GazeRevealer: Inferring Password Using Smartphone Front Camera

Yao Wang
Northwestern Polytechnical University
wangyao@mail.nwpu.edu.cn

Wandong Cai
Northwestern Polytechnical University
caiwd@nwpu.edu.cn

Tao Gu
RMIT University
tao.gu@rmit.edu.au

Wei Shao
RMIT University
wei.shao@rmit.edu.au

Ibrahim Khalil
RMIT Uniersity
ibrahim.khalil@rmit.edu.au

Xianghua Xu
Hangzhou Dianzi University
xhxu@hdu.edu.cn

ABSTRACT

The widespread use of smartphones has brought great convenience to our daily lives, while at the same time we have been increasingly exposed to security threats. Keystroke security is an essential element in user privacy protection. In this paper, we present GazeRevealer, a novel side-channel based keystroke inference framework to infer sensitive inputs on smartphone from video recordings of victim's eye patterns captured from smartphone front camera. We observe that eye movements typically follow the keystrokes typing on the number-only soft keyboard during password input. By exploiting eye patterns, we are able to infer the passwords being entered. We propose a novel algorithm to extract sensitive eye pattern images from video streams, and classify different eye patterns with Support Vector Classification. We also propose a novel enhanced method to boost the inference accuracy. Compared with prior keystroke detection approaches, GazeRevealer does not require any external auxiliary devices, and it relies only on smartphone front camera. We evaluate the performance of GazeRevealer with three different types of smartphones, and the result shows that GazeRevealer achieves 77.43% detection accuracy for a single key number and 83.33% inference rate for the 6-digit password in the ideal case.

CCS CONCEPTS

• **Security and privacy → Privacy protections**; • **Computing methodologies → Machine learning approaches**;

KEYWORDS

Password Inference, Gaze Estimation, Mobile Security

ACM Reference Format:
Yao Wang, Wandong Cai, Tao Gu, Wei Shao, Ibrahim Khalil, and Xianghua Xu. 2018. GazeRevealer: Inferring Password Using Smartphone Front Camera. In *EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous '18), November 5–7, 2018, New York, NY, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3286978.3287026

MobiQuitous '18, November 5–7, 2018, New York, NY, USA

ACM ISBN 978-1-4503-6093-7/18/11…$15.00
https://doi.org/10.1145/3286978.3287026

1 INTRODUCTION

Mobile payment has become a prevalent mode for online transaction and personal financial management. Various security risks arise in our daily life from the rapid development of mobile and ubiquitous computing applications. Among them, keyboard privacy presents the fundamental risk in mobile payment. The mobile payment system typically requires users to complete privacy-sensitive input with keyboard on their mobile devices such as bank card number, security code, and password. As a result, attackers can typically launch keystroke eavesdropping to reveal personal information from mobile users.

Leveraging side-channel attacks, keystrokes on traditional physical keyboards can be inferred through Trojan applications (a.k.a. keyloggers). Typical approaches include electromagnetic emanation based [3], acoustics signal based [26, 27], and video based [4]. However, in mobile scenarios, user interaction with smartphones has been changed. The popularity of virtual soft keyboard on smartphones eliminates the side-channel emanations (i.e., electromagnetic and acoustic signals) from physical keyboard. Therefore, attackers cannot leverage these signals to deduce keystrokes anymore. Besides, app permission restriction policies in smartphone operating systems restrain apps from intercepting keystrokes. As a consequence, Trojan apps cannot be directly launched to log keystrokes. Traditional approaches thereby face increasing challenges with smartphones. Recently, several smartphone keystroke inference attack approaches have been proposed. They essentially resemble the traditional approach, such as adopting WiFi signals [12], and do require peripheral camera equipment [21]. All of the above approaches need an external data receiver which should be placed close enough to the victim. Therefore, attackers start to pay their attentions to smartphone embedded sensors. For example, several works [5, 15, 17] show that keystrokes can be inferred in a stealthy manner with only a few benign permissions by using accelerometers, gyroscopes, and even audio sensors.

In our work, we present GazeRevealer, a new avenue for attackers to infer password keystrokes with a number-only soft keyboard on the touch screen of smartphone. GazeRevealer analyzes victim's eye patterns recorded from the front camera during password input. Our motivation derives from the obversation that the password input behavior on smartphone always involves eyes and fingers

coordinated motion, i.e., the finger always taps the key number that her/is eyes are staring at. Therefore, eye patterns can reflect the different number keystrokes on the numeric soft keyboard. In comparison with prior sensor-based attacks, GazeRevealer neither requires the victim to distinctly vibrate the phone nor relies on keypad tones during the input process, leading to a more stealthy and imperceptible solution.

The design of GazeRevealer presents three major challenges. 1) Our inference method is mainly based on the analysis of eye contour images. It hence requires a precise and effective method to clip eye image patches from each frame in the video. Since low-resolution smartphone front camera and poor lighting condition may generate low-quality video, traditional methods can only extract coarse and imprecise eye contour images. Parts of the iris and sclera may sometimes be excluded from the extracted patches in such scenario. To obtain fine-grained eye contour images while keeping the image preserving intact iris and sclera information, a precise method is highly desirable. Our investigation shows that the Maximum IsoCenter (MIC) based technique can precisely localize the pupil center even with low-quality images captured by smartphone front camera. Therefore, we can use the center position as the datum point and obtain the fine-grained eye contour image by clipping a certain pixels in its horizontal and vertical direction, respectively. 2) It is not a trivial task to extract keystroke eye images from the stream in order to recognize input passwords on the number-only soft keyboard. Keystroke eye images capture a user's eyes when typing a particular number. In reality, a user may not immediately enter password after the front camera is launched. As a result, there appear some non-sensitive images at the beginning of the video. Additionally, the duration of entering any key number is often short on an order of milliseconds. Therefore, keystroke eye images cannot be selected using standard methods such as fixed time interval [9]. To address this challenge, we adopt similarity which is measured by image histograms as the metric to distinguish keystroke eye images. Since the stream is divided into multiple segments by image similarity, we can pick out images from individual segments as the keystroke eye images. 3) An eye tracking algorithm can only estimate an approximation position of the key tapping. Key numbers on the smartphone soft keyboard are usually very close and dense, hence it is difficult to recognize the keystrokes. We investigate that different keystroke eye images reveal different positions of iris. Based on this observation, we regard pupil center as the center of iris and design an aided inference model to improve recognition accuracy. The model first measures the Euclidean distance between the pupil center of the eye image to be identified and the pupil center of each candidate decided by the eye tracking algorithm. Next, the model calculates a score based on the distance and the probability of each candidate. Eventually, we conclude the candidate with highest score as the key number we predict.

To the best of our knowledge, it is the first work to study the issue of inferring keystroke towards password input from eye pattern video recorded from smartphone front camera. In summary, the paper makes the following contributions.

- We design a novel side-channel attack technique that enables attackers to deduce the key numbers on touch screen tapped by a victim by analyzing the video stream of password entry eye patterns. This technique only requires the smartphone front camera permission which is usually deemed as normal and benign.
- We propose a keystroke eye image extraction algorithm, which uses pupil center position as the datum point to crop fine-grained eye contour images and leverages image similarity principle to determine the keystroke eye images from the image stream.
- We present an enhanced recognition method to gaze estimation, improving the inference accuracy for each key number significantly.
- We collect data from 12 participants in the experiment and evaluate our approach on three commercial off-the-shelf Android smartphones. The result shows that an individual's password can be effectively disclosed at an acceptable recovery rate.

2 BACKGROUND AND RELATED WORK

2.1 Threat Model

The basic assumption of threat lies in installing an untrusted application on smartphone. An increasing number of victims are jeopardized by a wide variety of malicious apps [11] which can be installed in many ways, such as drive-by-download, silent installation, and untrusted third-party market [6]. We therefore believe this assumption does make sense. The victim is also inquired whether to grant front camera access permission to the malicious apps. Notwithstanding potential safety hazards associated with cameras have been proposed in [17], this permission can still be acquired by disguising popular apps that have valid reasons for applying front camera access permission (e.g., selfie apps, mirror apps, and even some game apps). We assume that malicious app is simply run in the background as a standard app instead of being used to compromise any component of the device. We also assume that the malware can listen victim's sensitive events such as password input window and invoke front camera to log eye movements. This functionality may be easily carried out by malicious JavaScript codes [24]. Based on the sensor data conveyed to a remote server, attackers can further infer the entered password.

2.2 Keystroke Recognition

Smartphone embedded sensors provide side-channel attacks a capacious platform that can be used to eavesdrop user's interactions with the smartphone. Cai et al. [5] present an accelerometer based keystroke inference approach to infer the keys being typed on soft keyboard on smartphones. Later, Owusu et al. [15] apply a similar idea to extract victims' 6-character passwords on smartphones by making use of accelerometer readings. Similarly, Xu et al. [25] utilize motion sensors such as accelerometers and gyroscopes to infer user sensitive context on touch screen. Schlegel et al. [17] present an app by exploiting audio sensor to target privacy information. Moreover, Simon et al. [19] collect user touch-event orientation patterns from smartphone microphone and camera to infer PINs entry. Narain et al. [14] propose an architecture to infer key taps by applying a combination of microphone and gyroscope sensors.

The most related work is VISIBLE [21], in which the author also proposes a video-based keystroke inference method on mobile devices. Specifically, VISIBLE relies on the motion patterns of device's backside caused by different taps on the touch screen to infer different keystrokes. However, it needs an unnoticeable external camera to record the backside motions of the device and requires the device to be put at an angle to the table by a holder. In contrast, our approach only requires the malicious app to invoke the front camera to record victim's eye patterns during password entry, which is easy to launch and difficult to perceive.

2.3 Gaze Estimation

The rationale of GazeRevealer leverages on unique gaze patterns to recognize the relevant keystrokes. In the following section, we give a short introduction on the gaze estimation techniques.

Gaze estimation methods can be divided into two categories, shape-based and appearance-based tracking methods. The shape-based method exploits the pupil and Purkinje image features captured by HD cameras with near-infrared lights to infer gaze direction [7]. However, this method requires an unalterable distance between screen and user as well as calibrations for different users. When using low-resolution and low-light images, it is hard to recognize gaze location.

The appearance-based method extracts features from eye image or directly treats the pixels of eye image as the input vectors, and constructs a traditional machine-learning model [9] or a deep learning model [13] to estimate the gaze position on screen, which substantially regards the gaze tracking process as a regression problem. This method offers more feasibility in our situation, because it does not heavily rely on hardware equipment (e.g., no need for a HD camera and an infrared LED light), and free of calibration.

3 THE DESIGN OF GAZEREVEALER

3.1 System Overview

The primary goal of GazeRevealer is to deduce the sensitive information (i.e., password) users input on smartphone. Assuming that an attacker obtains a victim's video stream of eye patterns, GazeRevealer can start to infer keystrokes on the numeric soft keyboard of smartphone. As shown in Fig. 1, we give an overall framework of GazeRevealer which consists of three modules. 1) Keystroke eye image extraction module, which is used to automatically identify the eye images of different keystrokes from an input video stream; 2) Data preprocessing module, which processes the eye images to extract relevant features and reduces the dimension; 3) Keystroke recognition module, which calculates the features of different eye images after dimension reduction and determines the relevant keystrokes.

3.2 Keystroke Eye Image Extraction Module

3.2.1 Eye Detection and Image Normalization. In this stage, GazeRevealer first extracts the Region of Interest (ROI) of eyes from each frame which is extracted from victim's video stream. An example of the eye ROIs detection is presented in Fig. 2. For more accurate detection of eye positions, we first rapidly approximate rough face position from the frame by using a cascade classifier based on Local Binary Patterns (LBP) [1] to narrow down the detection area for

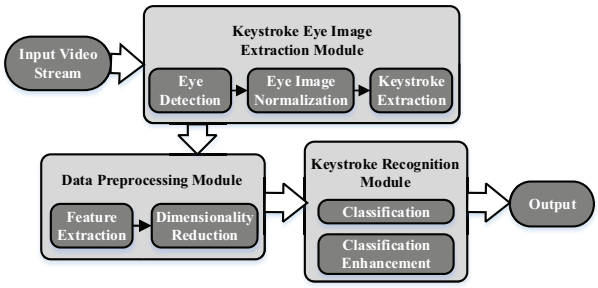
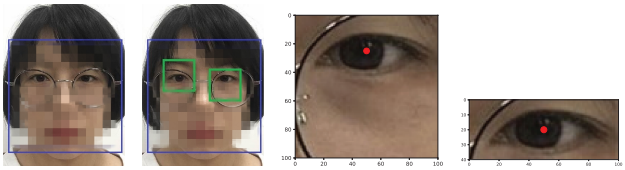


Figure 1: Framework of GazeRevealer



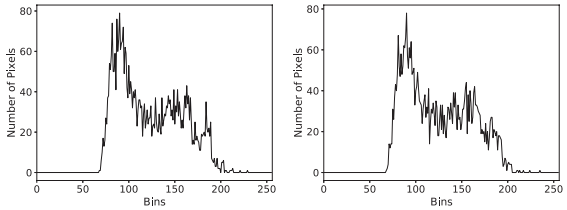
(a) Face position detection (b) Eye position detection (c) Eye ROI and pupil center (d) Final eye image

Figure 2: Eye Image Extraction Process

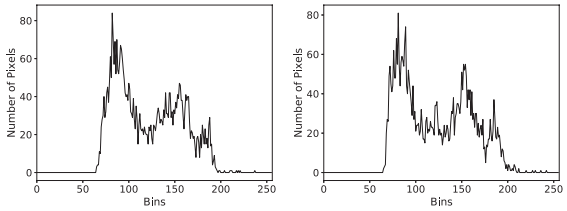
eye-pairs (i.e., the blue bounding rectangle in Fig. 2(a)). Once we procure the facial region, eye ROIs can be extracted as shown in Fig. 2(b), by means of a more precise Haar based classifier [23].

The aforementioned process can only limit the ROI size to fixed 100×100 pixels. The larger detected region may still contain some other ambiguous factors which are not conducive to gaze estimation, such as eyebrows, hairs, and eyeglass frames. Therefore, we require a much tighter region around eyes as our final eye image for further analysis. The most ideal region is to crop a certain quantity of pixels around the pupil center, this can get rid of interference as well as preserve the main information of eyes. Therefore, how to localize pupil center has to be taken into consideration. To solve this problem, we apply the MIC based technique [22] to estimate pupil center which is effective on those low-resolution images, such as captured from smartphone front camera. Considering poor ambient illumination in our situation, pupil center sometimes cannot be effectively detected by only the MIC based method. Therefore, we employ an additional mean shift algorithm to resolve the problem of inaccurate pupil center localization caused by undesirable lighting conditions. In Fig. 2(c), the red dot represents the position where the pupil center is localized. In what follows, the upper and lower area of the pupil center is clipped 20 pixels off, respectively, while the horizontal axis keeps 100 pixels unchanged. As shown in Fig. 2(d), the size of the final eye image is set to 40×100 pixels.

3.2.2 Keystroke Eye Images Extraction. After normalization, we obtain a sequence of fixed-size eye images. The major challenge for keystroke eye image extraction is how to single out the fixation images that can specifically represent the corresponding keystrokes from the sequence. The fixation refers to the visual gaze on a specific key which the victim is entering. In the field of eye tracking, the existing fixation image extraction schemes rely on certain defined



(a) An image in a particular fixation (b) Another image in the same fixation

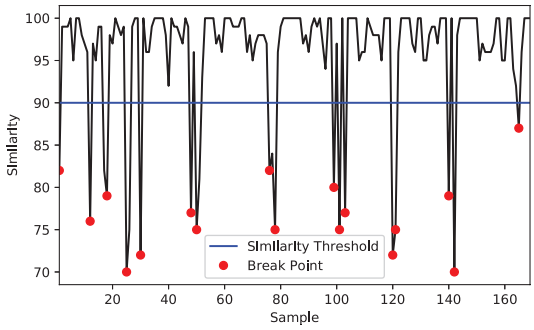


(c) An image in a saccade (d) An image in another particular fixation

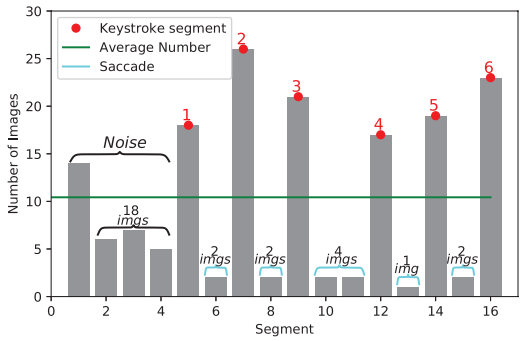
Figure 3: Image Histograms of Different Eye States

regulations. For example in [9], the video is divided into several chunks by fixed time intervals (i.e., predetermined fixed-length fixation time of several seconds), then different fixation images can be picked out from the chunks. This method is not effective in our scenario since fixation for a keystroke performs quickly with an unfixed duration (on an order of milliseconds) and non-keystroke images (imagine that a victim conducts glimpses on the screen instead of immediately watching the soft keyboard to input her/is password after the front camera is launched) at the beginning of our video stream are likely to be divided into chunks. To solve this problem, we propose a novel algorithm to precisely extract keystroke eye images, and this algorithm consists of three steps as follows:

- ***Image Similarity Estimation:*** An eye movement is comprised of fixations and saccades (e.g., a quick eye switch from one key to the next). Namely, two fixations are always separated by a single saccade. If we can determine the intersections of all fixations and saccades, the stream can be split into multiple segments. Then we select the center image of the fixation segment as the feature image for a particular keystroke. Based on this, we introduce image similarity to search the break points in the stream. Fig. 3 illustrates the comparison of histograms of four different eye images. Fig. 3(a) and Fig. 3(b) show the histograms of two eye images which are picked out from one particular fixation. We observe that different images from the same fixation would lead to quite similar histograms. Fig. 3(c) and Fig. 3(d) display the histograms of two eye images which are extracted from a saccade and another fixation, respectively. It can be observed that the two histograms differ from each other and from that in Fig. 3(a) and Fig. 3(b). The comparison motivates us to conclude that eye images in different status would cause intuitive transitions in histograms. We use histogram as the metric to quantize the similarity between two images as



(a) Image stream similarity



(b) Keystroke segment determination

Figure 4: Keystroke Image Extraction

follows:

$$S(h, h') = \frac{1}{n}\sum_{i=1}^{n}\left(1 - \frac{|h_i - h'_i|}{Max(h_i, h'_i)}\right)$$

where $S(h, h')$ (the value lies in $[0, 100]$) denotes the similarity between histograms (h and h') of two images and n is the total number of histogram bins ($n = 256$ in our case). The greater the value is, the higher the similarity would be.

- ***Image Stream Similarity Building:*** By using the above equation, we can calculate the i^{th} similarity between the i^{th} and the $(i+1)^{th}$ image in the stream which can be expressed as $\{S_i(I_i, I_{i+1})\}_{i=1:n-1}$ where n is the number of images in the stream. Given the Frames per Second (FPS) of camera f, and the video time duration t, n can be represented by $f \times t$. As shown in Fig. 4(a), a similarity waveform can be built after calculating the similarities of all images in the stream. We are only interested in the red dots with the similarity which is lower than the predefined threshold (the blue horizontal line), since they are likely to be the break points between fixations and saccades.
- ***Keystroke Image Extraction:*** To extract keystroke images, the essential issue is to determine the fixation segments for individual keystrokes, which should include as many keystroke images as possible while removing the non-keystroke images. As shown in Fig. 4(a), n break points divide the stream into $n-1$ segments. We count the number of images

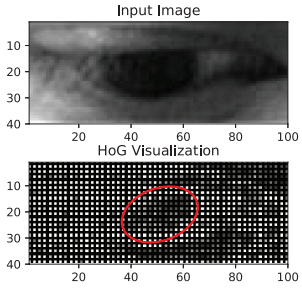


Figure 5: Feature Visualization

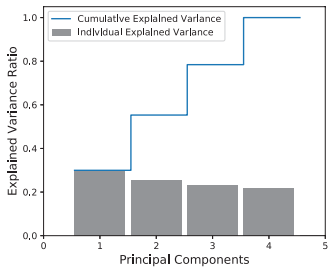


Figure 6: Principal Components Selection

in each segment, and set the average value of the segment images as the key threshold. In the following stage, we consider the segments with the number of images is higher than the threshold, which are likely to be the fixation segments. Since every two fixations are connected by a saccade, as long as the saccade segments are identified, we can determine the fixation segments. According to a study of eye saccades [20], latency of a reflexive saccade (e.g., saccades in password input) is usually less than 250ms, which suggests the number of images in a saccade is usually less than 8 ($\lceil 0.25 \times 30 \rceil$, where 30 is the front camera FPS) in our situation. We calculate the total number of images between the considered segments and check whether the number is lower than 8 (determination of saccades). The segments adjacent to the saccades are regarded as the fixations. Finally, we choose the center image of the fixation segment as the feature image of this keystroke. Fig. 4(b) shows the process of extracting keystroke images.

3.3 Data Preprocessing Module

After eye image data is collected, GazeRevealer conducts feature extraction and dimension reduction.

3.3.1 Feature Extraction. It is important to choose appropriate features to discriminate keystrokes. Since ambient illumination changes may arise in our situation, we select Histogram of Oriented Gradients (HoG) as the descriptor which is invariance to influence of illumination effects [10]. Besides, HoG can distinguish the iris and sclera in low-resolution eye images. In Fig. 5, the HoG features of an eye image are visualized. It is observed that iris region (highlighted by red bounding ellipse) is darker than the surrounding sclera region, which means HoG can effectively represent the eye pattern. Therefore, we choose HoG features for gaze estimation, using the following parameters: 9 orientations, 2×2 pixels per cell, and 2×2 cells per block.

3.3.2 Dimension Reduction. Dimension reduction plays a key role in gaze estimation. HoG features extracted from an eye image would result in high dimension (over 33k) and suffer from noise. In this work, GazeRevealer uses Principal Component Analysis (PCA) to find most correlated variables in the extracted feature space. PCA is expected to reduce the size of the original feature space to a lower dimension while remaining the representative information.

We can then select the most valuable components in the feature space after filtering with PCA. In Fig. 6, we observe that the first 4 components almost retain all variance of the original data. As a result, high dimensional original data can be compressed to 4 dimensions for further gaze estimation in this work.

3.4 Keystroke Recognition Module

GazeRevealer aims to recognize the keystrokes based on the relevant eye images. We show the process in the following two steps.

3.4.1 Classification. Keystroke recognition process is essentially a ten-class classification problem. Based on the features extracted from any eye-pair images, GazeRevealer estimates the corresponding key number. In our experiment, we use Support Vector Classification (SVC) [8] with Radial Basis Function (RBF) kernel, available in the scikit-learn machine learning library [16] as our classifier since it is efficient for classifying non-linear data. Other classifiers (e.g., Gaussian Process Classifier and Random Forest Classifier) are also deployed for comparison, and we confirm that SVC achieves the best performance.

SVC with RBF kernel is constrained by two parameters, C and γ. C trades off misclassification of training data against simplicity of the decision hyperplane. γ defines how far the influence of a sample can reach. The low values of γ means lower bias and higher variance while high values means higher bias and lower variance. The optimal values are selected from a prebuilt set of possible parameters, based on experiments that adopt 5-fold-cross-validation: for each pair of possible parameters, 4/5 of the data is used as training data, while the remaining 1/5 data is used as validation data for evaluating the performance of the classifier, the process is then repeated 5 times. We choose the pair of parameters that performs the best result as the final parameters for SVC. Specifically, in our experiment, the initial set of the parameters contains 6 values which is logarithmically spaced from 10^{-3} to 10^{2}. After testing, the best values for C and γ are 10^{0} and 10^{-1}, respectively.

3.4.2 Classification Enhancement. Although the classification algorithm can differentiate the most probable key number based on the relevant eye image, some unavoidable factors such as distance between eyes and smartphone, size of the screen, resolution of front camera, and even body posture (e.g., standing, sitting, and slouching) are more or less expected to negatively affect the classification results. In order to minimize such influence on key number inference, we propose an enhancement method to the basic classification algorithm to reinforce the inference results.

We first calculate the average pixel location of pupil center for each of the ten keystroke eye images on our dataset, that is $\{L^i_{avg}(x^i_{avg}, y^i_{avg})\}_{i=0:9}$, and take the ten tuples as the metrics for further use. For an input keystroke eye image, we compute its corresponding pupil center location $L(x, y)$ and the probability estimates for each of the ten classes ($\{P_i\}_{i=0:9}$) whose sum should be 1. We then arrange the ten probabilities in a descending order and choose the top n numbers as the candidates of the input eye image (determination of n would be discussed in Section 4.2). Simultaneously, we measure the Euclidean distance between the pupil center of the input eye image and the average pupil center of each of the n

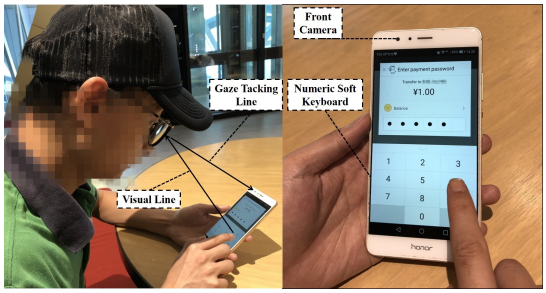


Figure 7: GazeRevealer Setup

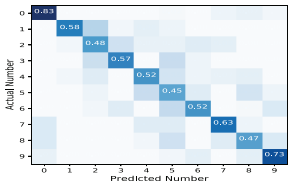


Figure 8: Confusion Matrix

1 (63, 13)	**2** (54, 17)	**3** (39, 16)
4 (61, 18)	**5** (50, 16)	**6** (42, 21)
7 (58, 27)	**8** (42, 23)	**9** (52, 25)
	0 (48, 32)	

Figure 9: Average Pupil Center Positions

candidates as follows:

$$\{D((x,y),(x^i_{avg},y^i_{avg}))\}_{i=1:n} = \{\sqrt{(x-x^i_{avg})^2+(y-y^i_{avg})^2}\}_{i=1:n}$$

where (x, y) is the pupil center of the input eye image and (x^i_{avg}, y^i_{avg}) represents the average pupil center of the selected candidate. Ultimately, a score for each candidate can be calculated as:

$$\{Score_i = P_i \times \lambda D_i\}_{i=1:n}$$

where P_i represents the classification probability of each of the candidates. D_i refers to the corresponding distance. λ is a weight value which equals to $1/{D_i}^2$. The formula denotes that the higher the score is, the higher possibility the candidate is actual the input key number that the keystroke eye image related to. GazeRevealer elects the candidate which has the maximum score as the predicted keystroke number.

4 EVALUATION

4.1 System Setup

We now move to evaluate GazeRevealer. We conduct our experiments on WeChat Pay, a popular online payment platform offered by the social media application WeChat in China. The experimental setting is illustrated in Fig. 7. When a payment interface is brought up on the screen, the front camera starts to record a video of the victim's eye movements during the password input process. The video sample will be then fed into GazeRevealer to infer the 6-digit password. We use three types of smartphones, i.e., Huawei Honor V8, Oppo R11, and Samsung Galaxy S5. These smartphones are equipped with a $30fps$ front camera and a screen size of 5.7 inches, 5.5 inches and 5.1 inches, respectively. In the setting of front camera for video capturing, we use the frame resolution of 1280×720 pixels and the X1 zoom level (i.e., the default zoom setting of front camera for most of the off-the-shelf smartphones) for all the experiments.

We recruit 12 participants (i.e., 4 females and 8 males) in our experiments. They are student volunteers aged between 22 to 33. During data collection, we follow a typical password entering scenario and instruct the participants to gaze at the key number that is being tapped. Participants who wear glasses are asked to adjust their glasses on the top of the nose, following a usual practice[1]. This is to avoid the potential occlusions from glass frames. Since the classification relies on specific features of the eye image, glass frames in the image may obfuscate the features, leading to misclassification.

We first conduct an experiment to evaluate the inference accuracy of the single key number and the inference accuracy of the 6-digit password. Next, we evaluate the robustness of GazeRevealer against various factors including the distance between victim's eyes and smartphone screen, the ambient illumination intensity, and the motion of victims. Finally, we evaluate the influences of user-dependency and user-independency, and compare the performance in these two cases.

4.2 Single Key Number Inference

In our keystroke recognition process, the performance of SVC classification is the key to the success in recognition while the enhancement method is used for improving inference accuracy. In this experiment, we evaluate different keystroke images which are used to recognize different numbers in a real-world scenario. For each type of the three smartphones, we collect the data from 12 participants in an office under normal lighting circumstance (i.e., in a range between 500-1000 *lux*). The distance between eyes and screen is 20 cm typically. Each of the participants is asked to perform 10 cycles, and for each cycle we obtain a video stream that records the eye patterns for the number from 0 to 9 by tapping the corresponding key on smartphone screen. We hence obtain a total number of 12 $participants$× 10 $cycles$×10 $numbers$ = 1200 $keypresses$. We apply the inference method described in Section 3 on the collected dataset and obtain the inference accuracy. In this experiment, we use 5-fold cross validation to evaluate the classifier for user-dependent recognition. For every 10 cycles data, 8 of them are used for training and the remaining 2 are used for testing.

We use Huawei Honor V8 as an example to elaborate the inference process. Firstly, we evaluate the performance of SVC on the relevant dataset in 5-fold cross validation. Fig. 8 shows the confusion matrix of classification result for Honor V8. For a specific key number, the confusion matrix presents the corresponding prediction accuracy which is shown along the diagonal regions. Darker areas in the figure denote higher predictive accuracy for a specific key number. We observe from the figure that the classifier typically confuses each actual input number with other two numbers. This phenomenon may due to the physical layout of numeric soft keyboard where each number has 2 to 3 closest neighbours.

An enhancement method described in Section 3.4.2 is proposed to improve the accuracy based on the above observation. In this method, we choose the first 3 highest-probability candidates, and determine the best candidate for each actual input number. The

[1]https://www.wikihow.com/Wear-Your-Glasses

Table 1: Inference Process of Number 3

Input key number: 3
Pupil center position: (45,18)
3 Candidates:
Number 3, $P_3 = 0.3365$
Number 5, $P_5 = 0.194$
Number 2, $P_2 = 0.1178$
Distance comparison:
$D_{3,3} = \sqrt{(39-45)^2 + (16-18)^2} \approx 6.32$
$D_{5,3} = \sqrt{(50-45)^2 + (16-18)^2} \approx 5.39$
$D_{2,3} = \sqrt{(54-45)^2 + (17-18)^2} \approx 9.06$
Scores:
$S_3 = P_3/D_{3,3} \approx 0.058$ √
$S_5 = P_5/D_{5,3} \approx 0.036$
$S_2 = P_2/D_{2,3} \approx 0.013$

Table 2: Inference Process of Number 1

Input key number: 1
Pupil center position: (61,10)
3 Candidates:
Number 2, $P_2 = 0.3275$
Number 4, $P_4 = 0.2166$
Number 1, $P_1 = 0.1408$
Distance comparison:
$D_{2,1} = \sqrt{(54-61)^2 + (17-10)^2} \approx 9.9$
$D_{4,1} = \sqrt{(61-61)^2 + (18-10)^2} = 8.0$
$D_{1,1} = \sqrt{(63-61)^2 + (13-10)^2} \approx 3.61$
Scores:
$S_2 = P_2/D_{2,1} \approx 0.033$
$S_4 = P_4/D_{4,1} \approx 0.027$
$S_1 = P_1/D_{1,1} \approx 0.039$ √

average pupil center positions of the ten numbers are summarized in Fig. 9. We use an example in Table 1 to illustrate the calculation process of our enhancement method. Although the distance value $D_{5,3}$ (i.e., the distance between the position of the candidate number 5 and the position of the actual input number 3) is smaller than $D_{3,3}$, the method ultimately chooses the candidate number 3 with the highest score as our inferred number. From the result shown in Table 1, we can see that the classifier recognizes the input key number correctly (the prediction probability of number 3 is the largest, i.e., 0.3365). It seems unnecessary to add the following distance comparison. In Table 2, we provide another example to explain its necessity. From the prediction probabilities, we can figure out that the classifier recognizes the actual input key number 1 as number 2 (the prediction probability of number 2 is 0.3275, which is the highest). However, after appending the distance comparison constraint, number 1 achieves the maximum score. As a result, we take the number 1 as the final inference result for the input key number. Fig. 10 presents the average inference accuracy of each key number on Honor V8. We observe a significant improvement on individual keys when employing the enhancement model to the classifier, i.e., the overall average accuracy of all the ten numbers increases from 57.09% to 77.43%.

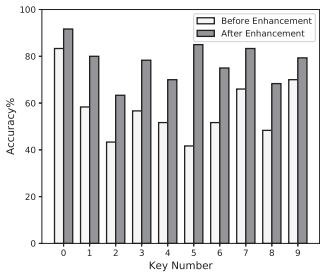


Figure 10: Inference Accuracy per Key

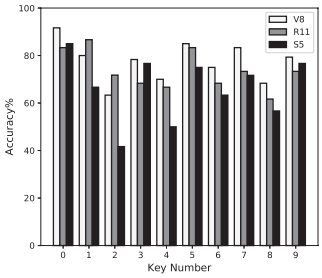


Figure 11: Inference Accuracy on Different Devices

Fig. 11 shows the accuracy of each key number on the three devices we used, i.e., Huawei Honor V8, Oppo R11, and Samsung Galaxy S5. The result shows that GazeRevealer achieves an overall average accuracy of 77.43% on V8, 73.68% on R11, and 66.33% on S5, respectively, for all the ten numbers. In the WeChat Pay app, the numeric soft keyboard size is adjusted to the size of its smartphone screen, users cannot customize the size of the soft keyboard. In other words, the bigger the smartphone screen size is, the larger its soft keyboard will be. From the result, we see that the size of smartphone screen does influence the inference accuracy. The bigger the screen size is, the higher the inference accuracy will be. V8 has the highest accuracy with a screen size of 5.7 inches, the accuracy of R11 with a screen size of 5.5 inches is slightly lower than that of V8. S5 with a screen size of 5.1 inches has the lowest accuracy among the three devices.

4.3 6-digit Password Inference

In this experiment, we evaluate the performance of GazeRevealer for 6-digit password inference. For each type of the three smartphones, we ask the 12 participants to input 50 randomly generated 6-digit passwords. The dataset contains a total of 12 $participants\times$ 50 $sets$ = 600 $samples$ for each smartphone.

The 600 passwords include 3600 key numbers. The results show that a total of 2770 key numbers are accurately inferred on V8 (76.94%), 2668 key numbers are recovered on R11 (74.11%), and 2357 key numbers are recovered on S5 (65.47%), respectively. For an integral 6-digit password, it fails when a single digit number is misclassfied. We introduce a premise which is similar to that in [12]. For deducing a 6-digit password, an attacker can implement several attempts to gain the correct password. It resembles a bit the brute-force attack which tries at most 999,999 times to crack a 6-digit password. We further investigate how many attempts it needs that GazeRevealer can correctly predict a 6-digit password. Each digit number is associated with an eye image. GazeRevealer analyzes the image and yields the first 3 candidates with the highest scores. The overall predicted score of a 6-digit password is defined as:

$$S_{overall} = \prod_{i=1}^{6} S_i$$

where S_i is the score of an individual digit number. As each digit number has 3 candidates, we obtain $3^6 = 729$ potential passwords

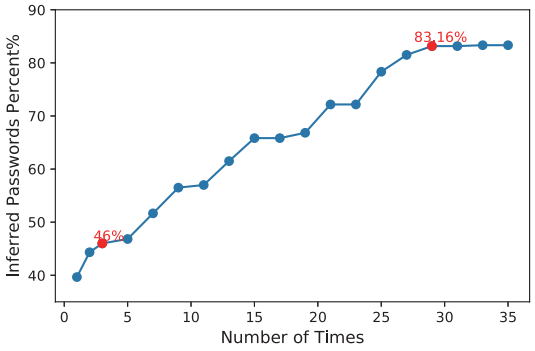


Figure 12: Inference Accuracy of 6-digit Password

Table 3: Inference Rate on Different Devices

Attempts	30	45	60	75
V8	83.16%	83.16%	83.33%	83.33%
R11	66.83%	79.5%	79.5%	79.83%
S5	51.16%	57.33%	68.5%	68.5%

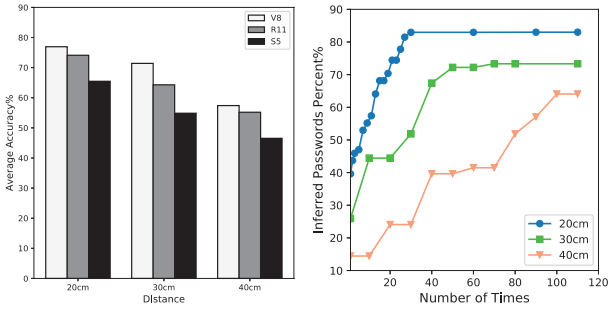
for a 6-digit password. It is much smaller than the number of attempts needed in the brute-force attack. We next arrange the potential passwords in a descending order by their scores. Thus, we can estimate how many attempts it needs to infer the actual password. Fig. 12 shows the inference rate of different attempts on V8, given only 1 trial. GazeRevealer is able to successfully infer 39.66% of the 600 passwords. Given 29 trials, the recovery rate can be significantly improved to 83.16%. In reality, most of the mobile payment apps have their own security policies, e.g., the app will be locked if the user enters the password incorrectly more than 3 times. In our attack, the password inference rate achieves 46% in 3 trials.

As we can see from Table 3, in order to achieve a relatively reasonable inference rate (i.e., less number of trials and higher rate of password inference), V8, R11 and S5 require approximately 30, 45, and 60 attempts, respectively. The corresponding inference rates are 83.16%, 79.5%, and 68.5%, respectively. The result also demonstrates that, for inferring 6-digit passwords in WeChat Pay, the recovery rate is correlated with the smartphone screen size. It is much easier to infer passwords on a smartphone with a larger screen size.

4.4 Influence of Distance

In real situations, the distance between victim's eyes and smartphone screen varies from one to another. According to a study in [18], people are likely to hold smartphones at a distance between 30 cm to 40 cm, and some people who are under age 25 tend to keep a distance as close as 18 cm or 20 cm. In this experiment, we use three distances, i.e., 20 cm, 30 cm, and 40 cm, to evaluate the performance of GazeRevealer. For each distance, We ask each participant to record videos for entering 50 randomly generated passwords on each of the smartphones. We finally obtain 600 password samples for each smartphone on each of the three distances.

Fig. 13(a) shows the average key number inference accuracy for each smartphone on different distances. The result shows that the accuracy of key number inference on the three devices decreases



(a) Average accuracy in different distance (b) 6-digit password inference accuracy comparison

Figure 13: Influence of Distance

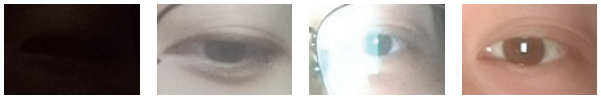
Table 4: Inference Rate in Different Distances

Device	Attempts	20cm	30cm	40cm
R11	100	79.83%	70.16%	61.67%
	150	80.17%	70.16%	61.83%
	200	80.17%	70.83%	62.5%
S5	100	68.5%	59.67%	53.17%
	150	68.66%	61.33%	53.83%
	200	68.83%	61.33%	56.83%

from 76.94%, 74.11%, and 65.47% to 57.4%, 55.19%, and 46.53%, respectively, when the distance increases from 20 cm to 40 cm. We observe a decrease of almost 20% for all the three devices. This indicates that the performance of GazeRevealer can be greatly affected by distance. It is mainly because with a fixed screen size, eye moving patterns become less apparent with longer distance. Hence, the recognition of the eye image for different key numbers reduces, leading to decrease in inference accuracy. Although distance has a relatively great impact on inference, GazeRevealer can still perform an acceptable accuracy on the 6-digit password inference given enough trials. Fig. 13(b) shows the 6-digit password inference rate in different distances on V8. In the cases of 30 cm and 40 cm, the recovery rate rises up to 73.33% and 64.07%, respectively, given over 100 trials. The inference results of R11 and S5 are listed in Table 4. We can see from the table, if given enough trials (200 attempts in our experiment), the inference rate on R11 using the distance of 20 cm, 30 cm, and 40 cm can achieve 80.17%, 70.83%, and 62.5%, respectively; meanwhile, the inference rate on S5 rises up to 68.83%, 61.33%, and 56.83%, respectively. We hence conclude that GazeRevealer can significantly reduce the search space of potential passwords.

4.5 Influence of Illumination Intensity

In this section, we evaluate the impact of illumination on inference rate. We investigate the usability of GazeRevealer under three different illumination scenarios–low illumination (in the range less than 50 *lux*, e.g., twilight and areas with dark surroundings), normal illumination (in the range 500-1500 *lux*, e.g., normal office work and library), and high illumination (in the range 10,000-30,000 *lux*, e.g., full daylight and sunlight). Fig. 14 illustrates the impact of different



(a) Low illumination (b) Normal illumination (c) High illumination (d) Only screen backlight

Figure 14: Eye Images in Different Illuminations

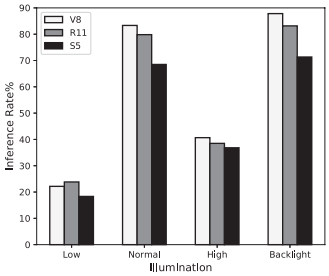


Figure 15: Impact of illumination on inference rate

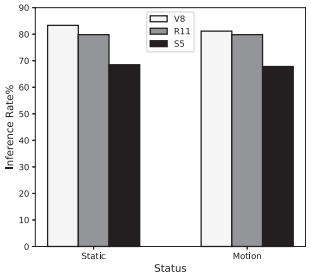


Figure 16: Impact of user status on inference rate

illuminations on eye images. Fig. 14(a) to Fig. 14(c) show images recorded under the three illumination scenarios. In low illumination or darkness circumstance, if we adjust the smartphone screen brightness to the maximum, we observe from Fig. 14(d) that the eye images of the victims without wearing glasses can be clearly recorded only relying on the light from screen. We do not consider the case of wearing glasses since it may lead to excessively strong light reflection on glasses. For the former three scenarios, we ask each participant to record videos for entering 50 randomly generated passwords on each smartphone under different illuminations; for the last one, we ask the participants to take out the glasses and record videos on each smartphone under low illumination by adjusting the screen backlight to the maximum. For every smartphone under four different scenarios, we obtain 600 video samples.

Fig. 15 shows the 6-digit password inference rate (given 100 trials in the experiment) on the three devices under different illuminations. When the illumination switches to low or high, the inference rates of the three devices are even lower than 25% or 40%, respectively. The inference rates under low illumination with maximum screen brightness are slightly higher than those under normal lighting condition, increasing from 83.33% on V8, 79.83% on R11, and 68.5% on S5 to 87.83%, 83.16%, and 71.33%, respectively. It shows that GazeRevealer can still work well under darkness circumstance only when the victims do not wear glasses and the screen backlight is adjusted to the maximum. From the overall results, GazeRevealer is greatly affected by illumination, excessively strong or weak ambient lighting condition may result in failure of GazeRevealer, which can be explained as follows. GazeRevealer fundamentally relies on gaze recognition from relatively high-quality eye images, low illumination usually leads to low image quality, and high illumination may cause light reflection, resulting in low-quality of the eye image. As a consequence, the accuracy of gaze recognition reduces on low-quality images, leading to decrease in password inference.

4.6 Influence of Motion

We now evaluate the impact of victims' walking motion on inference rate. In this experiment, we ask each participant to record videos for entering 50 randomly generated passwords on each device under two states, i.e., static and motion states (walking at a speed of 1.2 m/s approximately while keeping the smartphone steady in front of the face). For each device we used, we obtain 600 video samples for each of the two states.

Fig. 16 shows the inference rate of GazeRevealer with different user states (given 100 trials in the experiment). From the result, we see that the inference rates of 6-digit password nearly keep the same on the three devices when the user state changes. The result indicates that GazeRevealer is robust to the state of motion. The reason is that victims' steady walking motion has little impact on the recording of eye patterns and consequently cause little impact on the password inference.

5 DISCUSSION

5.1 Limitations

GazeRevealer is currently implemented in a lab environment. While our results are encouraging, several limitations need to be considered before real deployment.

- ***Front Camera FPS.*** The FPS of front camera directly relates to the number of images in a gaze fixation and saccade. Keystroke eye image extraction process relies on how many images in each segment. Different FPS result in different numbers of images in the fixation and saccade segments, which would affect the threshold. Consequently, it is hard to apply GazeRevealer to infer the victim's password on other type of smartphones with different FPS. For most of the off-the-shelf smartphones, the current front camera records videos at a speed of 30 FPS. In our experiment, we use a wide range of smartphones (Huawei Honor V8, Oppo R11, and Samsung Galaxy S5) with front cameras of 30 FPS to demonstrate the practicability of the proposed camera-based keystroke inference approach. To overcome this limitation, the most straightforward solution is to train and construct various models with different FPS.
- ***Controlled Input Fashion.*** In our experiment, we instruct the participants to gaze at the key number while entering. This is because the inference rationale is based on eye tracking which relies on the relevant eye image to estimate the gaze direction. In reality, victims may selectively or randomly gaze at certain numbers rather than every number of their password. In this case, we can construct a password dictionary and train probabilistic classifiers such as hidden Markov model (HMM) and neural network (NN) to reduce the complexity of password searching, which resembles the dictionary-based method in character keystroke recognition [4].
- ***Head Movement.*** Head movement will influence the features in the eye image such as the position of pupil center and the size of the eye contour. Consequently, it can significantly affect the accuracy of gaze estimation. GazeRevealer works on the prerequisite that the user keeps a relatively

fixed head position in front of smartphone during password input. In real application scenarios, head may shift around during a conversation while the eyes may still maintain tracking on the screen during password input. This problem can be mitigated using the schemes in [2], or deploying dual front cameras on smartphone (i.e., two different eye images will offer robust computation for gaze estimation).

5.2 Mitigation Strategies

The most direct countermeasure is to employ randomized layout of numeric soft keyboard, the exact number cannot be deduced even if an attacker is capable of figuring out the gaze position on the screen. However, randomizing soft keyboard provides defenses at the cost of usability. For example, it is hard to build muscle memory to type, and hence typing accuracy will be reduced.

Preventing data acquisition is an effective defense against the camera-based side-channel attack. Firstly, app stores such as Google Play should provide a comprehensive inspection mechanism to prevent malicious apps from displaying on the shelves and request every released app to declare the intention of accessing the front camera and other sensors. Secondly, users should selectively grant sensor permissions to apps on their smartphones especially payment apps.

Moreover, a more extreme solution is to eliminate the use of password. Biometrics-based authentication such as fingerprint identification, facial scan, and speech recognition may be an alternative to replace password.

6 CONCLUSION AND FUTURE WORK

In this paper, we propose a novel side-channel based keystroke inference approach using eye movement recordings captured by smartphone front camera. We present the detailed design of GazeRevealer and evaluate our approach on three types of commercial off-the-shelf smartphones. The evaluation results show the promise of employing front camera as the side channel to recognize the victim's password on smartphones. We study several external factors that may influence GazeRevealer on password inference, including the distance, the ambient illumination, and the motion. We also study the performance of GazeRevealer under user-independent case, and the result demonstrates that GazeRevealer is capable of launching the attack for a new victim. In contrast to prior works, our approach only relies on smartphone front camera without the need of complex and easily perceived external devices.

Due to the limitation of camera FPS, we will investigate the performance of GazeRevealer on other devices with different front camera FPS in our future work. We will also study the performance under different body gestures. Furthermore, we will deploy GazeRevealer in many other scenarios such as phone number dialing, smartphone unlocking, and keystroke inference on a full QWERTY soft keyboard.

ACKNOWLEDGMENT

This work is supported by the Key Science and Technology Program of Shaanxi Province, China (Grant No. 2015GY015), and ARC Discovery Grant DP180103932.

REFERENCES

[1] T. Ahonen, A. Hadid, and M. Pietikainen. 2006. Face description with local binary patterns: Application to face recognition. *IEEE Trans. PAMI* 28, 12 (2006), 2037–2041.
[2] A. Al-Rahayfeh and M. Faezipour. 2013. Eye tracking and head movement detection: A state-of-art survey. *IEEE JTEHM* 1 (2013).
[3] K. Ali, A. X. Liu, W. Wang, and M. Shahzad. 2015. Keystroke recognition using wifi signals. In *Proc. 21st Int. Conf. MobiCom*. 90–102.
[4] D. Balzarotti, M. Cova, and G. Vigna. 2008. Clearshot: Eavesdropping on keyboard input from video. In *IEEE Sympo. Security and Privacy*. 170–183.
[5] L. Cai and H. Chen. 2011. TouchLogger: Inferring keystrokes on touch screen from smartphone motion. In *Proc. 6th USENIX Workshop HotSec*.
[6] A. Eshmawi and S. Nair. 2013. Smartphone applications security: Survey of new vectors and solutions. In *IEEE Int. Conf. AICCSA*. 1–4.
[7] D. W. Hansen and Q. Ji. 2010. In the eye of the beholder: A survey of models for eyes and gaze. *IEEE Trans. PAMI* 32, 3 (2010), 478–500.
[8] C. W. Hsu and C. J. Lin. 2002. A comparison of methods for multiclass support vector machines. *IEEE Trans. NN* 13, 2 (2002), 415–425.
[9] Q. Huang, A. Veeraraghavan, and A. Sabharwal. 2015. TabletGaze: unconstrained appearance-based gaze estimation in mobile tablets. *arXiv:1508.01244* (2015).
[10] P. Koutras and P. Maragos. 2015. Estimation of eye gaze direction angles based on active appearance models. In *IEEE Int. Conf. on ICIP*. 2424–2428.
[11] O. Kubovic. 2016. 8 years of Android: malware, malicious apps, and how to stay safe. https://www.welivesecurity.com/2016/09/23/malicious-android-apps/. Accessed Sep. 23, 2016.
[12] M. Li, Y. Meng, J. Liu, H. Zhu, X. Liang, Y. Liu, and N. Ruan. 2016. When CSI meets public WiFi: Inferring your mobile phone password via WiFi signals. In *Proc. of the 2016 ACM SIGSAC Conf. on CCS*. 1068–1079.
[13] A. Mayberry, P. Hu, B. Marlin, C. Salthouse, and D. Ganesan. 2014. iShadow: design of a wearable, real-time mobile gaze tracker. In *Proc. of the 12th Annual Int. Conf. on MobiSys*. 82–94.
[14] S. Narain, A. Sanatinia, and G. Noubir. 2014. Single-stroke language-agnostic keylogging using stereo-microphones and domain specific machine learning. In *Proc. of the 2014 ACM Conf. on Security and privacy in wireless & mobile networks*. ACM, 201–212.
[15] E. Owusu, J. Han, S. Das, A. Perrig, and J. Zhang. 2012. ACCessory: password inference using accelerometers on smartphones. In *Proc. of the 12th Workshop on Mobile Computing Systems & Applications*. ACM, 9.
[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.
[17] R. Schlegel, K. Zhang, X. Y. Zhou, M. Intwala, A. Kapadia, and X. F. Wang. 2011. Soundcomber: A Stealthy and Context-Aware Sound Trojan for Smartphones.. In *Proc. of NDSS*. 17–33.
[18] T. Shibata, D. M. Kim, J.and Hoffman, and M. S. Banks. 2011. The zone of comfort: Predicting visual discomfort with stereo displays. *Journal of vision* 11, 8 (2011).
[19] L. Simon and R. Anderson. 2013. Pin skimmer: Inferring pins through the camera and microphone. In *Proc. of the 3rd ACM workshop on Security and privacy in smartphones & mobile devices*. ACM, 67–78.
[20] I. Sluganovic, M. Roeschlin, K. B. Rasmussen, and I. Martinovic. 2016. Using Reflexive Eye Movements for Fast Challenge-Response Authentication. In *Proc. of the 2016 ACM SIGSAC Conference on CCS*. 1056–1067.
[21] J. C. Sun, X. C. Jin, Y. M. Chen, J. X. Zhang, Y. C. Zhang, and R. Zhang. 2016. VISIBLE: Video-Assisted Keystroke Inference from Tablet Backside Motion.. In *Proc. of NDSS*.
[22] R. Valenti and T. Gevers. 2012. Accurate eye center location through invariant isocentric patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34, 9 (2012), 1785–1798.
[23] P. I. Wilson and J. Fernandez. 2006. Facial feature detection using Haar classifiers. *Journal of Computing Sciences in Colleges* 21, 4 (2006), 127–133.
[24] A. Wulf. 2011. Stealing Passwords is Easy in Native Mobile Apps Despite OAuth. https://welcome.totheinter.net/2011/01/12/stealing-passwords-is-easy-in-native-mobile-apps-despite-oauth/.
[25] Z. Xu, K. Bai, and S. C. Zhu. 2012. Taplogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors. In *Proc. of the 5th ACM conference on Security and Privacy in Wireless and Mobile Networks*. ACM, 113–124.
[26] T. Zhu, Q. Ma, S. F. Zhang, and Y. H. Liu. 2014. Context-free attacks using keyboard acoustic emanations. In *Proc. of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 453–464.
[27] L. Zhuang, F. Zhou, and J. D. Tygar. 2009. Keyboard acoustic emanations revisited. *ACM Trans. on Information and System Security (TISSEC)* 13, 1 (2009), 3.