: In our paper, we define a stroke as a mark made by a writing instrument *with a single touch to a surface*. This means that we consider the letter 'B' as a two-stroke letter, even if it has multiple turning points.

Letter Clusters: Based on our basic assumption about print-style writing, we divide the capital letters into three *clusters* according to the number of strokes. The three clusters are:

$$C_1 = \{\text{'C'}, \text{'G'}, \text{'L'}, \text{'M'}, \text{'O'}, \text{'S'}, \text{'U'}, \text{'V'}, \text{'W'}, \text{'Z'}\},$$

$$C_2 = \{\text{'B'}, \text{'D'}, \text{'J'}, \text{'K'}, \text{'P'}, \text{'Q'}, \text{'R'}, \text{'T'}, \text{'X'}\},$$

$$C_3 = \{\text{'A'}, \text{'E'}, \text{'F'}, \text{'H'}, \text{'I'}, \text{'N'}, \text{'Y'}\}.$$

Thus each letter in cluster C_u has u strokes ($u = 1, 2, 3$). The concept of clusters helps to narrow the word search range in the dictionary filter and reduce the number of classes in the word classifier.

We admit that due to people's different writing habits, for some people it is possible that some letters should be classified into different clusters. For instance, some people write letter 'J' with one stroke instead of two strokes. Sometimes the stroke order of the same letter would also vary with people. In this case the attacker may need to observe the victim's writing habit to guarantee the correct clustering. However, based on our basic assumption that victims' handwriting is print-style, we observe that people's print-style writing has limited variety, and the stroke order of the same letter written by a victim usually does not change. For instance, while there are two ways to write letter 'J', there is usually only one way to write letter 'X'. Thus the attacker can adjust the clusters and their corresponding training data by traversing different print writing styles to find the most reasonable guess for the attack scenario. Therefore we leave it to our future work to rule out the impact of the stroke number's diversity.

Stroke Detection

We firstly detect each stroke in the continuous sound stream. A naive method is setting a constant threshold and once the sound signal magnitude or instantaneous power exceeds the threshold, a stroke is detected. However, since noise levels are usually different in different environments, and the average noise power varies over time, it is hard to assign such a threshold. We use a method similar to the Constant False Alarm Rate (CFAR) algorithm [19] to solve this problem.

Let us assume that the noise power follows Gaussian distribution. Its average power and standard deviation at time t are denoted by $\mu(t)$ and $\sigma(t)$. We denote the amplitude of the received audio signal by a discrete time series $x(t)$, and use a sliding window of size W to calculate the average noise power at time t [20]:

$$\mu(t) = \frac{1}{W}A(t) + (1 - \frac{1}{W})\mu(t-1), \quad (1)$$

where

$$A(t) = \frac{1}{W}\sum_{k=t-W+1}^{t} |x(k)|^2. \quad (2)$$

The standard deviation at time t is calculated in a similar way:

$$\sigma(t) = \frac{1}{W}B(t) + (1 - \frac{1}{W})\sigma(t-1), \quad (3)$$

where

$$B(t) = \sqrt{\frac{1}{W}\sum_{k=t-W+1}^{t} (|x(k)|^2 - A(t))^2}. \quad (4)$$

A *potential* start point of a stroke is detected if

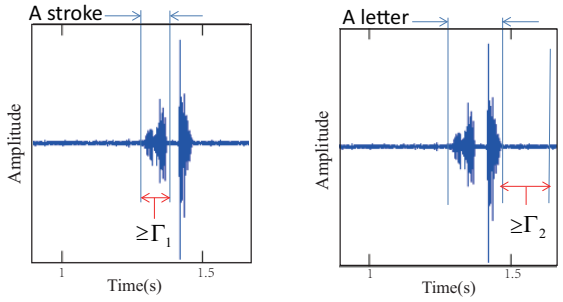
$$|x(t)|^2 > \mu(t) + \gamma_1\sigma(t), \quad (5)$$

and a *potential* end point of a stroke is detected if

$$|x(t)|^2 < \gamma_2\bar{\mu}, \quad (6)$$

where γ_1 and γ_2 are two constant parameters that are independent of noise levels. $\bar{\mu}$ is the average noise power when there is no input. The initial value of $\bar{\mu}$ is measured during the very beginning of recording before the first stroke appears. Then we update it by measuring the average noise power between each two adjacent detected letters. The reason why we do not use $|x(t)|^2 > \gamma_2\bar{\mu}$ for start point detection is that it is impossible to measure $\bar{\mu}$ if the start point has not been detected.

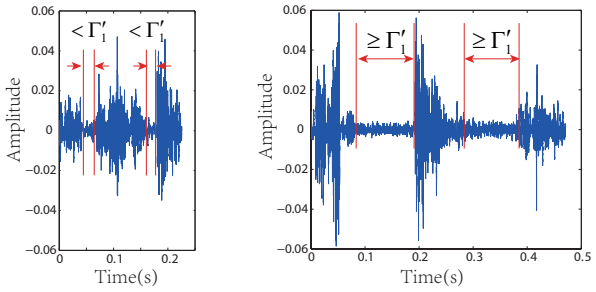
A stroke is detected if the time interval between the potential start and end point is longer than a constant threshold Γ_1, as shown in Figure 2(a). In this way the impact of burst noise can be avoided. If the time interval between two detected strokes is shorter than Γ_1', we combine these two strokes and consider them as a single stroke.



(a) Stroke detection. (b) Letter detection.

Figure 2. Example for input detection.

Impact of multiple turning points: Sometimes a stroke with multiple turning points would show similar features as a multi-stroke letter. One example is shown in Figure 3(a). According to our definition of a stroke, the letter 'M' contains one stroke. Due to the turning points included in the stroke, it looks like that the single fragment contains three strokes. However, by comparing its signal with that of the real three-stroke letter 'A' in Figure 3(b), it is easy to find that the time interval between each two adjacent "strokes" in 'M' is much shorter than that in 'A'. Thus the threshold Γ_1' can help to rule out the impact of false strokes. In this case, since both the two time intervals in 'M' are shorter than Γ_1', the three "strokes" are combined as a single stroke.



(a) Letter 'M' (b) Letter 'A'

Figure 3. Comparison between a multi-turning-point stroke and a multi-stroke letter.

Letter Detection

After each stroke has been detected, this subcomponent extracts the sound fragment for each letter. The start point of a letter is declared if it is the first start point of a stroke after the end point of the previous letter. The end point of a letter is declared if there is no stroke detected in constant duration

Γ_2 after the end point of a stroke. Then the end point of this stroke is the end point of the letter. One example is shown in Figure 2(b).

Then the measured sound signal between the start and end point of the letter is output and stored as a single fragment. The fragments detected will be further processed by the feature extraction subcomponent. We use s_k to denote the number of strokes for the kth letter in a word with N_{letter} letters. For example, word "VITAMIN" has $N_{letter} = 7$, and $s_1 = 1$ for letter 'V'$\in C_1$. Then $\{s_k | k \in [1, N_{letter}]\}$ is passed to the dictionary filter and the word classifier, which will help to select proper word clusters and narrow word search range.

Usage of Accelerometer

To combat the impact of near-field noise, we use a technique similar to [20]. When reading the sound sequence from the recorder, the input detector also reads the measured acceleration magnitude from the embedded accelerometer. If a possible letter is detected but the average acceleration in a window W_a around the start point of the letter is lower than a threshold $\Gamma_a \bar{a}$, the letter is ignored. Γ_a is a constant value and $\bar{a}$ is the average acceleration when there is no letter input. Since near-field burst noise such as human voice does not impact the output of the accelerometer, the input detector will not record the sound caused by the noise. In this way, the impact of near-field burst noise is partly avoided. However, frequent burst noise would still lead to the missing of letters and how to rule out its impact is still an open question. We leave this problem for our future work.

DICTIONARY FILTERING

One of the key design points of WritingHacker is reducing the size of candidate word set for the word classifier. We use a dictionary database to store the commonly used words and a filter to extract the validated words whose stroke numbers match with the input set of stroke numbers $\{s_k\}$. To achieve the best performance, the dictionary database should be scenario-specialized.

Scenario-Specialized Dictionary Database

It has been shown that the frequency of commonly used words follows Zipf's law [21], which implies that we can narrow our word search range to the most commonly used words. We can further specialize our dictionary database according to the attack scenarios. For illustration, we obtain the top 5,000 most common words in Intensive Care Units (ICUs) by analyzing the statistical data from the MIMIC II Clinical Database [22], which contains notes and reports from approximately 32,000 ICU patient admissions. As shown in Figure 4, the frequency of words still follows Zipf's law even under such a specialized scenario. We observe that the ranks of words vary with scenarios. For instance, the frequency of the word "vitamin" used in ICUs is much higher than that in daily life, which confirms the need of scenario-specialized databases.

In WritingHacker, once the attack scenario (e.g., in hospital) is determined, we use the most commonly used words in that scenario as the dictionary database. Note that because of the

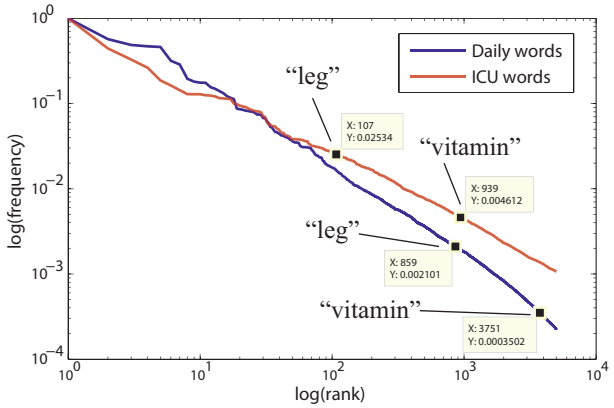


Figure 4. Comparison between top-5000 daily words and ICU words. Normalization has been applied for comparison.

nature of our system design, the system performance does not highly depend on the attack scenario. Instead, the accuracy of word recognizer is related to whether the dictionary database is constructed properly. If the database is not specialized according to the attack scenario or its size is too small, it is more likely that the words, written by the victim, are not covered by the database. However, if the size of the database is too large, the rarely used words would extend the word search range and thus impact recognition accuracy. We will evaluate the impact of the database size during our experiments.

Generating Candidate Word List

The component of dictionary filter has access to the scenario-specialized dictionary database and extracts candidate words according to the number of strokes for each letter in the word.

For each set of stroke numbers $\{s_k\}$ received from the input detector, the dictionary filter accesses the database and finds all the words that match with $\{s_k\}$, i.e., the number of letters is equal to the size of $\{s_k\}$, and the stroke number of the kth letter is equal to s_k. All the eligible words consist the candidate word list. For instance, for the received $\{s_1 = 3, s_2 = 2, s_3 = 2, s_4 = 1, s_5 = 3\}$, the candidate words extracted from the top-5000 daily words [23] are {"APPLE", "APPLY"}, one of which is the most possible word that the victim has written. The candidate word list is then passed to the word recognizer.

To speed up the process of stroke number matching, the words in the dictionary database can be pre-sorted according to the number of letters and then the stroke number for each letter.

WORD RECOGNITION

The component of word recognizer firstly extracts the features of the sound fragment for each letter, and then recognizes the whole word according to the features, the trained SVMs, and the candidate word list.

Features Extraction

For each sound fragment, we extract features mainly from frequency domain. Even if the features in frequency domain are usually impacted by the locations of handwriting, our experiments will prove that they still benefit the letter recognition. For each sound fragment, we calculate the FFT coefficients

$F(\omega)$, $\omega \in [0, \omega_0]$, and extract the following features, which are often used in audio classification [24]:

- *Brightness:* the centroid of frequency spectrum, denoted as

$$\omega_C = \int_0^{\omega_0} \omega |F(\omega)|^2 d\omega / \int_0^{\omega_0} |F(\omega)|^2 d\omega. \quad (7)$$

- *Bandwidth:*

$$\omega_B = \sqrt{\int_0^{\omega_0} (\omega - \omega_C)^2 |F(\omega)|^2 d\omega / \int_0^{\omega_0} |F(\omega)|^2 d\omega}. \quad (8)$$

- *Pitch Frequency:* $\omega_P = \arg \max_{\omega \in [0,\omega_0]} F(\omega)$.
- *Number of Large Frequency Peaks:* the number of the frequency peaks whose values are larger than $\gamma_3 \max_\omega F(\omega)$, denoted by n_P. γ_3 is a constant threshold and $0 < \gamma_3 < 1$. We set $\gamma_3 = 0.65$.
- *Mel Frequency Cepstrum Coefficients (MFCCs):* we firstly divide each sound fragment into frames with length W_f. Two adjacent frames have overlap W_o. The number of cepstrum coefficients for each frame is denoted by N_c. Thus, for each frame $i = 1, 2, ..., N_f$, we get a vector of MFCCs: $[m_1^i, ..., m_{N_c}^i]$. Then we calculate

$$m_j = \sum_{i=1}^{N_f} m_j^i, \ j = 1, ..., N_c. \quad (9)$$

  In our prototype, the sampling rate of audio signal is 22050 Hz, and we set $W_f = 128$, $W_o = 48$ and $N_c = 12$.

We denote the feature vector F for each sound fragment by:

$$F = [\omega_C, \omega_B, \omega_P, n_P, m_1, ..., m_{N_c}]^T. \quad (10)$$

Therefore, for a word with N_{letter} letters, the kth letter g_k is labeled by $\{s_k, F_k\}$. In our paper, we set the stroke number $s_k \in \{1, 2, 3\}$. Recall that s_k has been determined by the input detector.

Word Classifier

The function of word classifier consists of *letter scoring* and *word selection*. The former step evaluates the possibility for each detected letter to be a given letter. The latter step selects the most possible word according to the candidate word list and the results of letter scoring.

Letter Scoring

The algorithm we use to score the detected letters is based on the multi-class SVM. Since all letters have been divided into three clusters, we design specialized SVMs for each cluster. In detail, for the cluster C_u ($u = 1, 2, 3$), we use a set of SVMs V_{ij}^u to distinguish the difference between letter i and j, where $i, j \in C_u$ and $i < j$ in alphabetical order.

For each letter $l \in L = \{\text{'A'}, ..., \text{'Z'}\}$, we collect a set of training sound fragments. Each set should contain the data collected from multiple people's handwriting, as well as the actual written letters. For each sound fragment, we calculate its audio features as denoted by Equation (10). Thus, we get the feature sets T_l, $l \in L$. Then we train each V_{ij}^u ($u = 1, 2, 3$) with the features T_i and T_j. Note that in this case $i, j \in C_u \cap L$.

The steps of letter scoring are shown in Algorithm 1 (line 2-11). According to the s_k of each input letter, the feature set F_k is input to each $V_{ij}^{s_k}$, where $i, j \in C_{s_k}$ and $i < j$ (line 4). We use $p_k(l)$ to indicate the probability for g_k to be a given letter l in L. According to the classification result of $V_{ij}^{s_k}$, $p_k(i)$ or $p_k(j)$ is increased (line 5-8). We eliminate the impact of different cluster sizes in line 9-10. Note that if $u \neq s_k$, $\tilde{p}_k(l) = 0$ for all $l \in C_u$. In line 11, we apply the stroke duration based correction to further improve accuracy.

Algorithm 1: Word Classification

Input: $\{s_k, F_k\}$, $k \in [1, N_{letter}]$;
V_{ij}^u, $i, j \in C_u$, $i < j$, $u = 1, 2, 3$; C_u, $u = 1, 2, 3$;
$\widetilde{W}_m = \{\tilde{g}_k^m \in L | k \in [1, N_{letter}]\}$, $m \in [1, M]$;
Output: $\widehat{W}$;

```
// Initialization
```
1 $p_k(l) = 0$, $l \in L$, $k \in [1, N_{letter}]$; $\widehat{W} = \emptyset$;
```
// Letter Scoring
```
2 **foreach** $k \in [1, N_{letter}]$ **do**
3 **foreach** $i, j \in C_{s_k}$, $i < j$ **do**
4 Test F_k with $V_{ij}^{s_k}$;
5 **if** F_k *is classified as* i **then**
6 $p_k(i)$++;
7 **else**
8 $p_k(j)$++;
9 **foreach** $k \in [1, N_{letter}]$, $l \in L$ **do**
10 $\tilde{p}_k(l) = p_k(l) / (\text{size}(C_{s_k}) - 1)$;
11 Execute stroke duration based correction;
```
// Word Selection
```
12 **foreach** $m \in [1, M]$ **do**
13 $P_m = \sum_{k=1}^{N_{letter}} \tilde{p}_k(\tilde{g}_k^m)$;
14 $\widehat{W} = \widetilde{W}_{\hat{m}}$, where $\hat{m} = \arg \max_{m \in [1,M]} P_m$;
15 If ties exist, break ties;
16 **return** $\widehat{W}$;

Stroke Duration based Correction: Even though writing habits vary among people, we find that some common patterns are still followed. For instance, the time durations for some specific strokes are differentiable: the second stroke of 'Q' is much shorter than the other strokes in the two-stroke letters. It is possible to recognize this pattern without any machine learning algorithm. In our implementation we set a constant threshold r_Q. If the ratio between the time lengths of the first and the second stroke in a two-stroke letter g_k is larger than r_Q, we increase the $\tilde{p}_k(\text{'Q'})$ by p_Δ. During the implementation we set $r_Q = 3.3$ and $p_\Delta = 0.5$.

Word Selection

Because $\tilde{p}_k(l)$ indicates the degree to which the input g_k matches letter l, we can select the best estimate from the candidate word list.

The steps of word selection are also shown in Algorithm 1 (line 12-16). We denote the candidate word list by $\{\widetilde{W}_m | m \in$

$[1,M]\}$, where M is the number of the candidate words. Recall that the length of each candidate word is equal to that of the input word. Then for each $\widetilde{W}_m$, we evaluate the probability for it to be the input word by calculating P_m (line 12-13), and select the one with the highest P_m as our final output (line 14).

To break possible ties (line 15), for each candidate word in a tie, we calculate the number of letters that are the best match with the input letters, i.e.,

$$N_m = \sum_{k=1}^{N_{letter}} \mathbf{1}[\tilde{g}_k^m = \arg\max_{l \in L} \tilde{p}_k(l)]. \quad (11)$$

Then we output the word with the largest N_m in the tie. If the tie still exists, we output the candidate word with the highest frequency of use according to the statistical data in the database.

IMPLEMENTATION DETAILS

In this section, we describe the implementation details of WritingHacker. Except for the recorder, all components can be implemented either on smartphones or on other mobile computers controlled by attackers. In our prototype, we implement the input detector on a smartphone, and implement the dictionary filter and the word recognizer on a laptop.

Implementation of Recorder and Input Detector

We implement the recorder and the input detector on a commercial Android smartphone. Its model is Nexus S with Android Jelly Bean (4.1.2) as its operating system. The smartphone is equipped with 1 GHz CPU and 512 MB RAM.

Even though sometimes there are two built-in microphones on a smartphone, we use only one of them to ensure that WritingHacker is implementable on most mobile devices such as smartwatches. The sampling rate of microphone is 22050 Hz, and the format of audio data is PCM 16 bit per sample.

The working mode of the accelerometer is *SENSOR_DELAY_NORMAL*. It has three outputs for three directions: $a_x(t)$, $a_y(t)$ and $a_z(t)$. We calculate

$$a(t) = \sqrt{a_x(t)^2 + a_y(t)^2 + a_z(t)^2} \quad (12)$$

as the amplitude of acceleration, and set $W_a = 1000$, $\Gamma_a = 2.4$.

The input detector reads the output of recorder in real time and buffers the sound sequence. We set $W = 200$, $\gamma_1 = 16$, $\gamma_2 = 1$, $\Gamma_1 = 45$ ms, $\Gamma_1^{'} = 68$ ms, and $\Gamma_2 = 227$ ms.

Implementation of Dictionary Filter and Word Recognizer

We have implemented the dictionary filter and the word recognizer on a laptop (Thinkpad T430) using Java and Matlab script. The output of input detector is transmitted to the laptop once the attacker has regained the smartphone.

For the dictionary filter, we implement three separate dictionary databases according to different attack scenarios. For the scenario of daily life, we use the top-2000 and top-5000 commonly used word lists available in [23]. For clinic scenario, we extract all the words in the MIMIC II Clinical Database [22], and then use the top-9000 words as our database. For the word recognizer, we use the API of Spider [25] on Matlab to realize SVM. The features of sound fragments are calculated by Java.

MEASUREMENT AND EVALUATION

In this section, we evaluate the performance of our prototype system by conducting experiments on our testbed in a controlled environment. We set our test environment in one laboratory, where the near-field noise is low. The range of characters is 'A'-'Z'. We collect the training data set for the word classifier from two volunteers #1,2. Each volunteer writes each character in print style for 10 times. Thus for each character the size of training data is 20. Since the attacker can determine the type of the pen used by the victim, we omit the evaluation about the impact of different types of pens, and use the same type of ballpoint pens during the tests. The handwriting is on a paper with standard A4 size (210 mm $\times$297 mm) on a wooden desk. We place the smartphone on the same desk, and the distance between the smartphone and the handwriting is around $20-30$ cm. We train SVMs in the character spaces of C_1 - C_3, respectively. The volunteers (#1-9) in the following tests are required to follow the same letter clusters (C_1 - C_3). However, before the test they were not aware that their data are used for testing an eavesdropping system.

Performance of Input Detector

In this subsection we evaluate the performance of input detector. Three volunteers #3-5 write characters 'A'-'Z' repeatedly on the paper. Each volunteer repeats writing each letter by 40 times. The other settings are the same as those in the training phase. To compare the intercepted sound fragments with the original sound sequences, the recorder is configured to store the original sound sequence in the smartphone's storage. The total number of the written characters is 3120. According to our results of the comparison between the intercepted sound fragments and the actual sound sequences, the accuracy of letter detection is 92.88%. The false positive rate is 4.33%, while the false negative rate is 2.79%.

The main cause of false positive is the near-field burst noise during the experiment. Even though we have used the accelerometer to reduce its impact, the false positive still exists if the near-field burst noise happens frequently. We find that if the user is knocking on the desk while the near-field noise happens at the same time, the input detector would consider it as the start point of a letter because the output of the accelerometer exceeds the threshold $\Gamma_a \bar{a}$ and the duration of noise exceeds Γ_1.

The cause of false negative is that sometimes the sound of the volunteers' handwriting is so weak that the input detector does not detect the start point. One simple solution is making γ_1 smaller. However, this would lead to a higher false positive rate. The parameter γ_1 actually controls the trade-off between the false positive rate and the false negative rate.

Performance of Normal SVM

Before we evaluate the accuracy of word recognizer, we first prove that applying the normal SVM technique to our attack

scenario directly can only achieve low accuracy. In this subsection we evaluate the case where only normal SVM is applied to our system. To rule out the benefits of letter clustering and dictionary filtering, all the letters are considered as in the same cluster, and we evaluate the accuracy of letter recognition only. Besides the features in Equation (10), the feature extraction also takes the number of strokes as a normal feature for a single letter.

We firstly consider the common case where the attacker has no access to the labeled training data from the victim. After the system has been trained with the training data set from the two volunteers #1,2, we use the smartphone to record the handwriting of three other volunteers #3-5. During the test the three volunteers use a different wooden desk from that used during the training stage. The range of characters is 'A'-'Z'. The volunteers write each character in print style 20 times. The other settings are the same as those in the training stage.

To show the impacts of the unavailability of victims' training data and the changing writing locations, we also consider the case where the labeled training data from the victim are available. In the training stage, we collect the training data from the three volunteers #3-5 individually. Each letter in 'A'-'Z' is repeated 20 times. During the training, the locations of smartphone and handwriting are fixed. In the test stage we design two test cases. In both cases, each volunteer writes letter 'A'-'Z' 20 times. However, in *Case 1*, the handwriting locations are randomly distributed within 30 cm from the smartphone on a different desk. In *Case 2*, the handwriting is at exactly the same location on the same desk as that in the training stage.

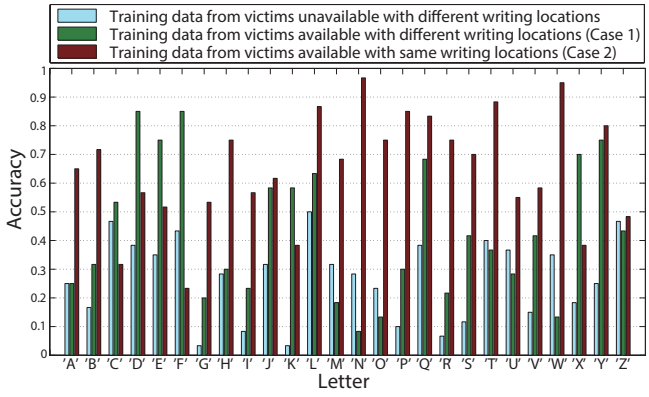


Figure 5. Accuracy of recognition for letters 'A'-'Z' using SVM only.

We calculate the average accuracy of letter recognition for all cases. The results are shown in Figure 5. When the training data from the victims are unavailable, the average accuracy is 26.79%. The average accuracies for Case 1 and Case 2 are 43.01% and 64.94%, respectively.

In Case 2, since the writing locations are fixed on the same desk, the multi-path effect in handwriting sounds' propagation is eliminated, which leads to the relatively high accuracy. In Case 1, even if the training data from the victims are still available, the change of writing surfaces and locations causes the mismatch between the training data and the test data. Although we have avoided using the amplitude-related features during feature extraction, the multi-path effect impacts the other features such as bandwidth and pitch frequency. Since attackers can hardly detect the accurate writing locations of victims, it is hard to rule out this negative effect.

It is obvious that when the labeled data from victims are not available, the average accuracy drops significantly. Since the training and test data are collected from different people, victims' various writing habits and changing writing locations lead to this drop. Even if the word-level semantic analysis can be applied to exclude some obviously-wrong letter recognition results, it is still hard to hit the real written word because of this low letter recognition accuracy. Thus we prove that using normal SVM technique only cannot achieve our design goals and applying the stroke number based letter clustering and dictionary filtering is necessary.

Performance of Word Recognizer

In this subsection we evaluate the accuracy of the final output of WritingHacker. We continue to use the training data collected from two volunteers #1,2. For the dictionary database, we use three different sets of words: top-2000 commonly used daily words, top-5000 commonly used daily words, and top-9000 commonly used medicine-related words from the MIMIC II Clinic database.

We repeat the test for each individual database. In each test, each of our seven volunteers #3-9 writes 1000 words selected from the database. For each word in the database, we assign its frequency of appearance as its weight. To select a word for test, we randomly select a word from the weighted database, i.e, the probability for a word to be selected is equal to its appearance frequency divided by the sum of all the frequency values in the database. Following this rule, during each test we collect 7000 samples. The results are shown in Figure 6.

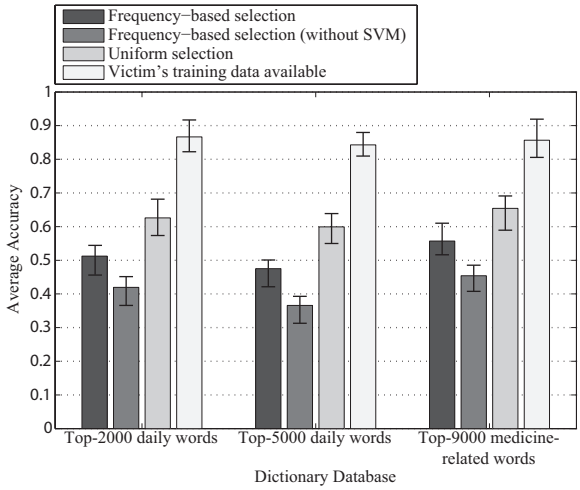


Figure 6. Accuracy of recognition for different databases.

To show the contribution of SVM based letter recognition, for each test we calculate the accuracy for the case where the final recognition result is selected from the candidate word list uniform-randomly, which rules out the contribution of SVM. We also repeat the tests for the case that words are selected

from the databases following uniform distribution. Furthermore, we collect 20 training data for each letter from each volunteer in #3-9, and evaluated the accuracy when the training data from victims are available. The selection of test words in this case also follows uniform distribution. The results are also included in Figure 6.

The results of experiments show that, under the condition of print-style writing and low near-field noise, the word recognizer of our system achieves an accuracy of around 50%-60% for all three databases. Since the dictionary filter narrows the space of candidate words, around 40% of written words can be correctly recognized even without any machine learning method. Applying the SVM further improves the accuracy. Moreover, the results for the medicine-related words confirm that WritingHacker can work well in specific attack scenarios such as clinics or hospitals. It can also be seen that the accuracy achieved when we apply frequency-based word selection is lower than that when words are uniformly selected. The reason is that the words used at a higher frequency are usually shorter, which enlarges the average size of candidate word lists in the case of frequency-based word selection and thus reduces the average accuracy. Usually, a victim only writes a few words in a privacy-related form, and the short function words such as 'the' are often omitted. Thus in the real case a higher accuracy should be achieved, which is confirmed by the test result that the recognition accuracy for the MIMIC II Clinic Database is higher than that for the daily words.

The system achieves high accuracy when the training data from the victim are available, which implies the possibility for a new input method. We discuss this in the next section.

During this test we assume that the words for test are included in the dictionary database. In the real case, it is possible that the written word does not hit the database. Although Zipf's law ensures that the hit rate would be high enough if the size of database is sufficiently large, we still design a test to evaluate the impact of the dictionary database's size in the following subsection.

Impacts of Dictionary Database Size

The size of the dictionary database controls the trade-off between the hit rate and the recognition accuracy under the condition that the word has hit the database. A larger database size implies that the word written by the victim is covered by the database with higher probability. However, this increases the space of candidate words for word recognizer and thus would reduce the accuracy even if the word has hit the database. On the other hand, if the database is too small, it is highly probable that the word written by the victim does not exist in the database, which leads to the failure of recognition directly. Fortunately, since the frequency distribution of commonly used words follows Zipf's law, it is possible for attackers to maintain a relatively small database while achieving an acceptable accuracy.

To evaluate the impact of dictionary database size, we use the top-9000 commonly used medicine-related words as a baseline database, and change the size of the dictionary database from top 1000 to 9000 with step 500. Similar to the previous tests, we assign the frequency of appearance as a word's weight in the baseline database. The words for test are randomly selected from the weighted baseline database, where the probability for a word to be selected is proportional to its appearance frequency. If the selected word is not covered by the dictionary database, the recognition for this word fails. Otherwise, it is recognized with the dictionary database. The same test procedure is followed as in the previous test, and 7000 samples are collected for each size of the dictionary database.

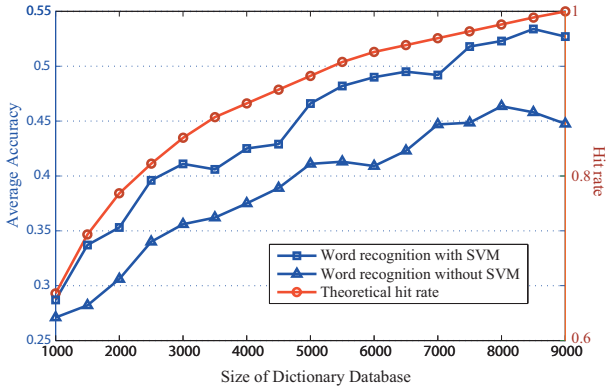


Figure 7. Accuracy of recognition when size of dictionary database changes.

Figure 7 shows the average accuracy when the size of dictionary database varies. We also draw the theoretical hit rate as well as the recognition accuracy when SVM is not applied and the output is randomly selected from the candidate word list. When the size of the dictionary database is much smaller than that of the baseline database, the enlargement of database size significantly increases the hit rate, and thus improves recognition accuracy. However, when the database's size is large enough, because of the Zipf's distribution of word usage, the growth rate of the hit rate is much lower. Meanwhile, the increased database size enlarges the candidate word lists, which in turn reduces the accuracy of the SVM based word classifier. This effect can be clearly observed in the case where SVM is not used: the accuracy drops when the database size is larger than 8000. The usage of SVM slows this drop. However, the slow-increasing hit rate makes the average recognition accuracy no longer increase significantly.

These results imply that it is unnecessary for the attacker to collect all the possible words for the specialized attack scenario. When the size of the dictionary database has been large enough, collecting more rarely used words does not help a lot. These results also confirm that even though our previous experiment uses subsets of English words only, the experiment still reflects the performance of WritingHacker in the real-word case because the top commonly used words have been covered by the dictionary databases.

Impacts of Distance between Device and Handwriting

One common concern about our attack method is the impact of the distance between the attacker's device and the victim's handwriting. To avoid being noticed by the victim, the attacker usually can only deploy the device such as the smartphone far away from the victim. In this subsection we prove that a

longer distance between the device and the handwriting does not seriously impact accuracy.

During the test we change the distance between the smartphone and the volunteer' handwriting from 40 cm to 200 cm with step 40 cm. For each test two volunteers #6,7 write 200 words which are randomly and uniformly selected from the top-2000 commonly used daily words. The training data come from volunteers #1,2, which are the same with those used in the previous tests. We repeat the test on two desks with different materials: wood and plastics.

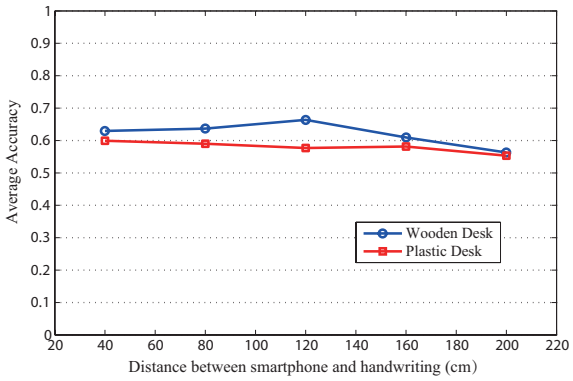


Figure 8. Average accuracy of recognition when distance and desk material change.

The experimental results in Figure 8 show that even though the accuracy decreases slightly with the distance, it is not seriously impacted and WritingHacker still keeps a recognition accuracy higher than 55% in both cases. This is caused by the truth that acoustic waves propagating along surfaces follow a lower attenuation rate, and thus the handwriting signals received by the smartphone through the desks are still identifiable. The comparable results for the wooden and plastic desk imply that it is possible for WritingHacker to maintain performance on different types of surface materials.

DISCUSSION

Possible Ways to Improve Accuracy

In our paper we propose a general attack method for unspecified writing scenarios. There are still several possible ways to further improve the performance of our method by taking the writing constraints into account. For instance, WritingHacker can provide a list of candidates (such as second/third-most-probable words) for each recognized word to help human observation. If a word from the handwriting on the form "Medical History" is recognized as "ANYWAY", and the second-most-probable word is "ANEMIA", it is easy for the attacker to correct the result by using the second candidate word.

Currently, we mainly focus on the word-level handwriting recognition, because people often write separate words instead of sentences when filling out forms. However, the sentence-level recognition is still applicable in the scenarios where victims write whole sentences. Attackers can further correct the recognized results based on sentence-level semantic laws and human observation. The candidate words can also help in this case.

Defend Against Attack

To defend against the eavesdropping attack proposed in this paper, a potential victim should check the desk for suspicious devices such as smartphones or detectaphones. Writing in a noisy environment can significantly increase the difficulty of eavesdropping. However, it is still possible to segregate the sound of handwriting by using multiple microphones [26]. If possible, the potential victim should also avoid print-style writing and use joined-up writing, which invalidates the stroke detector.

New Text Input Method

The high recognition accuracy achieved when the training data from victims are available implies the possibility that our system can be used as a new handwriting input method for small mobile devices. For small devices such as smartwatches, typing or writing on their small screens is inconvenient, and sometimes speech input methods are either disturbing or unavailable in case of broken network connections. If the accuracy of our technique could be further improved, it would allow users to write on desks with their fingers conveniently. We leave it to our future work to further improve the accuracy in this case.

Limitations of WritingHacker

As shown by the experiment settings, currently WritingHacker only works under controlled environments. We focus on the circumstance where victims follow certain print-style writing, and the near-field noise is low. In our evaluation, we collect data from nine volunteers, which seems to be a relatively small subject population, because a larger subject population would lead to a greater diversity of writing habits. Thus, WritingHacker is still a proof-of-concept system. However, WritingHacker has revealed the danger of a new audio-based attack on handwriting, which is the main purpose of our paper. It is our future work to design a fully functional system and conduct experiments with a larger group of subjects.

CONCLUSION

In this paper we present WritingHacker, a prototype system which explores the possibility of audio-based eavesdropping on handwriting via mobile devices. Based on letter clustering, we designed the components of input detector, dictionary filter and word recognizer, which enable mobile devices to record and recognize victims' handwriting. WritingHacker makes it possible for an attacker to violate a victim's privacy by keeping a mobile device (e.g., a smartphone) touching the desk being used by the victim. The experimental results show that, under certain conditions, the accuracy of word recognition reaches 50%-60%, which reveals the danger of privacy leakage through the sound of handwriting.

ACKNOWLEDGMENT

This research was funded by the National Science Foundation under award number CNS-1330491. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the sponsors.

REFERENCES

1. D. Asonov and R. Agrawal, “Keyboard acoustic emanations,” In IEEE symposium on security and privacy, pp. 3-11, 2004.
2. L. Zhuang, F. Zhou and J. D. Tygar, “Keyboard acoustic emanations revisited,” In Proceedings of the 12th ACM Conference on Computer and Communications Security, pp. 373-382, 2005.
3. Y. Berger, A. Wool and A. Yeredor, “Dictionary attacks using keyboard acoustic emanations,” In Proceedings of the 13th ACM conference on Computer and communications security, pp. 245-254, 2006.
4. T. Halevi and N. Saxena, “Keyboard acoustic side channel attacks: exploring realistic and security-sensitive scenarios,” International Journal of Information Security, pp. 1-14, 2014.
5. T. Deselaers, D. Keysers, J. Hosang and H. Rowley, “GyroPen: Gyroscopes for Pen-Input with Mobile Phones,” IEEE Transactions on Human-Machine Systems, vol. 45, no. 2, pp. 263-271, 2015.
6. C. Xu, P. H. Pathak and P. Mohapatra, “Finger-writing with Smartwatch: A Case for Finger and Hand Gesture Recognition using Smartwatch,” In Proceedings of the 16th ACM International Workshop on Mobile Computing Systems and Applications, pp. 9-14, 2015.
7. M. Zhang, P. Yang, C. Tian, L. Shi, S. Tang and F. Xiao, “SoundWrite: Text Input on Surfaces through Mobile Acoustic Sensing,” In Proceedings of the 1st ACM International Workshop on Experiences with the Design and Implementation of Smart Objects, pp. 13-17, 2015.
8. D. Balzarotti, M. Cova and G. Vigna, “Clearshot: Eavesdropping on keyboard input from video,” In IEEE Symposium on Security and Privacy, pp. 170-183, 2008.
9. D. X. Song, D. Wagner, and X. Tian, “Timing Analysis of Keystrokes and Timing Attacks on SSH,” In USENIX Security Symposium, vol. 2001, 2001.
10. P. Marquardt, A. Verma, H. Carter and P. Traynor, “(sp) iPhone: decoding vibrations from nearby keyboards using mobile phone accelerometers,” In Proceedings of the 18th ACM conference on Computer and communications security, pp. 551-562, 2011.
11. H. Wang, T. T.-T. Lai and R. R. Choudhury, “MoLe: Motion Leaks through Smartwatch Sensors,” In Proceedings of the 21st ACM Annual International Conference on Mobile Computing and Networking, pp. 155-166, 2015.
12. R. Plamondon and S. N. Srihari, “Online and off-line handwriting recognition: a comprehensive survey,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22(1), pp. 63-84, 2000.
13. T. Plötz and G. A. Fink, “Markov models for offline handwriting recognition: a survey,” International Journal on Document Analysis and Recognition (IJDAR), vol. 12(4), pp. 269-298, 2009.
14. Ø. D. Trier , A. K. Jain and T. Taxt, “Feature extraction methods for character recognition-a survey,” Pattern recognition, vol. 29(4), pp. 641-662, 1996.
15. M. Nakai, T. Sudo, H. Shimodaira and S. Sagayama, “Pen pressure features for writer-independent on-line handwriting recognition based on substroke HMM,” 16th International Conference on Pattern Recognition, vol.3, pp. 220-223, 2002.
16. J. S. Wang , Y. L. Hsu and C. L. Chu, “Online handwriting recognition using an accelerometer-based pen device,” 2nd International Conference on Advances in Computer Science and Engineering, pp. 229-232, 2013.
17. M. Bashir and J. Kempf, “Reduced dynamic time warping for handwriting recognition based on multidimensional time series of a novel pen device,” International Journal of Intelligent Systems and Technologies, WASET, vol. 2, no. 9, 2008.
18. S. Zhou, Z. Dong, W. J. Li and C. P. Kwong, “Hand-written character recognition using MEMS motion sensing technology,” IEEE/ASME International Conference on Advanced Intelligent Mechatronics, pp. 1418-1423, 2008.
19. R. Blum, S. Kassam, and H. Poor, “Distributed Detection With Multiple Sensors I. Advanced topics,” Proceedings of the IEEE, vol. 85, no. 1, 1997.
20. J. Wang, K. Zhao, X. Zhang and C. Peng, “Ubiquitous keyboard for small mobile devices: harnessing multipath fading for fine-grained keystroke localization,” In Proceedings of the 12th ACM annual international conference on Mobile systems, applications, and services, pp. 14-27, 2014.
21. S. T. Piantadosi, “Zipfąŕs word frequency law in natural language: A critical review and future directions.” Psychonomic bulletin & review, vol.21, no. 5 pp. 1112-1130, 2014.
22. M. Saeed, M. Villarroel, A. T. Reisner, G. Clifford, L.-W. Lehman et al., “Multiparameter intelligent monitoring in intensive care II (MIMIC-II): a public-access intensive care unit database,” Critical Care Medicine, vol. 39, no. 5, p. 952, 2011.
23. Word Frequency Dat, Corpus of Contemporary American English, accessed March 13, 2016, http://www.wordfrequency.info
24. G. Guo and S. Z. Li, “Content-based audio classification and retrieval by support vector machines,” IEEE Transactions on Neural Networks, vol. 14, no. 1, pp. 209-215, 2003.
25. J. Weston, A. Elisseeff, G. BakIr, F. Sinz. The Spider. http://people.kyb.tuebingen.mpg.de/spider/
26. M. Aoki, M. Okamoto, S. Aoki, H. Matsui, T. Sakurai and Y. Kaneda, “Sound source segregation based on estimating incident angle of each frequency component of input signals acquired by multiple microphones,” Acoustical Science and Technology, vol. 22, no. 2, pp. 149-157, 2001.