VISIBLE: Video-Assisted Keystroke Inference from Tablet Backside Motion

Jingchao Sun*, Xiaocong Jin*, Yimin Chen*, Jinxue Zhang*, Rui Zhang†, and Yanchao Zhang*
* Arizona State University
{jcsun, xcjin, ymchen, jxzhang, yczhang}@asu.edu
† University of Hawaii
ruizhang@hawaii.edu

***Abstract*—The deep penetration of tablets in daily life has made them attractive targets for keystroke inference attacks that aim to infer a tablet user's typed inputs. This paper presents VISIBLE, a novel video-assisted keystroke inference framework to infer a tablet user's typed inputs from surreptitious video recordings of tablet backside motion. VISIBLE is built upon the observation that the keystrokes on different positions of the tablet's soft keyboard cause its backside to exhibit different motion patterns. VISIBLE uses complex steerable pyramid decomposition to detect and quantify the subtle motion patterns of the tablet backside induced by a user's keystrokes, differentiates different motion patterns using a multi-class Support Vector Machine, and refines the inference results using a dictionary and linguistic relationship. Extensive experiments demonstrate the high efficacy of VISIBLE for inferring single keys, words, and sentences. In contrast to previous keystroke inference attacks, VISIBLE does not require the attacker to visually see the tablet user's input process or install any malware on the tablet.**

I. Introduction

The past few years have witnessed the proliferation of tablets in everyday life. According to a Gartner report [1], global tablet shipments will reach 321 million and surpass PC shipments in 2015. Being lighter than laptops and having larger touchscreens than smartphones, tablets perfectly fill the gap between laptops and smartphones and have become an indispensable category of mobile computing devices. People are increasingly using tablets in every aspect of life, including voice/video communications, Internet browsing, web transactions, online banking, reading, multimedia playing, etc.

The deep penetration of tablets in people's daily life has made them attractive targets for various keystroke inference attacks that aim to infer a user's typed inputs (such as usernames, passwords, SSNs, and emails) on the tablet touchscreen. Although existing authentication schemes [2]–[5] can prevent unauthorized access to mobile devices, prior work has shown that an attacker can successfully infer the PIN or even the words entered on the soft (tablet) keyboard by surreptitiously video-recording a target user's input process and then analyzing the reflection of the touchscreen, spatial hand dynamics, or the relative finger positions on the touchscreen [6]–[13]. These studies commonly assume that the attacker can capture a user's interaction with the touchscreen with little or no visual obstruction, which greatly limits the applicability of these attacks.

In this paper, we propose VISIBLE, a novel video-assisted keystroke inference framework that allows an attacker to infer a tablet user's typed inputs on the touchscreen by recording and analyzing the video of the tablet backside during the user's input process. VISIBLE is motivated by our observation that the keystrokes on different positions of the tablet's soft keyboard cause its backside to exhibit different motion patterns. In contrast to previous keystroke inference techniques [6]–[13], VISIBLE does not require the attacker to visually see the victim's input process and thus enables much more surreptitious keystroke inference from a distance.

The design of VISIBLE faces two major challenges. First, the backside motion caused by user keystrokes is very subtle and requires effective methods to detect and quantify. Second, since the soft keyboard on the tablet is much smaller than a normal keyboard, the motion patterns caused by tapping adjacent keys are close, making accurate differentiation particularly challenging. To tackle these two challenges, VISIBLE uses complex steerable pyramid decomposition to detect and quantify the subtle keystroke-induced motion patterns of the tablet backside, differentiates different motion patterns using a multi-class Support Vector Machine, and refines the inference results using a dictionary and linguistic models.

We thoroughly evaluate the performance of VISIBLE via comprehensive experiments on an Apple iPad 2 tablet and a Google Nexus 7 tablet. Our experiment results show that VISIBLE can infer a single key entered on the alphabetical soft keyboard with an average accuracy of 36.2% and that the correct key is within the inferred key's one-hop and two-hop neighbors with probabilities 83.6% and 97.9%, respectively. Similarly, VISIBLE achieves an accuracy of 38% for single-key inference on the PIN soft keyboard, and the correct key is within the inferred key's one-hop neighbors with probability 68%. For word inference, VISIBLE can produce a list of candidate words, and the correct word is in the top-5, top-10, top-25, and top-50 candidate words with probabilities 48.0%, 63.0%, 77.8%, and 92.6%, respectively. We also show that the attacker can successfully infer typed sentences based on the linguistic relationship between adjacent words. These experiment results confirm the high efficacy of VISIBLE.

NDSS '16, 21-24 February 2016, San Diego, CA, USA

http://dx.doi.org/10.14722/ndss.2016.23060

The rest of this paper is organized as follows. Section II presents the related work. Section III introduces some background knowledge for video processing. Section IV describes the adversary model. Section V details the design of VISIBLE. Section VI evaluates VISIBLE through extensive experiments. Section VII concludes this paper and discusses possible countermeasures and future work.

II. Related Work

In this section, we briefly introduce the prior work most related to VISIBLE. Prior keystroke inference attacks can be broadly classified into two categories: video-based attacks and sensor-based attacks.

a) Video-based Attacks: In this category, the adversary uses video-based side channels in combination with computer vision techniques to infer a user's typed inputs. Early work along this line focuses on physical keyboards. Backes *et al.* [6], [7] exploited the reflections of screens on glasses, tea pots, spoons, plastic bottles, eyes of the user, walls, and even the user's clothes to recover the content displayed on the computer monitor. Balzarotti *et al.* [8] introduced an attack that automatically recovers the typed text solely from a video of the user typings by analyzing the light diffusion surrounding the key change. This attack requires a camera to directly record the finger typings on the physical keyboard.

There have also been some video-based attacks on the soft keyboards of touchscreen mobile devices. In [9], Maggi *et al.* presented an attack that automatically recognizes typed inputs from the key magnifications of touchscreen mobile devices. Raguram *et al.* [10] showed how to automatically reconstruct the text input on a soft keyboard from the reflection of the device's screen on the victim's sunglasses. Xu *et al.* [11] introduced an attack to accurately reconstruct the text input on a mobile device by tracking the positions of the victim's fingers as they move across the soft keyboard. In [12], Yue *et al.* showed how to infer the user input even if neither text nor popup can be seen from the video of user typings. This attack exploits the homographic relationship between touching images and a reference image showing a soft keyboard. All these attacks require the attacker to acquire a video capturing the victim's typings on the touchscreen or the touchscreen reflection.

Our work is most related to [13], in which Shukla *et al.* introduced a video-based attack on the PIN-entry process of a smartphone that decodes the typed PIN by exploiting the spatiotemporal dynamics of the hands during typing. Both VISIBLE and the attack proposed in [13] only require the attacker to video-record the backside of a smartphone, which was considered safe previously. VISIBLE, however, has much wider applicability than [13]. In particular, the attack introduced in [13] requires the attacker to record the victim's hand movements during the PIN-entry process, which is not always possible. For example, the victim's hand movements are very likely to be obscured by the tablet itself. In contrast, VISIBLE works even if the victim's hand movements are not visible from the video of the device backside.

b) Sensor-based Attacks: Tremendous efforts have been made on inferring user inputs on mobile devices from the data generated by various on-board sensors. It has been shown in [14] and [15] that the user's password can be inferred from the smartphone's accelerometer data. Moreover, some recent work [16], [17] demonstrated a similar attack that exploits the data from both accelerometer and gyroscope. Other on-board sensors that have been exploited include microphones and front cameras [18], [19]. All these work require the attacker to obtain sensor data via either malicious applications (e.g., malicious Apps or web scripts) or unprotected network transmissions, which limit their applicability. In contrast, VISIBLE only requires the attacker to record the video of the tablet backside during the victim's typing process, which is both easier to launch and more difficult to detect.

Also related is the work on using on-board sensors to infer the keystrokes of nearby physical keyboards. In [20], Zhuang *et al.* showed how to recover typed keys from sound recordings of a user's typings on a physical keyboard. Berger *et al.* [21] presented another attack that infers the user input from the acoustic emanations of the physical keyboard with the assistance of a dictionary. A similar attack was presented in [22], which also uses acoustic emanations of the physical keyboard but does not need a language model or dictionary. In [23], the authors demonstrated an attack that infers the typed keys of a physical keyboard from the vibration caused by each keystroke detected by a nearby smartphone's accelerometer. Such attacks, although effective, can only be used when the attacker is near the victim due to the short transmission range of acoustic and vibration signals. In contrast, VISIBLE can be launched from a much larger distance.

III. Video Processing Basics

In this section, we introduce two computer vision techniques, phase-based optical flow estimation and complex steerable pyramid decomposition, underlying VISIBLE.

A. Phase-based Optical Flow Estimation

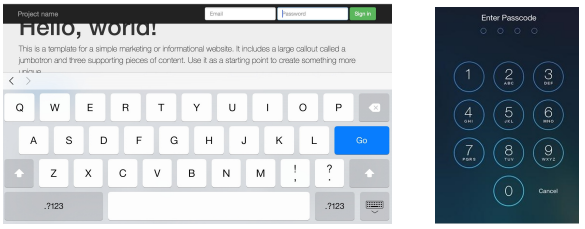
An optical flow refers to apparent motion patterns of image objects between two consecutive frames caused by the object or camera movement. Optical flow estimation is the process of characterizing and quantifying the object motions in a video stream, often for motion-based object detection and tracking systems. Phase-based optical flow is a popular optical flow estimation technique which estimates the motion field using phase information. For example, constant phase contours are tracked by computing the phase gradient of spatiotemporally bandpassed images, which provides a good approximation to the motion in [24]. As another example, Gautama *et al.* [25] proposed to estimate motion by computing the temporal gradient of the phases of a partially bandpassed video. In comparison with other flow estimation techniques, phased-based estimation methods are more robust to smooth shading, lighting variations, and small deviations from image translations.

B. Complex Steerable Pyramid Decomposition

Steerable pyramid decomposition [26] is a standard technique that decomposes an image according to spatial scale, orientation, and position to capture the variance of a texture in both intensity and orientation, which has been widely used in image processing and motion detection. Since an image may

(a) An iPad and a holder. (b) Attack scenario. (c) The same attack scenario from a different angle.

Fig. 1. Examples of a tablet holder and an attack scenario.



(a) Alphabetical keyboard. (b) PIN keyboard.

Fig. 2. Alphabetical and PIN soft keyboard illustrations.

contain multiple objects of different sizes, and these objects may contain features of different sizes and be at different distances from the viewer, any analysis procedure that is only applied at a single scale may lose information at other scales. To simultaneously detect multiple objects' motion patterns, analysis need be carried out at different scales simultaneously [27]. In addition, the same object may also exhibit totally different features in different orientations. To comprehensively analyze the features of an object and detect its motion pattern, it is necessary to decompose it in different orientations.

Complex steerable pyramid decomposition [28] extends the original steerable pyramid decomposition by representing an image in a complex form comprising real and imaginary parts. In comparison with steerable pyramid decomposition, complex steerable pyramid decomposition additionally measures local phase and energy in some texture descriptors. Such measures have proved important throughout computer vision. Using complex steerable pyramid decomposition, we can obtain the phase and amplitude of each pixel of an image at each spatial scale and orientation over time.

IV. Adversary Model

We consider a victim user with a tablet such as iPad 2 or Nexus 7. We assume that the victim places the tablet on a tablet holder (e.g., the one shown in Fig. 1(a)) on a desk and types on a soft keyboard. Such scenarios are very common in daily life, e.g., in conferences or seminars where researchers take notes or write emails. We focus on two types of soft keyboards in this paper, the alphabetical and PIN keyboards, as shown in Fig. 2. The extension of VISIBLE to the alphanumeric soft keyboard is left as future work.

We assume that an attacker intends to infer any typed input on the victim's tablet, which can be single keys, words, or sentences. This enables VISIBLE to infer any sensitive typed input on the tablet, such as usernames, passwords, and emails. The victim is alert to shoulder-surfing attacks in the sense that the attacker cannot get too close to the victim during his typing process. In addition, the attacker is unable to see the tablet touchscreen or the victim's hand movement during his typing process from any direction. Moreover, we assume that the attacker cannot obtain the sensor data by running malware such as Trojans or malicious web scripts on the victim's tablet. These assumptions make previous video-based attacks [6]–[13] and sensor-based attacks [14]–[23] inapplicable.

We assume that the attacker has the following capabilities. First, the attacker can use camcorders with advanced lens to record the backside of the victim's tablet during his input process, possibly from a long distance. Second, the attacker can record the attack scenario and reconstruct it afterwards. Specifically, the attacker can measure the angle between the victim's tablet and the desk, the angle between the tablet and the camcorder, and the distance between the tablet and the camcorder by analyzing multiple images taken from different angles and positions using distance and angle estimation algorithms [29]. Finally, the attacker has the same holder and the tablet with the same soft keyboard layouts as the victim's.

V. VISIBLE Framework

In this section, we give an overview of VISIBLE and then detail each step of the attack.

A. VISIBLE Overview

VISIBLE infers the victim's typed inputs from the video of tablet backside motion. The high-level design of VISIBLE is shown in Fig. 3, which consists of eight steps as follows.

1) *Video Recording and Preprocessing*: In this step, we record a video capturing the motion of the tablet backside during the victim's typing process. We assume neither the touchscreen nor the victim's hand movement can be seen from the video. We crop the video clip to keep the text-input part only.
2) *Areas of Interests (AOIs) Detection and Selection*: In this step, we detect all the areas with texture information on the tablet backside and select a few areas as AOIs for further processing. Exemplary AOIs are the buttons, camera, loudspeaker, logo, and texts on the tablet backside.

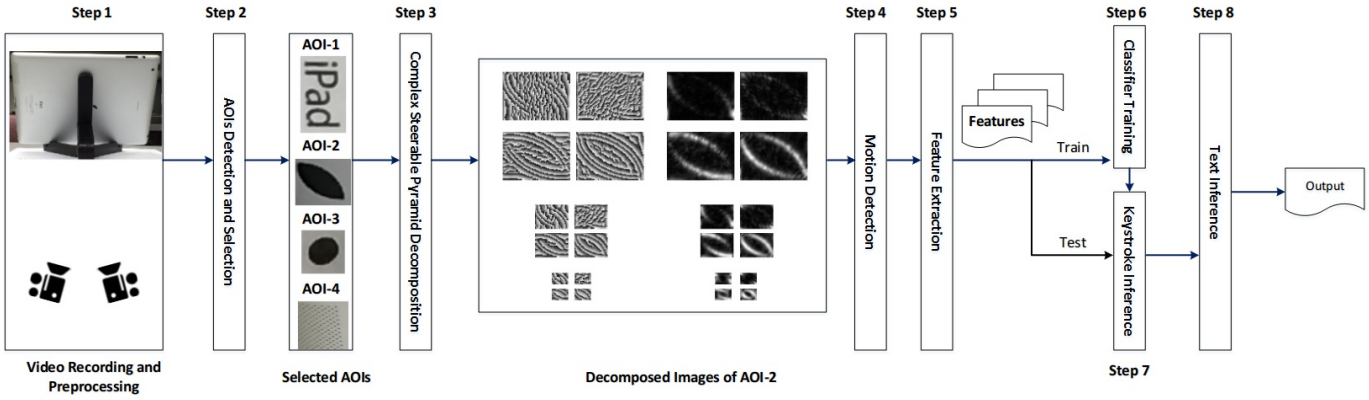


Fig. 3. The VISIBLE framework.

3) *AOI Decomposition*: In this step, we decompose each selected AOI in each frame using complex steerable pyramid decomposition.
4) *Motion Detection via Phase Variances*: In this step, we analyze the phase variances of each AOI over time and quantify the corresponding motion amplitude.
5) *Feature Extraction*. In this step, we extract features in both temporal and spatial domains to represent the motion patterns of each AOI.
6) *Classifier Training*. In this step, we let multiple attackers mimic the victim's typing process, record videos of their tablet backsides, and process their videos to train a classifier.
7) *Keystroke Inference*. In this step, we use a classifier trained in Step 6) to test the collected data from the victim's typing process.
8) *Text Inference*. In this step, we infer the possible words by considering meaningful alphabetical combinations using a dictionary and exploit the relationship between adjacent words to further infer sentences.

In what follows, we present each step in detail.

B. Video Recording and Preprocessing

One or more camcorders can be used to video-record the tablet backside during the victim's typing process. For more accurate inference results, the video should capture the entire backside of the tablet, and the image resolution should be sufficiently high. In our experiments, we use two camcorders that focus on the left-half and right-half of the tablet backside, respectively. By doing so, each camcorder only needs to focus on a relatively small area with sufficiently high resolution to capture the detailed texture information on the tablet backside.

In our study, we find that the following four factors affect subsequent motion detection.

- *Image resolution and frame rate*. The image resolution determines the amount of detailed textural information that can be obtained from the image and should be as high as possible. The frame rate is the number of frames taken within one second, and a higher frame rate could help capture more detailed motion information. We recommend 1080p60[1] HD or higher resolutions and frame rates.
- *Zoom setting*. The zoom setting of the camcorder is jointly determined by the distance between the camcorder and the tablet and the lens properties. We recommend zooming in as much as possible while capturing the entire tablet backside.
- *Light condition*. The bright light condition can result in better motion detection, as the imaging component of camcorders generates larger random noise in low-light condition that pollutes the motion signal.
- *Recording angle*. The angle between the camcorder and tablet need be adjusted to capture the entire tablet backside, which can be easily satisfied in practice.

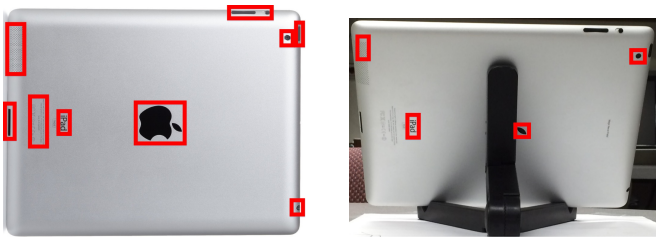
We also video-record the attack scenario layout to measure the distances and angles between the camcorders and the target tablet as well as the angle between the tablet and the desk. This is important for reconstructing the attack scenario later. In practice, the attacker can take multiple images of the attack scenario from different angles and estimate the distances and angles of interest using standard distance and angle estimation algorithms such as [29].

After obtaining the video of the tablet backside, we manually crop the unwanted part such that the remaining video contains only the typing process of interest that is easily identifiable in practice. In practice, for the PIN keyboard, since the user usually needs to press power or home button first, the keys entered subsequently are very likely to be PINs. Similarly, for the alphabetical keyboard, the typing process can be identified by continuous typings (e.g., small arm movements). Finally, we decompose the cropped video into a series of frames. Note that video croppings can be automated after more sophisticated and extensive programming effort.

C. AOIs Detection and Selection

After obtaining a series of frames, we proceed to identify all the Areas of Interests (AOIs) on the tablet backside, where each AOI refers to an area containing unique texture

[1] 1080p60 denotes that the camcorder can take 1920x1080 videos at a rate of 60 frames per second.



(a) Possible AOIs. (b) Selected AOIs.

Fig. 4. Possible and selected AOIs on an iPad 2's backside.

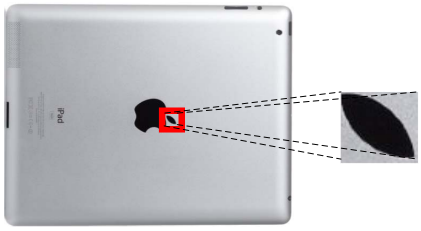


Fig. 5. An example of a selected AOI, AOI-2.

information. In computer vision, the texture information of an image refers to the relationship of nearby pixels. Given a frame, we use the textured area detection algorithm [30] to identify the areas with texture information and select them as AOIs for subsequent motion detection.

Fig. 4(a) shows an image of the iPad 2 backside, where the areas enclosed by rectangles are identified AOIs, which include the power button, the voice up and down buttons, the silence button, the backside camera, the ear plug, the loudspeaker, the logo, the manufacture information, and other decorative patterns. These AOIs can be used to detect the tablet backside motion. In practice, almost every tablet's backside contains adequate areas with rich texture information. Even if the tablet is protected by a backside cover like iPad Smart Case, the backside of the cover itself still contains areas with rich texture information making it easy to find enough AOIs for subsequent motion detection.

We then select a subset of the AOIs that are near the edges of the tablet backside and separated from each other, as these AOIs tend to have larger and more distinctive motions than others, making them more capable of differentiating the motion patterns caused by different keystrokes. Fig. 4(b) shows the backside image of an iPad 2 placed on a holder, where the areas specified by rectangles are the selected AOIs.

D. Decompositions of Selected AOIs

Now we have a series of frames extracted from the cropped video, each containing the same set of selected AOIs. For each frame, we use complex steerable pyramid decomposition [28] to decompose each selected AOI into complex sub-bands. As an example, Fig. 5 shows an AOI that is part of the Apple logo, and Fig. 6 shows the image decomposed via complex steerable pyramid decomposition. More specifically, Figs. 6(a) to 6(d) show the real part, imaginary part, phase, and amplitude of the decomposed image at three scales and four orientations, respectively. We can see from Fig. 6(c) that the image has more features at orientation 3 and 4 and less features at orientation 2. Since the phase variations are proportional to subtle motions, Fig. 6(c) indicates that more subtle motions can be detected at orientation 3 and 4. Finally, we obtain a decomposed complex steerable pyramid for each selected AOI in each frame.

E. Motion Detection via Phase Variances

To estimate the motion for each selected AOI over time, we first compute the pixel-level motion. As in [31], [32], for each selected AOI, we first decompose its frame series using complex steerable pyramid and then compute the pixel-level motion from the amplitude and phase of its pixels. Specifically, complex steerable pyramid decomposition adopts a filter bank to decompose each frame into complex sub-bands corresponding to each scale and orientation. The complex steerable pyramid decomposition of a frame at time t at scale r and orientation θ can be written as

$$A(t,x,y,r,\theta)e^{i\phi(t,x,y,r,\theta)}, \tag{1}$$

where x and y are the pixel coordinates in x-axis and y-axis at scale r, respectively, and $A(t,x,y,r,\theta)$ and $\phi(t,x,y,r,\theta)$ are the amplitude and phase at coordinate $(x,\ y)$ of the decomposition at scale r and orientation θ, respectively.

We then calculate the phase variance at scale r and orientation θ as

$$\Delta\phi(t,x,y,r,\theta) = (\phi(t,x,y,r,\theta) - \phi(t_0,x,y,r,\theta)) \bmod 2\pi, \tag{2}$$

where t_0 is the time for any initial frame. According to [25], the phase variations $\Delta\phi(t,x,y,r,\theta)$ are approximately proportional to displacements to image structures along the corresponding scale and orientation.

Finally, we estimate each selected AOI's motion using its pixel-level motion. Since the pixel-level phase variance $\Delta\phi(t,x,y,r,\theta)$ is an approximation of the pixel-level motion, an intuitive way to estimate the motion of the AOI is to sum the phase variation $\Delta\phi(t,x,y,r,\theta)$ of all its pixels. However, the pixel-level phase variance approximates the pixel-level motion only if the area has rich texture information. For areas with little texture information, the pixel-level phase variance is random due to background noise. To simultaneously strengthen the pixel-level phase variation for areas with rich texture information and weaken the pixel-level phase variation for areas with little texture information, we compute a weighted sum of phase variances at scale r and orientation θ as

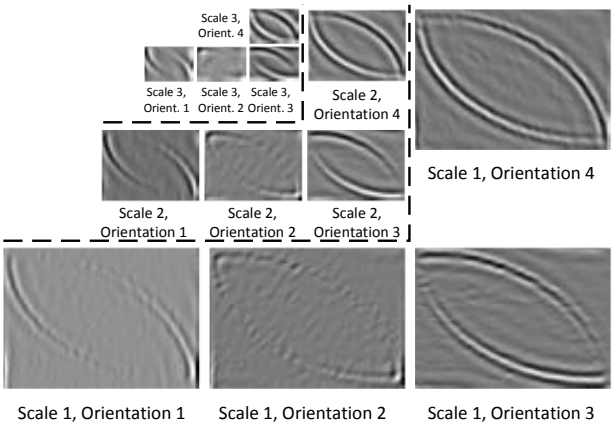
$$\Phi(t,r,\theta) = \sum_{x,y} A(t,x,y,r,\theta)^2 \Delta\phi, \tag{3}$$

where $A(t,x,y,r,\theta)$ is the measure of texture strength.

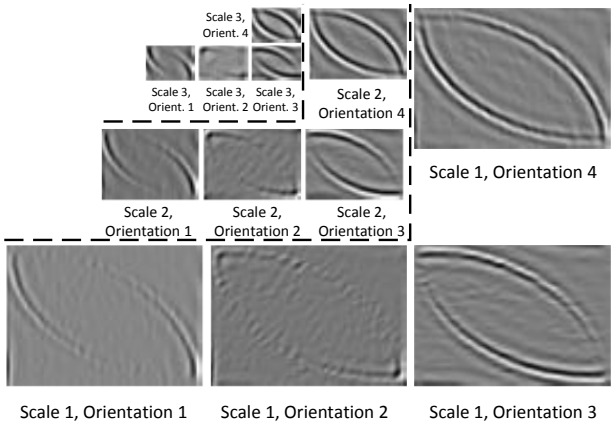
Since a frame is decomposed into multiple scales and different orientations, we sum the motions for all the scales and orientations to obtain the estimated motion for the specific AOI as

$$\Psi(t) = \sum_{r,\theta} \Phi(t,r,\theta) = \sum_{r,\theta,x,y} A(t,x,y,r,\theta)^2 \Delta\phi. \tag{4}$$

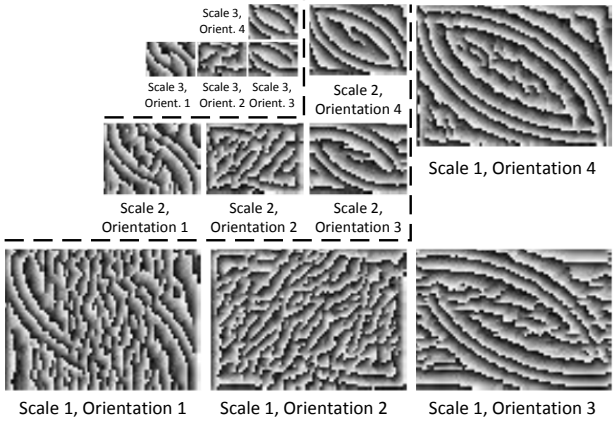
Fig. 7 depicts the motion signals of the apple stem in Fig. 5 during a word-entry process. We can see the typed word with



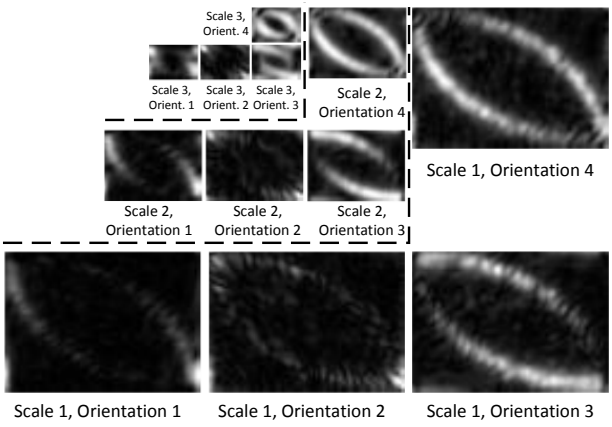
(a) Real parts of the decomposed image at each scale and orientation.



(b) Imaginary parts of the decomposed images at the same scale and orientation.



(c) Phase of oriented band-pass images at each scale and orientation.



(d) Amplitude of the decomposed images at the same scale and orientation.

Fig. 6. A 3-scale, 4-orientation complex steerable pyramid representation of AOI-2.

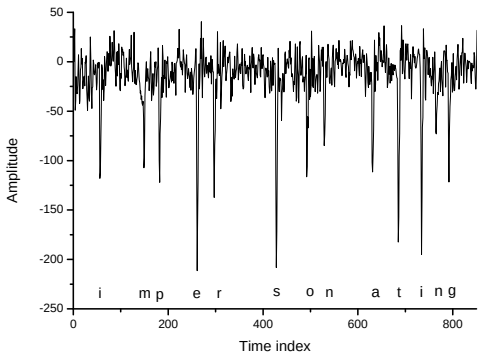


Fig. 7. Motions of the apple stem in Fig. 5 for typing "impersonating".

thirteen letters each corresponding to a peak in amplitude, i.e., a sudden significant change in $|\Psi(t)|$.

F. Feature Extraction

Now we extract temporal and spatial features from selected AOIs' motion signals to represent the motion patterns. The former are obtained from the motion signals' time domain, and spatial features depict the motion relationship among different AOIs that is capable of reflecting the posture of the tablet.

To extract temporal features, we represent the motion sequence of each AOI as a vector that specifies the time-varying motion amplitude and then derive the following features for each AOI.

- *Skewness*. This refers to the third central moment which measures the asymmetry of the vector.
- *Kurtosis*. This is the fourth central moment which measures the peakedness or flatness of the vector.
- *Maximum motion amplitude*. The maximum motion amplitudes are different for different AOIs.
- *Relative and absolute differences between maximum motion amplitudes*. Assume that there are n selected AOIs. We define a ratio vector which comprises the ratio of the maximum motion amplitude of the i-th AOI to that of the $(i+1)$-th AOI for all $i \in [1, n-1]$. We also define a *difference vector* comprising the maximum motion amplitude of the i-th AOI subtracted by that of the $(i+1)$-th AOI for all $i \in [1, n-1]$.

To extract spatial features, we denote the motions of all AOIs by matrix $A_{m\times n}$, where m is the time index, n is the number of AOIs, and $A_{i,j}$ is the j-th AOI's motion amplitude at time i. We derive the following spatial features.

- *1-norm*. The 1-norm of $A_{m\times n}$ is calculated as

$$||A_{m\times n}||_1 = \max_{1\le j\le n} \sum_{i=1}^{m} |a_{ij}| \ ,$$

  which is its maximum absolute column sum.

- *2-norm*. As in [16] we calculate three 2-norm features from $A_{m\times n}$. Let R_i denote the ith row of $A_{m\times n}$ for all $i \in [1, m]$. The 2-norm of each R_i is given by

$$||R_i||_2 = \sqrt{\sum_{j=1}^{n} |a_{ij}|^2}.$$

  We then extract the mean, maximum, and minimum from

$$[||R_1||_2, ||R_2||_2, \ldots, ||R_m||_2]^T.$$

- *Infinity-norm*. The infinity-norm of $A_{m\times n}$ is

$$||A_{m\times n}||_\infty = \max_{1\le i\le m} \sum_{j=1}^{n} |a_{ij}| \ ,$$

  which is its maximum absolute row sum.

- *Frobenium-norm*. The frobenium-norm is calculated as

$$||A_{m\times n}||_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2} \ ,$$

  which is the square root of the squared sum of the matrix elements.

- *Pearson correlation*. The Pearson correlation measures the correlation of the motion vectors of different AOIs during the same typing process. For two motion vectors V_i and V_j of two AOIs i and j, respectively, the Pearson correlation is defined as

$$P_{ij} = \frac{\mathsf{cov}(V_i, V_j)}{\sigma_i \sigma_j},$$

  where $\mathsf{cov}(V_i, V_j)$ is the covariance between V_i and V_j, and σ_i and σ_j are the standard deviation of V_i and V_j, respectively.

G. Classifier Training

To train a classifier, we first reconstruct the attack scenario from the images taken in the video recording phase using standard distance and angle estimation algorithms such as [29].

We then let multiple attackers type on every key position of the soft keyboard for multiple times, during which we record the videos of the tablet backside as well as the typed keys. We finally obtain the training data set consisting of NKM samples, where N is the number of attackers that mimic the victim, K is the number of keys on the soft keyboard, and M is the number of times each key is typed by each attacker.

We use a multi-class Support Vector Machine (SVM) [33] with C-SVC type and linear kernel to distinguish different typed keys. Specifically, we use the implementation in WEKA [34] with default parameters. Since we have already obtained NKM labeled typing samples, we feed them into WEKA to obtain a trained multi-class SVM classifier.

H. Keystroke Inference

In this step, we use the motion data extracted from the video recording of the victim's tablet backside and the trained classifier to obtain a candidate key set for each key the victim typed. Specifically, for a backside video capturing the victim's typing process, it is processed through Steps 1 to 5 to output 36 features in total, including four skewness features, four kurtosis features, four maximum motion features, six relative difference features, six absolute difference features, six Pearson correlation features, one one-norm feature, three two-norm features, one infinity-norm feature, and one Frobenium-norm feature. We then use the trained multi-class SVM classifier to predict one key. Since the distance between two adjacent keys in both alphabetical and PIN keyboards is very small, it is possible for the key entered by the victim to be misclassified as neighboring keys. We therefore let the SVM classifier output a candidate key set consisting of all the keys that are no more than h hops from the predicated key, where h is a parameter determined by the attacker.

I. Text Inference

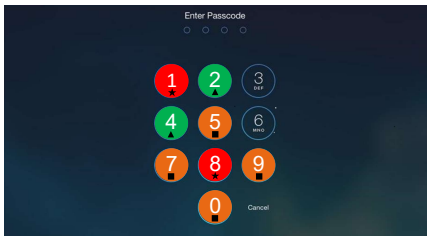
In this step, we further infer the entered text by using a dictionary and a linguistic relationship between adjacent words. Specifically, for a word consisting of W letters, we can obtain W candidate letter sets in the previous step. We use the "corn-cob" dictionary [35] that contains over 58,000 lower-case English words and is also used by previous work [21]. First, we list all the combinations of the possible words and filter out the combinations that are not in the dictionary. We then manually select one word from each of the candidate lists to form a meaningful sentence by considering the linguistic relationship between adjacent words. As an alternative, given the candidate word list for each word, we may use a well-studied n-gram model such as [36] generated from linguistic statistics to generate candidate sentences.

VI. Performance Evaluation

In this section, we evaluate the performance of VISIBLE through extensive experiments on a 9.7-inch Apple iPad 2 tablet with iOS 8 and a 7-inch Google Nexus 7 tablet with Android 4.4. The experiments involved eight participants in total, and the data collection process has been approved by Institutional Review Board (IRB) at our institution. We intend to answer the following five questions in our evaluations.

1. What is the (single-)key inference accuracy on alphabetical and PIN keyboards, respectively?
2. What is the word inference accuracy on the alphabetical keyboard?
3. Is it possible to infer a victim's typing sentences?
4. How do the inference results differ on different tablets (e.g., an iPad 2 and a Nexus 7)?

(a) One-hop and two-hop neighbors of letters "a" and "j".



(b) One-hop neighbors of keys 1 and 8.

Fig. 8. Examples of one and two-hop neighbors on alphabetical and PIN keyboards.

5. What are the impacts of environmental factors (e.g., the light conditions, the angle between the tablet and the camcorders, and the imperfect reconstruction of attack scenario) on keystroke inference accuracy?

A. Experiment Design

In our experiment, we used two commercial off-the-shelf (COTS) camcorders to video-record the tablet backside during the victim's typing process. One camcorder is a Panasonic HC-V700 with $21\times$ zoom lens, which can record 1080p60 HD videos and feature an intelligent zoom function that supports up to $46\times$ zoom. The second camcorder is a Sony FDR-AX100 with $10\times$ zoom lens, which can record 4Kp30[2] or 1080p60 HD videos and support up to $160\times$ zoom.

We placed an Apple iPad 2 tablet with iOS 8 on a holder as shown in Fig. 4(b) and two camcorders 1.8 meters away from the tablet. The distance between the attacker's camcorders and the victim's tablet can be increased as long as the attacker is equipped with more advanced lens (e.g., telephoto lens) to video-record the tablet backside at a distance. The angle between each camcorder and the tablet was 90 degree by default, and we evaluated the impact of different angles as well. The two camcorders focused on the left-half and right-half of the tablet backside, respectively. We simultaneously used two camcorders because one camcorder cannot simultaneously include all the AOIs and have sufficiently high resolution for each AOI.

Let $\Omega^h(i)$ be key i's h-hop neighborhood, including key i itself. As two examples, Fig. 8(a) shows the one-hop and two-hop neighbors of letters "a" and "j", where the orange and yellow keys (marked by triangle and square) are the one-hop and two-hop neighbors, respectively. Fig. 8(b) shows the one-hop neighbors of keys 1 and 8, where the green and orange keys (marked by triangle and rectangle) are the neighbors of key 1 and key 8, respectively.

We use the following two metrics to evaluate the inference accuracy of VISIBLE.

- *Pinpoint accuracy* P_i. The probability that a key i typed by the victim is correctly inferred as i.
- *h-hop accuracy* P_i^h. The probability that a key i typed by the victim is inferred as some key in $\Omega^h(i)$.

By letting $\Omega^0(i) = \{i\}$, we can see that the pinpoint accuracy is a special case of h-hop accuracy, as $P_i = P_i^0$. We consider both pinpoint and h-hop accuracies for two reasons. First, the capability of narrowing down a typed key to a small area still poses a serious threat to user privacy, as the attacker can still learn sensitive information. Second, considering the neighborhood of a key instead of only the key itself is particularly important for word and sentence inference, as we will see shortly. In this paper, we consider $h = 0, 1, 2$, and 3 for alphabetical keyboard and $h = 0$ and 1 for PIN keyboard.

[2]4Kp30 denotes that the camcorder can take 3840×2160 video at a rate of 30 frames per second.

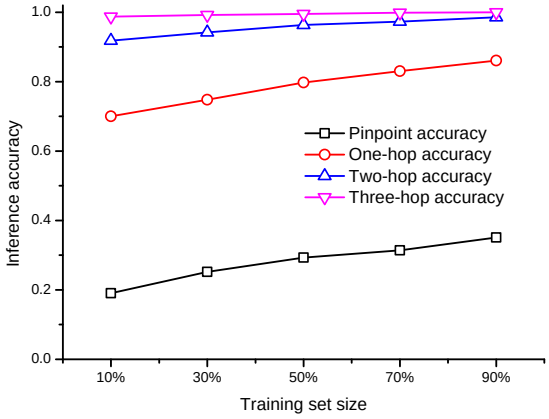


Fig. 9. Impact of the training set size.

B. Alphabetical Keyboard Experiment

We first report the performance of VISIBLE on the alphabetical keyboard of an iPad 2 tablet with iOS 8, on which keystroke inference is challenging for two reasons. First, the distance between two adjacent keys is very small, while we need to distinguish at least 26 different keys. Second, the alphabetical keyboard is usually located at the bottom of the touchscreen which makes the motions caused by keystrokes less noticeable.

In this experiment, we involved four participants and let each participant type each English letter 20 times and collected $20 \times 26 \times 4 = 2080$ keystrokes in total. We selected a portion of data from the collected dataset as a training set and used the rest of the collected data as a test set. We trained a multi-class SVM classifier using the training set and then tested it using the test set. We used 10-fold cross-validation to test the performance of the key inference of VISIBLE.

Table I compares the pinpoint and h-hop accuracies of VISIBLE and random guess for each English letter on the alphabetical keyboard. We can see that the pinpoint accuracy

TABLE I. KEY INFERENCE RESULTS FOR ALPHABETICAL KEYBOARD, WHERE VIS AND RG DENOTE VISIBLE AND RANDOM GUESS, RESPECTIVELY.

Key	P_i		$\|\Omega^1(i)\|$	P_i^1		$\|\Omega^2(i)\|$	P_i^2		$\|\Omega^3(i)\|$	P_i^3	
	VIS	RG		VIS	RG		VIS	RG		VIS	RG
a	33.8%	3.84%	5	78.8%	19.2%	8	95.0%	30.7%	11	100%	42.2%
b	36.3%	3.84%	5	78.8%	19.2%	14	98.8%	53.8%	18	100%	69.1%
c	52.5%	3.84%	5	71.3%	19.2%	14	93.8%	53.8%	18	98.8%	69.1%
d	21.3%	3.84%	8	91.3%	30.7%	14	98.8%	53.8%	17	100%	65.3%
e	22.5%	3.84%	6	70.0%	23.0%	14	98.8%	53.8%	17	98.8%	65.3%
f	27.5%	3.84%	8	91.3%	30.7%	14	98.8%	53.8%	20	100%	76.8%
g	25.0%	3.84%	8	88.8%	30.7%	14	98.8%	53.8%	20	100%	76.8%
h	16.3%	3.84%	8	95.0%	30.7%	14	100%	53.8%	19	100%	73.0%
i	21.3%	3.84%	6	85.0%	23.0%	11	100%	42.2%	14	100%	53.8%
j	20.0%	3.84%	8	83.8%	30.7%	13	98.8%	49.9%	17	100%	65.3%
k	22.5%	3.84%	7	88.8%	26.9%	11	98.8%	42.2%	14	100%	53.8%
l	42.5%	3.84%	5	85.0%	19.2%	9	100%	34.6%	12	100%	46.1%
m	50.0%	3.84%	4	80.0%	15.4%	11	98.8%	42.2%	15	100%	57.6%
n	31.3%	3.84%	5	77.5%	19.2%	13	97.5%	49.9%	17	100%	65.3%
o	41.3%	3.84%	5	88.8%	19.2%	8	98.8%	30.7%	11	100%	42.2%
p	47.5%	3.84%	3	81.3%	11.5%	7	98.8%	26.9%	8	100%	30.7%
q	30.0%	3.84%	4	70.0%	15.4%	8	90.0%	30.7%	11	98.8%	42.2%
r	40.0%	3.84%	6	80.0%	23.0%	14	95.0%	53.8%	20	97.5%	76.8%
s	28.8%	3.84%	8	76.3%	30.7%	11	95.0%	42.2%	14	100%	53.8%
t	30.0%	3.84%	6	86.3%	23.0%	14	97.5%	53.8%	20	100%	76.8%
u	51.3%	3.84%	6	97.5%	23.0%	13	100%	49.9%	17	100%	65.3%
v	45.0%	3.84%	6	83.8%	23.0%	12	98.8%	46.1%	19	100%	73.0%
w	31.3%	3.84%	6	72.5%	23.0%	11	95.0%	42.2%	14	100%	53.8%
x	41.3%	3.84%	6	92.5%	23.0%	11	100%	42.2%	15	100%	57.6%
y	27.5%	3.84%	6	81.3%	23.0%	12	98.8%	46.1%	19	100%	73.0%
z	77.5%	3.84%	4	98.8%	15.4%	9	100%	34.6%	12	100%	46.1%
Avg.	36.2%	3.84%	5.9	83.6%	22.7%	11.7	97.9%	44.9%	15.7	99.8%	60.3%

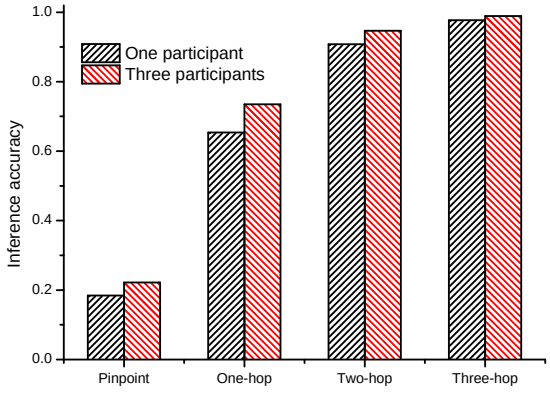


Fig. 10. Impact of the number of participants.

of VISIBLE for each key ranges from 16.3% for letter "h" to 77.5% for letter "z". The average pinpoint accuracy of VISIBLE across all 26 letters is 36.2%, which is almost an order of magnitude higher than 3.84% of random guess. Since different keys' h-hop neighborhoods have different sizes for the same h, we calculate each P_i^h for random guess based on the actual number of keys within key i's h-hop neighborhood (i.e., $|\Omega^h(i)|$) to ensure fair comparisons. We can see that for both VISIBLE and random guess, the P_i^h for each key i increases as h increases, which is expected, as the larger the neighborhood being considered, the higher the probability that a key inferred as some key in the neighborhood, and vice versa. Moreover, the $P^h(i)$ of VISIBLE is always significantly higher than the corresponding $P^h(i)$ of random guess. We also calculate the average P_i^1, P_i^2, and P_i^3 of VISIBLE across all 26 keys as 83.6%, 97.9%, and 99.8%, respectively, which are much higher than corresponding 22.7%, 44.9%, and 60.3% of random guess. Meanwhile, note that the average P_i^h may only be used with caution to compare the performance of two different techniques, due to the difference in the size of keys' neighborhood.

We also notice that pinpoint and h-hop accuracies of the letters at corner positions (i.e., "q", "z", "p", and "m") are higher than those of the letters at the center (e.g., "g" and "h"). This is because typing the letters at corner positions causes more distinguishable motion patterns than those at the center. Moreover, we can see that the pinpoint and h-hop accuracies of letter "z" are much higher than those of other three letters at the corner positions. The reason behind such disparity is that our selected AOIs are not evenly distributed. As shown in Fig. 4(b), the distances between letter "z" and selected AOIs are greater than those of other letters, and typing "z" thus causes more distinguishable motion patterns.

Fig. 9 shows the impact of the training set size on the inference accuracy. As expected, increasing the training set size can slightly improve the key inference accuracy. Fig. 10 shows the impact of the number of participants in the training set. We can see that as the number of participants increases, the key inference accuracy slightly increases in all the cases. In addition, a small number of participants is sufficient for

TABLE II. LIST OF WORDS USED TO TEST THE ATTACK.

Word	Length	Word	Length	Word	Length	Word	Length
paediatrician	13	pomegranate	11	unphysical	10	platinum	8
interceptions	13	feasibility	11	institute	9	homeland	8
abbreviations	13	polytechnic	11	extremely	9	security	8
impersonating	13	obfuscating	11	sacrament	9	between	7
soulsearching	13	difference	10	dangerous	9	spanish	7
hydromagnetic	13	wristwatch	10	identity	8	nuclear	7
inquisition	11	processing	10	emirates	8		

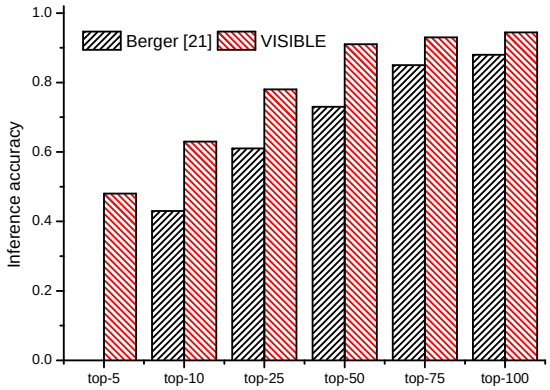


Fig. 11. Word inference accuracy.

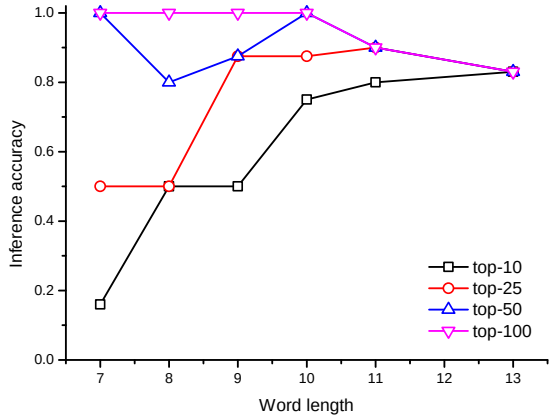


Fig. 12. Word inference accuracy vs. the word length.

achieving acceptable key inference accuracy, which means that VISIBLE requires very few attackers to launch.

C. Word Inference Experiment

We now report the experimental results of the word inference attack on the iPad 2 tablet. In this experiment, we involved two participants and let each participant enter each word in Table II (which is also used in [21]) once to evaluate the word inference accuracy. In total, we collected 2×27 words with 7∼13 letters, where all the letters in the words are in lower-case. As in [21], we used the "corn-cob" dictionary [35] consisting of more than $58,000$ English words. For each letter in each word, we first used the trained multi-class SVM classifier to predict one key and obtained a candidate key set consisting of all the keys that are less than two hops from the predicated key. Then for each word, we obtained a candidate word list by filtering out the combinations that are not in the "corn-cob" dictionary [35].

Fig. 11 compares the overall word inference accuracy of VISIBLE and the technique proposed in [21] for the tested words in Table II. To enable direct comparison, we view the size of the candidate word list output by VISIBLE as the lowest possible rank of the correct word in the candidate word list if the correct word is in candidate word list. In other words, if a candidate word list of c words contains the correct word typed by the victim, then we say the correct word is among the top-k candidate words for any $k \geq c$. As shown in Fig. 11, the correct word is among the top-5 candidate words output by VISIBLE 48% of the time, which means that nearly half of the words in Table II have a candidate word list with no more than 5 words. Besides, we can see that the correct word is among top-10, top-25, and top-50 candidate words with probabilities 63%, 78%, and 93%, respectively. In contrast, the technique in [21] infers the correct word in top-10, top-25, and top-50 candidate words with probabilities 43%, 61%, and 73%, respectively. VISIBLE thus achieves much higher accuracy for word inference than [21].

Fig. 12 shows that word inference accuracy increases as the word length increases. Two reasons account for this trend. First, a longer word has more constraints in letter combinations and thus fewer candidate words in the dictionary. Second, according to statistics of English words, the number of words of length seven is the largest among all words, so words with seven letters have the most candidate words in the dictionary, which leads to a lower inference accuracy.

D. Sentence Inference Experiment

Next, we report VISIBLE's performance for inferring complete sentences on the iPad 2 tablet. For this experiment, we used Enron Email Dataset [37]–[39], which comprises over 600,000 emails generated by 158 employees of the Enron Corporation and is also used in previous work [40] to test the sentence inference accuracy. We asked one participant to select two sentences from the dataset and enter the selected sentences using the alphabetical keyboard of the iPad 2 tablet. The attacker video-recorded the sentence-entry process using two camcorders and used a multi-class SVM classifier trained by the keystroke data. The attacker then performed word inference to obtain a candidate word list for each word and finally chose one word from each candidate word list to form a meaningful sentence based on the linguistic relationship between adjacent words.

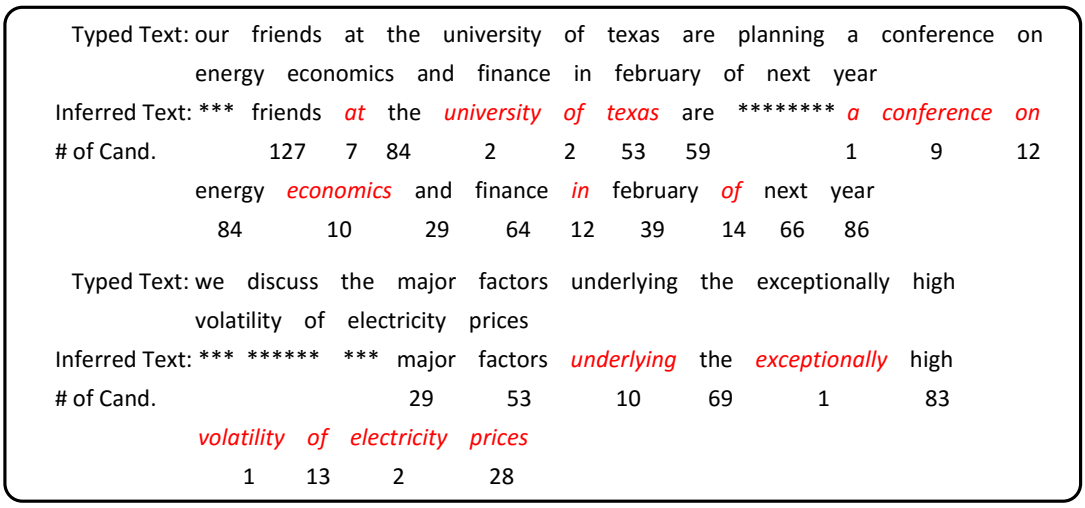


Fig. 13. Sentence inference results.

Fig. 13 illustrates the input sentences and the results inferred by VISIBLE. The number under each word is the number of candidate words output by VISIBLE. The red italic words are the ones correctly chosen by the attacker. The black non-italic words are the ones in the candidate word list but hard to choose. Symbol "*" indicates that the corresponding word is not correctly inferred during word inference. More detailed investigations find that the incorrectly inferred words are due to one or two misclassified letters. We expect that the sentence inference accuracy of VISIBLE can be dramatically improved by incorporating more advanced linguistic models.

E. PIN Keyboard Experiment

We now evaluate the key inference performance on the PIN keyboard of the iPad 2 tablet. In this experiment, we involved three participants. Intuitively, the key inference on the PIN keyboard is more difficult than that on the alphabetical keyboard for mainly two reasons. First, all the keys are located in a relatively small area in the central part of the touchscreen. Second, the typed keys are very likely to be random, and there are no relationships (e.g., linguistic relationship) between adjacent keystrokes.

Table III compares the pinpoint and 1-hop accuracies of VISIBLE and random guess for each key on the PIN keyboard. We can see that the pinpoint accuracy of each key ranges from 21% for number "9" to 61% for "c" cancel key. The average pinpoint accuracy of VISIBLE across all 26 letters is 38%, which is more than four times of that of random guess, i.e., $\frac{100}{11} = 9\%$. When considering one-hop neighborhood, the P_i^1 of VISIBLE for each key is still much higher than that of random guess. We can see that the average P_i^1 of VISIBLE across all 11 keys is 68%, which is much higher than 36% of random guess. Again, the average P_i^1 should be only used with caution to compare the inference accuracies of two techniques.

Moreover, comparing Tables I and III, we can see that although the PIN keyboard has fewer keys than the alphabetical keyboard, the key inference accuracy of the PIN keyboard is not dramatically higher than that of the alphabetical keyboard. The reason is that the keys of the PIN keyboard reside in a relatively small area in the central part of the touchscreen, and the motion patterns caused by different keys are not so distinguishable. In contrast, even though the alphabetical keyboard has more keys, they are located in a relatively large area from the left side to the right side of the touchscreen. The motion patterns of different keys, especially distant ones, cause more distinguishable motion patterns.

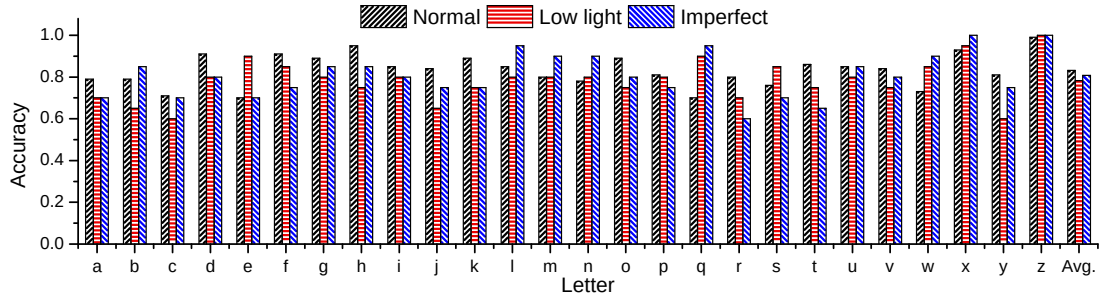
TABLE III. KEY INFERENCE RESULTS FOR PIN KEYBOARD.

Key	P_i		$\Omega^1(i)$	$\lvert\Omega^1(i)\rvert$	P_i^1	
	VIS	RG			VIS	RG
1	21%	9%	1, 2, 4	3	58%	27%
2	25%	9%	1, 2, 3, 5	4	75%	36%
3	45%	9%	2, 3, 6	3	63%	27%
4	55%	9%	1, 4, 5, 7	4	81%	36%
5	40%	9%	2, 4, 5, 6, 8	5	66%	45%
6	35%	9%	3, 5, 6, 9	4	64%	36%
7	44%	9%	4, 7, 8	3	73%	27%
8	23%	9%	5, 7, 8, 9, 0	5	53%	45%
9	27%	9%	6, 8, 9, c	4	61%	36%
0	47%	9%	0, 8, c	3	72%	27%
c	61%	9%	0, 9, c	3	80%	27%
Avg.	38%	9%	-	4	68%	36%

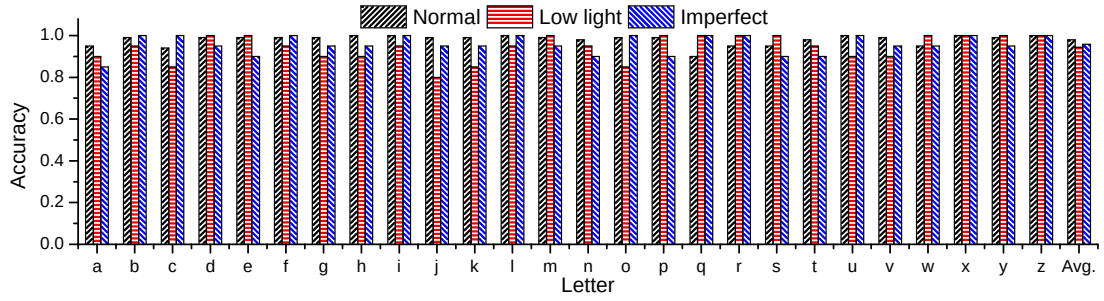
F. Impact of Environmental Factors

We also evaluate the impact of a number of environmental factors on the performance of VISIBLE.

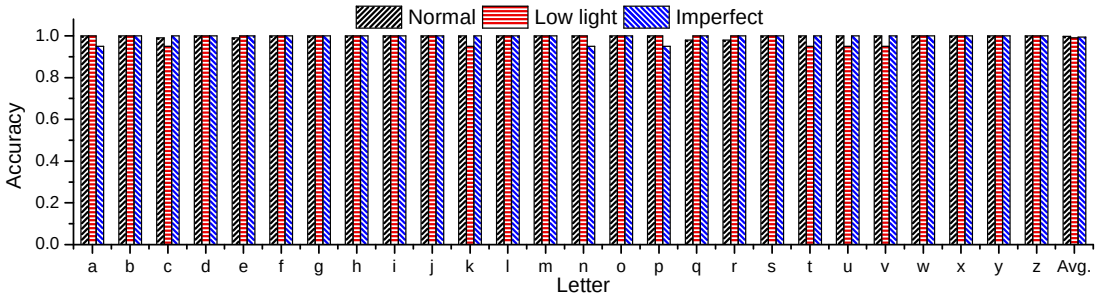
a). Different Light Conditions. Our attack relies on analyzing the video recordings of the tablet backside during the victim's typing process, while the video quality is affected by light conditions. In general, the low-light condition will lead to increased video noise, streaking, blurred motion, and poor focus. We did key inference experiments under light conditions of 400 lux (normal) and 180 lux (low light). Fig. 14 shows the key inference results for each key. We can see that the key inference accuracy decreases slightly as the light condition changes from 400 lux to 180 lux, which is expected. However, the key inference result under 180 lux is still quite acceptable,



(a) One-hop accuracy.



(b) Two-hop accuracy.



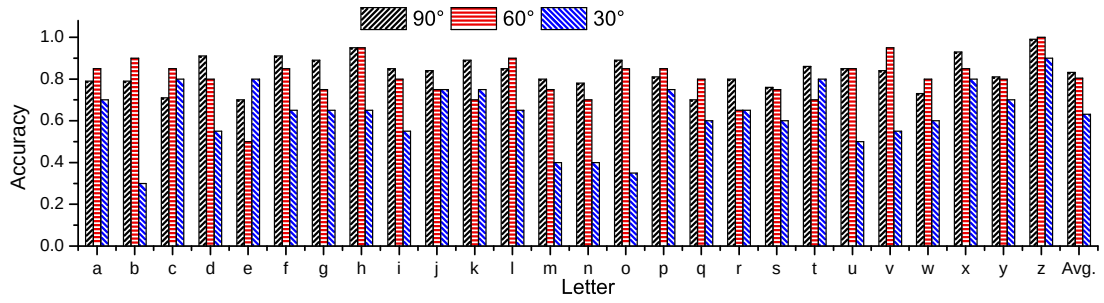
(c) Three-hop accuracy.

Fig. 14. Alphabetical keyboard inference accuracy under different light conditions and imperfect reconstruction of the attack scenario.

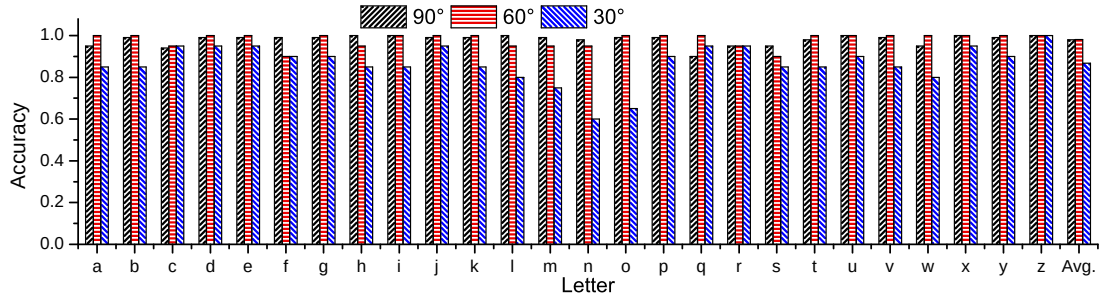
which highlights the wide applicability of VISIBLE in low-light conditions.

b). Different angles between camcorders and the tablet. The performance of VISIBLE is also affected by the angles between the camcorders and the tablet. In previous experiments, the angle between the camcorders and tablet was 90 degree. We changed the angle to 60 and 30 degrees while keeping the distance between the camcorders and tablet unchanged. The experimental result is shown in Fig. 15 for each key's inference accuracy by considering one-hop, two-hop, and three-hop neighbors. We can see that in each of three subfigures, 90 and 60 degree angles lead to similar key inference accuracy which is nevertheless better than that of the 30 degree angle. The reason is as follows. Each camcorder has a specific Depth of Field (DOF) that is the distance between the nearest and farthest objects in a scene that appear acceptably sharp in an image. If the angle between the camcorders and the tablet is 90 or 60 degree, all the AOIs are in the DOF of the camcorders so their motions can be clearly recorded. However, if the angle between the camcorders and the tablet is too small, the camcorders cannot contain all the AOIs in their DOF, which leads to blurred AOI images and thus inaccurate estimation of tablet backside motions. If the angle has to be small due to practical constraints, the attacker can use multiple camcorders to record the motions of different AOIs to obtain sharp image of each AOI.

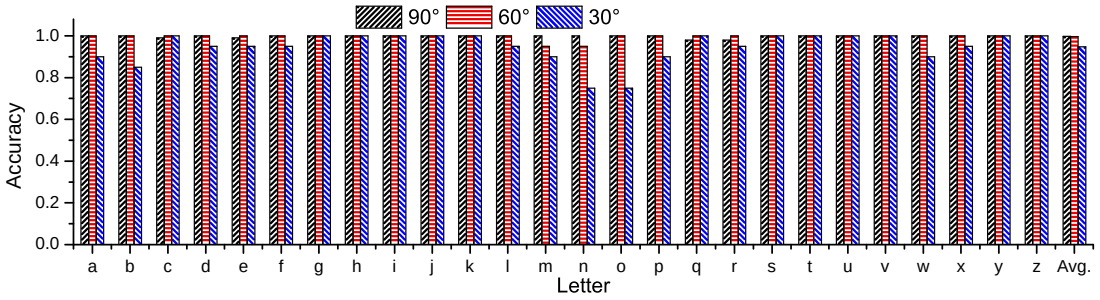
c). Imperfect reconstruction of the attack scenario. As mentioned in Section V-B, to launch a successful key inference attack, the attacker needs to reconstruct the attack scenario based on recorded images. However, the reconstructed layout cannot be exactly the same as the true layout. We therefore evaluate the impact of imperfect reconstruction of the attack scenario. For this experiment, we changed the location of the camcorders randomly by five centimeters and the position of the tablet by three centimeters and then redid the key inference experiment. Fig. 14 shows the key inference accuracy when the attack scenario is not perfectly reconstructed. We can see that the key inference accuracy for each key is only slightly lower than that under perfect reconstruction, which shows the robustness of VISIBLE against small environment change. Note that attack scenario reconstruction is under the full control of the attacker and does not involve the victim. Its accuracy depends only on the quality of the recorded images, and we expect the reconstructed attacker scenario to be accurate in practice.



(a) One-hop accuracy.



(b) Two-hop accuracy.



(c) Three-hop accuracy.

Fig. 15. Alphabetical keyboard inference accuracy for different angles between the tablet and camcorders.

On the other hand, if the environment changes significantly during video recording, e.g., the victim changes position or moves the tablet for more than 10 centimeters, the attacker may need to launch a new round of attack to obtain accurate inference result.

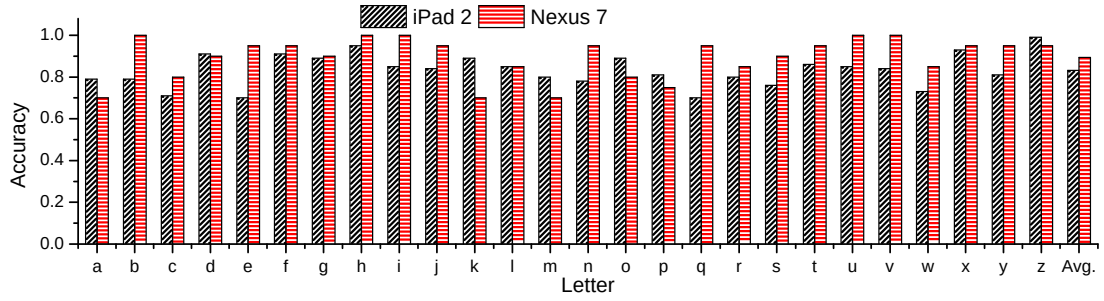
G. Experiments on a Google Nexus 7 Tablet

To demonstrate the universality of VISIBLE, we also did experiments on a Google Nexus 7 tablet with a 7-inch touchscreen which is smaller than that of an iPad 2. Backside motion estimation on Nexus 7 is easier than that on an iPad 2 tablet for two reasons. First, the size of Nexus 7 is smaller than that of iPad 2, so we were able to video-record the clear tablet backside motion with only one camcorder. Second, the Nexus 7's backside has more texture information (e.g., logo and dots) which enables motion estimation at more parts of the tablet backside.

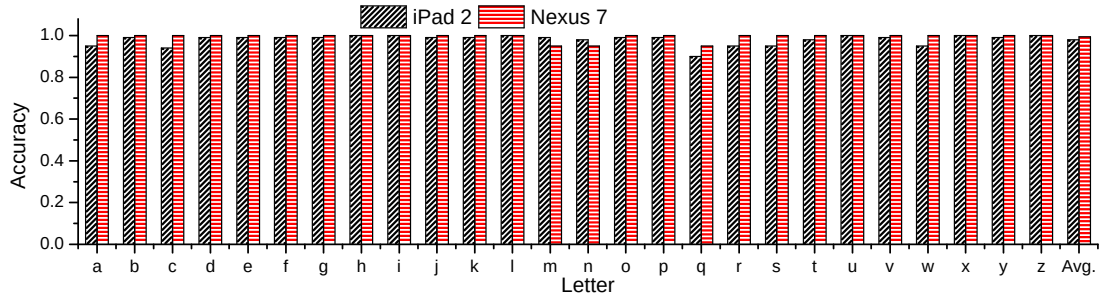
Fig. 16 compares the performance of VISIBLE on a Google Nexus 7 tablet with Android 4.4 with that on an iPad 2 tablet with iOS 8. It is easy to see that the key inference accuracy of VISIBLE is similar on both tablets. This means that VISIBLE is applicable to smaller-size tablets as long as there are sufficient areas with texture information on the tablet backside, which holds for almost all tablets. Besides, we can find that the performance on Nexus 7 is slightly better than that on iPad 2. The reason is that the Nexus 7's backside has more texture information for the attacker to extract motion information, while the iPad 2's backside has less texture information (as shown in Fig. 4(a)). As mentioned in Section V-C, AOIs near the edges of the tablet backside and separated from each other tend to have larger and more distinctive motions than others, making them more capable of differentiating the motion patterns caused by different keystrokes. Therefore, VISIBLE performs better on the tablets with rich texture information on their backsides.

VII. Conclusion, Countermeasures, and Future Work

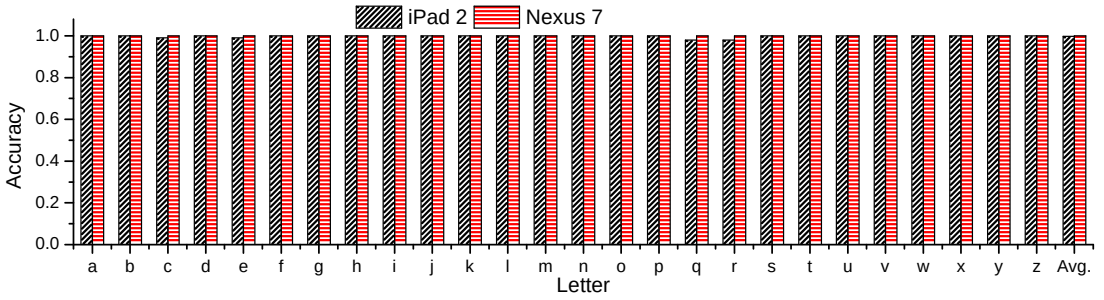
In this paper, we proposed VISIBLE, a video-assisted key inference attack framework to infer the victim's typing content based on the video recordings of the tablet backside. We adopted complex steerable pyramid decomposition to obtain the subtle motions on the tablet backside and used machine



(a) One-hop accuracy.



(b) Two-hop accuracy.



(c) Three-hop accuracy.

Fig. 16. Alphabetical keyboard inference accuracy on a Google Nexus 7 tablet and an iPad 2 tablet.

learning techniques to infer the typed keys, words, and sentences. We thoroughly evaluated the performance of VISIBLE via extensive experiments. Our results show that VISIBLE can achieve high key inference accuracy for both PIN and alphabetical soft keyboards and correctly infer the victim's typed words or sentences with very high probability.

There are several possible countermeasures against VISIBLE. The most straightforward defense is to design a large featureless cover to cover the stand or the tablet to prevent the attacker from finding useful AOIs in the recorded video. The second possible defense is to randomize the layouts of the PIN and alphabetical soft keyboards, such that the attacker cannot recover the typed keys even if he can infer the keystroke positions on the touchscreen. This defense may be effective, but it sacrifices the user experience, as the user needs to find every key on a random keyboard layout during every key-typing process. Another possible defense is to have on-board vibrators generate vibrations during the typing process to mask the motions caused by the user's typing process. However, unlike smartphones, most current commercial off-the-shelf tablets are not equipped with on-board vibrators. The ultimate solution is to cover the whole tablet (both the front and back sides). Though most effective, this solution is inconvenient and might be socially awkward. The investigation of these defenses is left as future work.

To the best of our knowledge, VISIBLE is the first attempt to utilize the backside motion of tablets for keystroke analysis. There are still many issues worth investigation in addition to the countermeasures above. As we mentioned in Section V-B, higher resolutions can help us video-record more texture information details on the tablet backside, and higher frame rates could video-record more motion details overtime. We plan to test VISIBLE with more advanced camcorders with higher resolutions and frame rates. We also seek to investigate the impact of optical and digital zoom and the distance between camcorder and the victim's tablet. In this paper, we only consider the lower-case letters in the English alphabet. In practice, more contents such as upper-case letters, punctuation, characters, and key combinations might be typed by the victim. Further studies on this issue are challenging but meaningful. Our current study assumes that the victim places the tablet with a holder on a desk, while another common scenario is to hold the tablet by hand. In this case, the motion of the tablet backside is the combination of the motions of the holding hand

and the non-holding hand's keystrokes. Keystroke inference in this scenario is much more challenging because we need to cancel the time-varying motion of the holding hand. In VISIBLE, the attacker needs to reconstruct the attack scenario to obtain a training data set to infer the victim's typed inputs. Although feasible, it is not so convenient. A more attractive way is to build a unified and normalized model, which could automatically transfer motions video-recorded in different distances and angles to a unified and normalized distance and angle. This will greatly improve the convenience of launching our proposed attack and deserves further investigations.

ACKNOWLEDGMENT

We would also like to thank our shepherd, Patrick Traynor, and the anonymous NDSS reviewers for insightful comments. This work was partially supported by the US National Science Foundation under grants CNS-1514381, CNS-1513141, CNS-1421999, CNS-1514014, and CNS-1422301.

REFERENCES

[1] "Gartner: Device shipments break 2.4b units in 2014, tablets to overtake pc sales in 2015," http://techcrunch.com/2014/07/06/gartner-device-shipments-break-2-4b-units-in-2014-tablets-to-overtake-pc-sales-in-2015/.

[2] M. Shahzad, A. Liu, and A. Samuel, "Secure unlocking of mobile touch screen devices by simple gestures: You can see it but you can not do it," in *ACM MobiCom'13*, Miami, FL, Sep. 2013, pp. 39–50.

[3] L. Li, X. Zhao, and G. Xue, "Unobservable re-authentication for smartphones," in *NDSS'13*, San Diego, CA, Feb. 2013.

[4] J. Sun, R. Zhang, J. Zhang, and Y. Zhang, "Touchin: Sightless two-factor authentication on multi-touch mobile devices," in *IEEE CNS'14*, San Francisco, CA, Oct. 2014.

[5] Y. Chen, J. Sun, R. Zhang, and Y. Zhang, "Your song your way: Rhythm-based two-factor authentication for multi-touch mobile devices," in *IEEE INFOCOM'15*, Hongkong, China, Apr. 2015.

[6] M. Backes, M. Dürmuth, and D. Unruh, "Compromising reflections-or-how to read lcd monitors around the corner," in *IEEE S&P'08*, Oakland, CA, May 2008.

[7] M. Backes, T. Chen, M. Duermuth, H. Lensch, and M. Welk, "Tempest in a teapot: Compromising reflections revisited," in *IEEE S&P'09*, Oakland, CA, May 2009.

[8] D. Balzarotti, M. Cova, and G. Vigna, "Clearshot: Eavesdropping on keyboard input from video," in *IEEE S&P'08*, Oakland, CA, May 2008.

[9] F. Maggi, A. Volpatto, S. Gasparini, G. Boracchi, and S. Zanero, "A fast eavesdropping attack against touchscreens," in *IAS'11*, Melaka, Malaysia, Dec. 2011.

[10] R. Raguram, A. White, D. Goswami, F. Monrose, and J.-M. Frahm, "ispy: Automatic reconstruction of typed input from compromising reflections," in *AMC CCS'11*, Chicago, IL, Oct. 2011.

[11] Y. Xu, J. Heinly, A. White, F. Monrose, and J.-M. Frahm, "Seeing double: Reconstructing obscured typed input from repeated compromising reflections," in *AMC CCS'13*, Berlin, Germany, Oct. 2013.

[12] Q. Yue, Z. Ling, X. Fu, B. Liu, K. Ren, and W. Zhao, "Blind recognition of touched keys on mobile devices," in *AMC CCS'14*, Scottsdale, AZ, Nov. 2014.

[13] D. Shukla, R. Kumar, A. Serwadda, and V. Phoha, "Beware, your hands reveal your secrets!" in *AMC CCS'14*, Scottsdale, AZ, Nov. 2014.

[14] L. Cai and H. Chen, "Touchlogger: Inferring keystrokes on touch screen from smartphone motion," in *USENIX HotSec'11*, Berkeley, CA, Nov. 2011.

[15] E. Owusu, J. Han, S. Das, A. Perrig, and J. Zhang, "Accessory: Password inference using accelerometers on smartphones," in *ACM HotMobile'12*, San Diego, CA, Feb. 2012.

[16] E. Miluzzo, A. Varshavsky, S. Balakrishnan, and R. Choudhury, "Tapprints: your finger taps have fingerprints," in *ACM MobiSys'12*, Low Wood Bay, Lake District, UK, June 2012, pp. 323–336.

[17] Z. Xu, K. Bai, and S. Zhu, "Taplogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors," in *ACM WiSec'12*, Tucson, AZ, Feb. 2012.

[18] L. Simon and R. Anderson, "Pin skimmer: Inferring pins through the camera and microphone," in *SPSM'13*, Berlin, Germany, Nov. 2013.

[19] S. Narain, A. Sanatinia, and G. Noubir, "Single-stroke language-agnostic keylogging using stereo-microphones and domain specific machine learning," in *ACM WiSec'14*, Oxford, United Kingdom, Jul. 2014.

[20] L. Zhuang, F. Zhou, and J. Tygar, "Keyboard acoustic emanations revisited," in *ACM CCS'05*, Alexandria, VA, Nov. 2005.

[21] Y. Berger, A. Wool, and A. Yeredor, "Dictionary attacks using keyboard acoustic emanations," in *ACM CCS'06*, Alexandria, VA, Nov. 2006.

[22] T. Zhu, Q. Ma, S. Zhang, and Y. Liu, "Context-free attacks using keyboard acoustic emanations," in *ACM CCS'14*, Scottsdale, AZ, Nov. 2014.

[23] P. Marquardt, A. Verma, H. Carter, and P. Traynor, "(sp)iphone: Decoding vibrations from nearby keyboards using mobile phone accelerometers," in *ACM CCS'11*, Chicago, IL, Nov. 2011.

[24] D. Fleet and A. Jepson, "Computation of component image velocity from local phase information," *Int. J. Comput. Vision*, vol. 5, no. 1, pp. 77–104, Aug. 1990.

[25] T. Gautama and M. V. Hulle, "A phase-based approch to the estimation of the optical flow field using spatial filtering," *IEEE Trans. Neural Netw.*, vol. 13, no. 5, pp. 1127–1136, Sep. 2002.

[26] E. Simoncelli and W. Freeman, "The steerable pyramid: A flexible architecture for multi-scale derivative computation," in *ICIP'95*, Washington, DC, Oct. 1995.

[27] E. Adelson, C. Anderson, J. Bergen, P. Burt, and J. Ogden, "Pyramid methods in image processing," *RCA Engineer*, vol. 29, no. 6, pp. 33–41, 1984.

[28] J. Portilla and E. Simoncelli, "A parametric texture model based on joint statistics of complex wavelet coefficients," *Int. J. Comput. Vision*, vol. 40, no. 1, pp. 49–70, Oct. 2000.

[29] A. Torralba and A. Oliva, "Depth estimation from image structure," *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 1226–1238, Sep. 2002.

[30] R. Bergman, H. Nachlieli, and G. Ruckenstein, "Detection of textured areas in images using a disorganization indicator based on component counts," 2007, HP Laboratories Israel Technical Report.

[31] N. Wadhwa, M. Rubinstein, F. Durand, and W. T. Freeman, "Phase-based video motion processing," *ACM Trans. Graph*, vol. 32, no. 4, pp. 80:1–80:10, Jul. 2013.

[32] A. Davis, M. Rubinstein, N. Wadhwa, G. Mysore, F. Durand, and W. Freeman, "The visual microphone: Passive recovery of sound from video," *ACM Trans. Graph*, vol. 33, no. 4, pp. 79:1–79:10, Jul. 2014.

[33] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[34] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten, "The weka data mining software: An update," *SIGKDD Explor*, vol. 11, no. 1, pp. 10–18, Nov. 2009.

[35] "corn-cob dictionary," http://www.mieliestronk.com/wordlist.html.

[36] P. Brown, P. deSouza, R. Mercer, V. Pietra, and J. Lai, "Class-based n-gram models of natural language," *Comput. Linguist.*, vol. 18, no. 4, pp. 467–479, Dec. 1992.

[37] B. Klimt and Y. Yang, "The enron corpus: A new dataset for email classification research," in *Machine learning: ECML 2004*. Springer, 2004, pp. 217–226.

[38] "Enron email dataset," https://www.cs.cmu.edu/ ./enron/.

[39] "Parakweet lab's email intent data set," https://github.com/ParakweetLabs/EmailIntentDataSet.

[40] D. Ping, X. Sun, and B. Mao, "Textlogger: Inferring longer inputs on touch screen using motion sensors," in *ACM WiSec'15*, New York, NY, Jun. 2015.