Your Eyes Reveal Your Secrets: An Eye Movement Based Password Inference on Smartphone

Yao Wang, Wandong Cai, Tao Gu, *Senior Member, IEEE*, and Wei Shao

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 essential to 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 movement patterns, we are able to infer the passwords being entered. We propose a novel algorithm to extract sensitive eye images from video streams, and classify these images with Support Vector Classification. We also propose a novel classification enhancement algorithm to further improve classification accuracy. Compared with prior keystroke detection approaches, GazeRevealer does not require any external auxiliary devices, and it only relies on smartphone front camera. We evaluate the performance of GazeRevealer on several smartphones under different real-life usage scenarios. The results show that GazeRevealer achieves an inference rate of 77.89 percent for single key number and an inference rate of 84.38 percent for 6-digit password in the ideal case.

Index Terms—Keystroke inference, gaze estimation, mobile security

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 (e.g., keyloggers). Typical approaches include electromagnetic emanation based [1], acoustics signal based [2], [3], 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. As a result, attackers cannot leverage these signals to deduce keystrokes anymore. Besides, app permission restriction policies in smartphone operating systems restrain apps from intercepting keystrokes. Trojan applications cannot run directly on smartphones 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 [5], and requiring peripheral camera equipment [6], [7], [8]. They all need an external data receiver which should be placed close enough to the victim. To simplify the inference, attackers start to pay their attentions to smartphone embedded sensors. For example, several works [9], [10], [11] show that keystrokes can be inferred in a stealthy manner with only a few benign app permissions by using accelerometers, gyroscopes, and audio sensors.

In this paper, we present GazeRevealer, a new avenue for attackers to infer user passwords entered on smartphone's touchscreen. GazeRevealer essentially analyzes user eye movements recorded from the smartphone front camera during password input. Our motivation derives from the key observation that user behavior of entering passwords on smartphone always involves coordinated motion between eyes and fingers, i.e., the finger usually taps the key number at which her/is eyes are staring. In other words, eye movement patterns reflect and can be co-located with different keystrokes on soft keyboard. Using this unique nature, GazeRevealer records the eye movements video when a victim enters her/is password and then extracts sensitive images from the video. By processing and analyzing these sensitive

- *Y. Wang and W. Cai are with the Northwestern Polytechnical University, Xi'an 710129, China. E-mail: wangyao@mail.nwpu.edu.cn, caiwd@nwpu.edu.cn.*
- *T. Gu and W. Shao are with the RMIT University, Melbourne, VIC 3000, Australia. E-mail: {tao.gu, wei.shao}@rmit.edu.au.*

Manuscript received 2 Jan. 2019; revised 13 June 2019; accepted 6 Aug. 2019. Date of publication 14 Aug. 2019; date of current version 1 Oct. 2020. (Corresponding author: Tao Gu.)
Digital Object Identifier no. 10.1109/TMC.2019.2934690

images, GazeRevealer is able to infer the corresponding password on touchscreen. In comparison with prior sensor-based attacks [9], [10], [11], 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 faces four major challenges.

1) Our approach relies essentially on the analysis of eye contour images to clip eye image patches from video frames. The existing image clippers can only crop image from a fixed position. In our mobile scenarios, this may lead to the problem that parts of the iris and the sclera may be excluded from the clipped patches. To preserve eye images intact, a precise clipping method is required. In GazeRevealer, we use the Maximum IsoCenter (MIC) based technique [12] to grasp the pupil center from an image precisely and rapidly. We then use the pupil center as the datum to clip a certain pixels in its horizontal and vertical direction separately. In this way, the completeness of eye images is ensured.
2) It is not a trivial task to extract sensitive images that are generated by password input from the video. Because a user may not immediately enter password after the front camera is activated, this could result in noise images at the beginning of the video. Additionally, the time interval between two keys is not fixed. Therefore, we cannot adopt the methods that are commonly used in eye tracking tasks [13] to extract the sensitive images in our scenario. Through investigation we find that different eye states cause intuitive transitions in image histogram. To address this challenge, we leverage similarity that is measured by image histograms as the metric to distinguish sensitive images from the video.
3) In our attack scenario, the victims may not keep their heads still during password input. Head movements could negatively affect the accuracy of gaze estimation, thereby degrade the inference rate. To solve this issue, we create a 3D geometry model for head pose. Based on this model, the angles of head movements (i.e., yaw, pitch, and roll) can be calculated from the 2D front facing images captured by the camera. Then we combine the angle features of head pose with the image features extracted from the eyes and use machine learning algorithms to estimate different keystrokes.
4) In our experiments, we find that the eye tracking algorithm only estimates an approximate position of the key tapping. The recognition accuracy is not ideal enough to get a better inference rate. This can be expected because the layout of digits on smartphone's soft keyboard is typically compact. Study in [14] shows that human gaze direction enters the cornea and passes through the pupil, impling that the pupil naturally follows the movements of gaze direction. By exploiting this unique biological nature, we design an auxiliary model in our eye tracking algorithm to facilitate identifing the most related key number of an eye image.

In summary, this paper makes the following contributions.

- We design a novel side-channel attack approach that enables attackers to infer a victim's password on smartphone touchscreen by analyzing the video of eye movements. Our approach only requires the front camera permission on smartphones that is commonly deemed as normal in daily use, thus potentially jeopardizing mobile device security.
- We propose a sensitive image extraction algorithm, which takes the pupil center as the datum to crop fine-grained eye contour images, then utilizes an image similarity based method to determine the sensitive images from the video.
- We develop an auxiliary model for gaze estimation algorithm, aiming to enhance its classification accuracy. Experiment result shows that the auxiliary model effectively improves the average inference rate of single key number from 59.03 to 77.89 percent.
- We recruit 26 participants in our experiment and evaluate GazeRevealer on three commercial off-the-shelf smartphones. The result shows that GazeRevealer is capable to identify 6-digit password at a rate of 84.38 percent in the best case.

The rest of the paper is organized as follows. Section 2 gives the background and related work. Section 3 describes the detailed design of our system. Section 4 presents the evaluation and discussion is summarized in Section 5. We give the conclusion and future work in Section 6.

2 Background and Related Work

2.1 Threat Model

In our attack, we are not aiming to hack those genuine payment-related apps and manipulate their functionalities. We only require GazeRevealer to be installed on the victim's smartphones and run in the background. According to the report [15] in 2018, an increasing number of smartphone users are jeopardized by untrusted apps, which can be installed in many ways, such as drive-by-download, silent installation, and third-party application market. Many studies have shown that it is not challenging to install malware on smartphones [16], [17]. We therefore believe this assumption does make sense.

The front camera access permission is also required to be granted to GazeRevealer by the users. We believe that most users have no misgivings about granting this permission if GazeRevealer is disguised as normal apps. For example, it is understandable for users to authorize the front camera permission to the selfie apps and mirror apps. It is not our wishful thinking that users are not sufficiently vigilant for this permission. According to the finding in [18], the attacker can acquire the camera permission by any app with some inventiveness. In addition, World Wide Web Consortium (W3C) has lately updated the HTML Media Capture specification to facilitate web access to the camera by JavaScript, which is supported by Android 3+ and iOS 6+ [19]. This implies that GazeRevealer can be more handily embedded in a website and launched online, without installing any applications on the user's smartphones.

GazeRevealer listens to the user's sensitive events, such as pop-ups of password field and number pad. It activates

the front camera to record videos when sensitive events are detected, and turns off the camera once the events are finished. With some coding efforts, this event listener could be easily implemented [20]. After recording, GazeRevealer starts to infer the victim's sensitive inputs.

2.2 Gaze Estimation

The rationale of GazeRevealer is based on gaze estimation techniques that estimate the direction of gaze and track the position of the eyes from eye images. Combined the advanced techniques in computer vision and the current low-cost sensors, gaze estimation has been used in many areas such as marketing to identify which products consumers are interested in. Gaze estimation can be divided into two categories, shape-based and appearance-based tracking approaches.

The shape-based approaches mainly rely on a model of the eyeball and geometrically calculate the gaze direction [21], [22]. In order to build a reliable model of the eye, these approaches require an unalterable distance between screen and user, as well as calibrations for different users. When using multiple cameras, each camera and screen should be placed in a fixed position. Subsequently, a slight change in such parameters may result in a great estimation error [23]. In addition, when using low-resolution cameras and variable lighting conditions, it is difficult for the these approaches to robustly estimate gaze location [24].

Compared with the shape-based approaches, the appearance-based approaches track the gaze direction directly from eye images. Specifically, these approaches treat the process of gaze tracking as a regression problem, they use machine learning and deep learning algorithms to learn mapping functions from eye images to gaze locations [13], [25]. Because these approaches commonly extract high-dimensional features from an entire eye image and map the features to low-dimensional gaze locations, they are capable to handle low-quality images and adaptive to variable lighting conditions [26]. A major issue of these approaches is that they are significantly affected by head poses, requiring an assumption of keeping head pose fixed [27]. In this paper, we leverage on the appearance-based gaze estimation approach due to its superiority for processing low-quality images and managing lighting condition changes. In addition, by taking into account differnet head poses in mobile scenarios, we create a model to mitigate its impact on estimation.

2.3 Keystroke Inference Attacks

Keystroke inference attacks have been developed based on the sensitive information captured from side channels and sensors, falling in the following categories.

1) *WiFi Signal Based*. The attacker infers the keystrokes through WiFi signal. This is motivated that keystrokes will cause different finger motions, which will lead to unique changes in WiFi's channel state information (CSI). Many studies have demonstrated its effectiveness. Ali et al. [1] introduced WiKey that leverages the distinct changes in the CSI caused by the motions of user's finger to identify keystrokes on an external keyboard. Similarly, Zhang et al. [28] and Li et al. [5] also used CSI signals to eavesdrop the graphical unlock patterns and the 6-digit passwords on mobile devices, respectively.

In these works, the attacker must first deploy WiFi devices near the victims. Note that the device should be close enough to the victims (e.g., 30 cm in [1] and 1 m in [5]). If a target who has high security awareness does not connect to public WiFi, these approaches will hence fail. They also require the users to keep a fixed hand gesture in a motionless condition when typing. Besides, the distance and direction between the WiFi antenna and the victims should be stable, since CSI values may vary with any change in these settings. These highly controlled requirements seriously affect their practical application. We only require a video recording of victim's eye movements generated during password input. With no constraints on fixed typing gestures and user behaviors, GazeRevealer is more applicable to reality.

2) *Video Based*. By using an external camcorder, an attacker records victim's sensitive motions to infer keystrokes. Balzarotti et al. [4] recovered the text entered on a keyboard by analyzing a video of victim's typing motion. Maggi et al. [7] used a feature-based template-matching approach to recognize the keystrokes on touchscreen. By recording touchscreen reflections from victim's sunglasses or directly recording victim's touchscreen through shoulder surfing, Raguram et al. [29] reconstructed the text being typed on smartphones. Analogously, Xu et al. [30] exploited reflections from the eyeball when a victim types on touchscreen to detect key presses. Sun et al. [6] recorded a video of the motion patterns of the device's backside caused by taps on touchscreen to infer keystrokes. In [31] and [32], the authors analyzed the shadow formation around the fingertip and the hand dynamics from a video of victim's typing process, respectively, to recover the typed text on smartphones. In [8], the authors used a camcorder to capture victim's eye movements when typing on smartphones, then extracted gaze trace from the video to infer keystrokes.

Similar to WiFi-based attacks, these approaches also require an external device nearby, i.e., camcorder, to capture victim's full view of the sensitive information when typing (e.g., touchscreen reflections from sunglasses or eyeballs, finger motions, and eye movements). If an area is crowded, the victim may be shielded from the view of the camcorder by surrounding obstructions, these attacks will consequently not work. This limitation does not exist in our attack, we only rely on the smartphone's embedded camera. Furthermore, these attacks adopt a wide range of complicated image analysis techniques for motion tacking and eye tracing. Our approach only employs a generic image processing method and simple mathematic representations of eyes and head to identify keystrokes. The simplification of our method implies that it can easily be conducted by a new attacker who has little or even no image processing background.

3) *Sensor Based*. Smartphone embedded sensors provide side-channel attacks to a capacious platform that can be used to eavesdrop user's interactions with the device. Cai et al. [9] presented an accelerometer based inference approach to infer the keys on smartphone's soft keyboard. Later, Owusu et al. [10] applied a similar idea to extract victim's 6-character password on smartphones by using accelerometer readings. Xu et al. [33] utilized the combination of accelerometer and gyroscope to deduce sensitive context on smartphone's touchscreen. Schlegel et al. [11] developed an application by exploiting audio sensor to target privacy information.

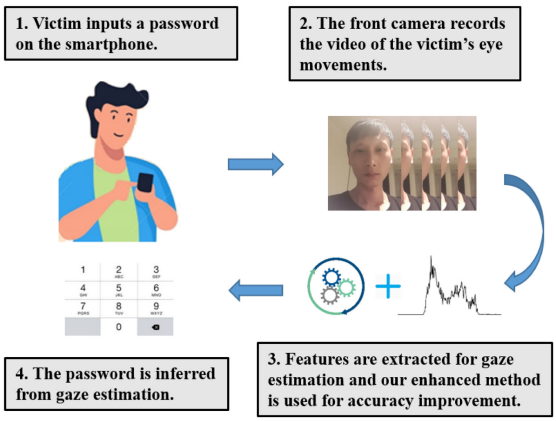


Fig. 1. Illustration of the password inference from eye movements on smartphone.

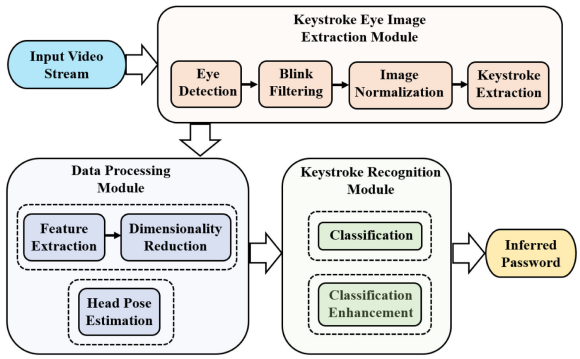


Fig. 2. Framework of GazeRevealer.

Moreover, microphone and camera on smartphones were also adopted to collect victim's sensitive touch-events for inferring keystrokes [18], [34].

These attacks more or less require victim to type in a specific manner. For example, in [9] and [33], a victim is required to hold smartphone by one hand and type on touchscreen by the other hand. In [10] and [18], a victim is assumed to type only using thumbs. In our attack scenario, we do not restrict typing habits, the participants in our experiments may input passwords in their own ways. In addition, the key issue to separate our attack from the sensor based attacks lies in the extendable application range. Because we only require a video of eye movements for password input, our attack has the potential as a side-channel to implement on other devices, such as bank ATMs and door locks, using pinhole camera, as well as laptops with internal web camera. Comparatively, it is difficult to apply the sensor based attacks to such scenarios since they rely on a variety of mobile sensors.

The most related work to this paper is EyeTell [8]. In this paper, the authors developed an attack to infer victim's keystrokes on the touchscreen relying on a video of the eye movements captured by an external HD camcorder. There are several notable factors that distinguish our work from EyeTell. (1) To capture eye movements, they placed an external camcorder in front of a target meters away. Given a victim who conceals her/is eye motions from the view of the camcorder, for example, the victim can lower the head to type on smartphone or pedestrians pass through during recording, their attack would hence fail. Our attack relies on the built-in front camera on smartphone, it is thereby more reliable to capture eye movements. (2) To identify keystrokes, EyeTell adopts a number of intricate techniques for image processing and modeling of the eye trace on touchscreen. Our attack only relies on a generic image processing method and uses simple mathematic and geometric representations of head poses and gaze locations to distinguish different keystrokes. The simplicity of our attack makes it much easier to launch. (3) Their attack was shown to work effectively only within a limited recording angle (i.e., 5 degree), even using a wide-angle camcorder. This is because they did not take into account the head postures which are sensitive to eye tracking. In our attack, we do not assume that the victim keeps a relatively fixed head/typing gestures during password input. We design a 3D geometry model for the head to estimate the constantly changing head postures. Besides, we get rid of eye blink images from the video to eliminate their negative effects on inference, which is not addressed in EyeTell. Overall, with these unique features, GazeRevealer offers a low-cost (i.e., we do not require an additional HD camcorder) and thoroughly different breed of side-channel attack from that in [8].

3 System Design

In this section, we introduce the system design of GazeRevealer and its key modules and algorithms.

3.1 High-Level Overview

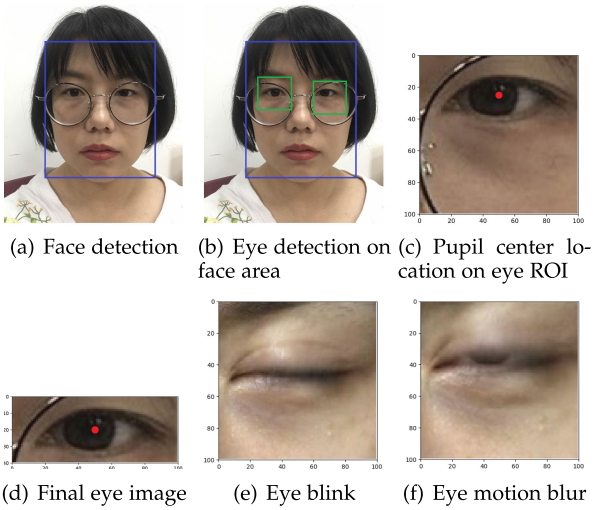
The primary goal of GazeRevealer is to deduce the sensitive information (i.e., password) victims input on smartphone. As illustrated in Fig. 1, GazeRevealer starts to work when keyboard events are detected. The front camera is invoked to record video when a victim inputs a password. Once recording is finished, GazeRevealer extracts features from image frames and adopts machine learning based methods to estimate digits the victim inputted. To improve the estimation accuracy, we propose an enhanced method based on pupil's center location. In the last inference stage, 6-digit password is deduced by applying a candidate election method to the estimated digits.

Fig. 2 presents the overall framework of GazeRevealer, which consists of three main 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 processing module, which extracts relevant features and estimates head poses from eye images; (3) Keystroke recognition module, which determines the keystrokes based on the extracted features and head poses of different eye images.

3.2 Keystroke Eye Image Extraction Module

3.2.1 *Eye Detection, Image Normalization, and Eye Blink Filtering*

In this stage, GazeRevealer first extracts the Region of Interest (ROI) of eyes from each image frame. The process of eye ROI extraction is presented in Fig. 3. For more accurate detection of eye positions, we first rapidly approximate face position from the frame by using a cascade classifier based on Local Binary Patterns (LBP) [35] to narrow down the detection area for eye-pairs (i.e., the blue bounding rectangle in Fig. 3a). After procuring the facial region, eye ROI can be



(a) Face detection (b) Eye detection on face area (c) Pupil center location on eye ROI

(d) Final eye image (e) Eye blink (f) Eye motion blur

Fig. 3. Eye image extraction process. (a)-(d) illustrate the four steps of the eye image extraction. (e) and (f) are the examples of eye blink and eye motion blur, respectively, in which the pupil detection fails.

accurately extracted by using the Haar-based classifier [36], as shown in Fig. 3b. If eye detection fails in this process, we delete the corresponding images from the video stream.

We scale down the eye ROI to a fixed size (i.e., 100×100 pixels in our scenario). In the detected eye ROI, it still contains some factors that are not conducive to gaze estimation, such as eyebrows, hairs, and eyeglass frames. As a consequence, we require a much tighter region around eyes for gaze prediction. 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. To locate the pupil center from low-resolution images (i.e., those captured by smartphone front camera), we utilize the MIC based method [12], which is effective on those low-resolution images. In poor lighting conditions, pupil center may not effectively be detected, we increase the contrast to compensate for the low illumination images. As shown in Fig. 3c, pupil center is located by the red dot. Next, the upper and lower areas of the pupil center are clipped 20 pixels off, respectively, while the horizontal axis keeps 100 pixels unchanged. As shown in Fig. 3d, the size of the final eye image is normalized to 40×100 pixels. In Fig. 3, we take the left eye as an example to illustrate the extraction process. The right eye is processed using the same method.

Eye blinks and eye motion blur are also useless information for gaze estimation and need to be filtered out. Eye blink is a rapid closing of the eyelid, which results in the disappearance of the pupil from the video, as shown in Fig. 3e. Eye motion blur is the streak-like effect in the frame that occurs when the eye blinks, as shown in Fig. 3f, part of the pupil is covered by the eyelid. Intuitively, these interferences can be detected through determining the existence of the pupil. If the pupil detection fails, the eye blinks and eye motion blur are detected. We can also use the MIC based method [12] in this process, since the pupil center is located by this method mainly based on the relatively unbroken pupil image.

3.2.2 Keystroke Eye Images Extraction

After normalization, we obtain two correlated sequences of fixed-size eye images (i.e., the left eye and the right eye). In general, human two eyes are yoked so that they blink at the same time and point towards the same fixation position [37]. In this work, we do not consider special cases such as strabismus. Besides, the study [38] also demonstrated that measurements obtained in one eye are similar to those of the other eye. In this stage, we use the sequence of a single eye to determine the fixation images that can typically represent the corresponding keystrokes. After the keystroke images are determined in the sequence of the single eye, we consequently extract the images from the sequence of the other eye by their correlation properties. The fixation refers to the visual gaze on a specific key that the victim is entering. In the field of eye tracking, the existing fixation image extraction schemes rely on certain defined regulations. For example in [13], the video is divided into several chunks by fixed time intervals (i.e., predetermined duration of the fixation), then different fixation images can be picked out from the chunks. This method is not effective in our scenario since fixation for a key during password input is usually transient (on an order of milliseconds) and non-keystroke images at the beginning of the video are likely to be divided into such chunks. To address this problem, we propose a novel algorithm to precisely extract keystroke eye images. The extraction algorithm consists of three steps as follows:

1) *Image Similarity Estimation.* An eye movement is comprised of fixations and saccades (e.g., a quick eye switch from one key to the next). If we can determine the break points of all fixations and saccades, the stream can be split into multiple segments. Then we can extract image from 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. 4 illustrates the comparison of histograms of four different eye image frames over gray-scale intensity. Figs. 4a and 4b show the histograms of two frames that are selected from the same fixation segment (i.e., key number 1). We compare the histograms in Fig. 4e and observe that different frames under same eye state would lead to similar histograms. Fig. 4c displays the histogram of a frame extracted from key number 2 fixation. As can be observed in Fig. 4f, different eye states result in dissimilar histograms. Fig. 4d presents the histogram of a frame in the saccade moving from number 1 to number 2. We compare it with the histograms of number 2 and number 1 in Figs. 4g and 4h, respectively. It can be observed that they are different from each other. The comparison motivates us to conclude that frames in different eye states would cause intuitive transitions in histogram. We therefore use histogram as the metric to quantize the similarity between two images as follows:

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

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.

2) *Image Stream Similarity Building.* By using the above equation, we can calculate the ith similarity between the ith 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

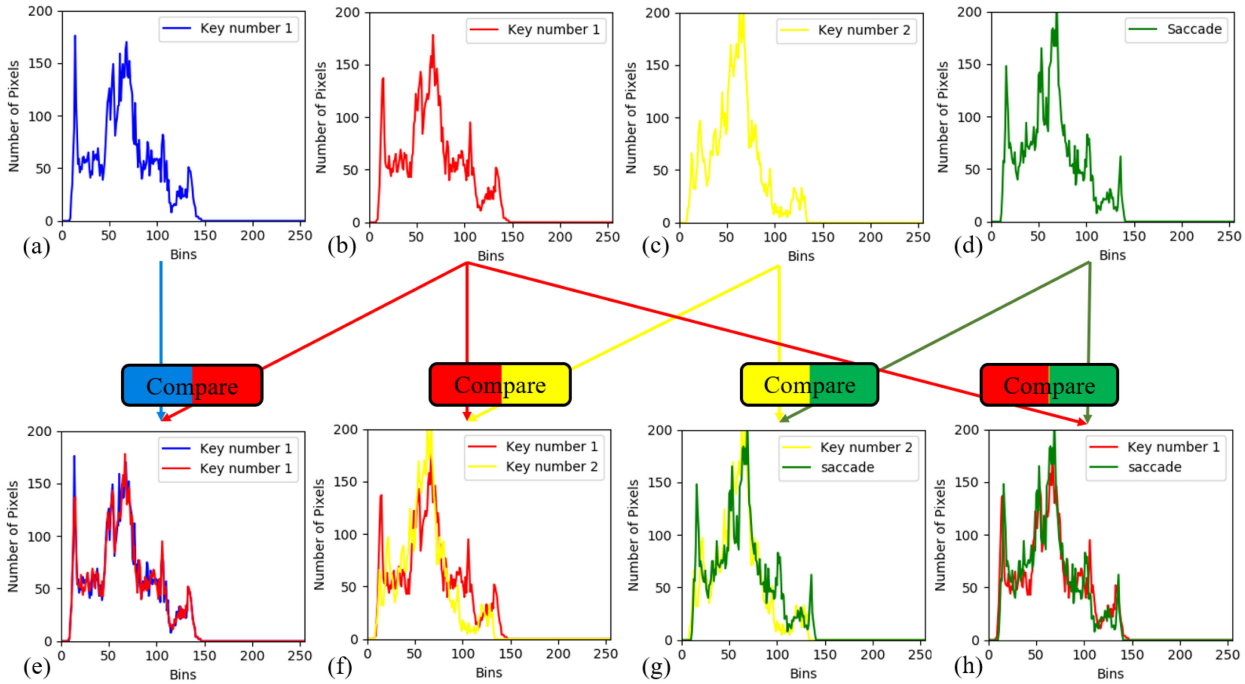
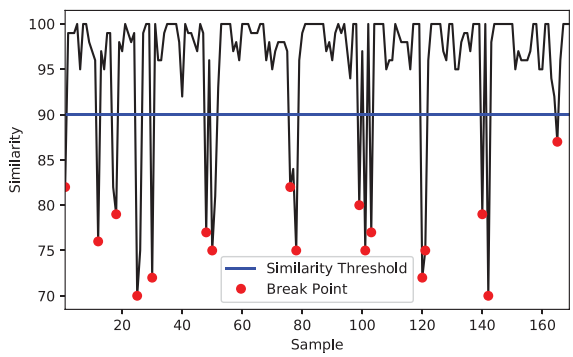
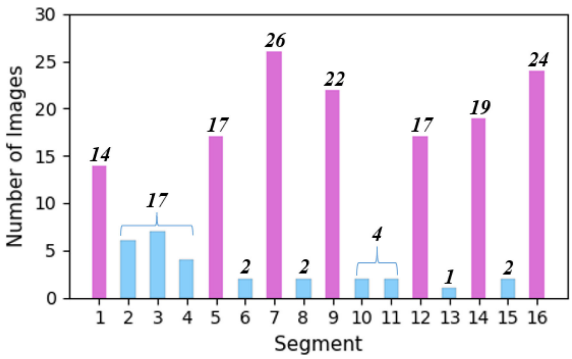


Fig. 4. Image histograms of different eye states. (a) and (b) are the histograms of two different eye image frames when looking at key number 1, (c) is the histogram when looking at kye number 2, and (d) is the histogram of saccade when moving the sight from key number 1 to key number 2. (e)-(h) are the overlaid plots for intuitive comparison.

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. 5a, we take a video of a 6-digit password input as an example to illustrate. After calculating the similarities of all images in the sequence, a similarity waveform can be built. We focus on the red dots where the similarities are lower than the predefined threshold, they are likely to be the break points between fixations and saccades. The threshold denoted by the blue horizontal line is empirically set to 90 in our experiments. It is defined based on the procedure of similarity comparison on our dataset, as illustrated in Fig. 4, which separates the fixation and the saccade for most subjects. We do not consider the impact of light changes on similarity calculation, since the durations of fixation and saccade are typically short during password input (approximately a few hundred milliseconds), light conditions basically maintain stable in such a short time. Therefore, we adopt simple histogram-based method with an empirically predefined threshold to compare eye images rather than using more sophisticated methods such as deep learning.



(a) Image stream similarity waveform



(b) Keystroke segment determination

Fig. 5. Keystroke image extraction.

3) *Keystroke Image Extraction.* After separating the image sequence into several segments by the break points, we next determine the fixation segments of the keystrokes. We continue to use the example in Fig. 5a, 17 break points divide the sequence into 16 segments. In Fig. 5b, we count the number of images in each segment, and select the segments that have numbers satisfying the threshold constraint as our potential keystroke segments (denoted by the purple bars). We measure the threshold constraint based on our investigation of the 26 participants in the experiment. By inspecting the video sequences of 6-digit password input, we find that a keystroke fixation typically contains 11-28 frames and a saccade between two keystroke fixations usually lasts 1-7 frames. This finding is consistent with that in [39] where the authors stated that the duration of a reflexive saccade is usually less than 250 ms (i.e., 8 frames recorded by the 30 fps camera in our scenario). To determine the six keystroke segments from the seven fixation segments (i.e., Segment 1, 5, 7, 9, 12, 14,

and 16), we count the saccade frames between every two fixations and check whether the result falls within the range 1-7 frames. As we can see from Fig. 5b, the saccades between the six consecutive fixation segments, i.e., Segment 5, 7, 9, 12, 14, and 16, satisfy the constraint. We hence regard these six fixations as the keystroke segments for the 6-digit password. Finally, we choose the center frame in each keystroke segment as the key frame to represent the corresponding keystroke. This is because blurry transition frames (e.g., from saccade to fixation) may exist at the beginning and at the end of the keystroke segment, while the center frame is more stable. In Algorithm 1, we conclude the the above three steps for keystroke eye images extraction.

Algorithm 1. Keystroke Frames Extraction

Input: Frame sequence: $F = \{f_1, f_2, \ldots, f_n\}$;
Break point set: $B = \phi$;
Frame segment set: $Seg = \phi$;
Potential keystroke segment set: $S_p = \phi$;
Keystroke segment set: $S = \phi$;
Output: Six keystroke frames:
$F_{key} = \left\{F^1_{key}, F^2_{key}, \ldots, F^6_{key}\right\}$;

Compute similarity $sim_i(f_i, f_{i+1}), i = 1, 2, \ldots, n-1$ using Equation (1);
for $i = 1$ to $n-1$ **do**
  if $sim_i < 90$ **then**
    append f_i to B;
  end if
end for
Update $B = \{f_{i^1}, f_{i^2}, \ldots, f_{i^j}, \ldots, f_{i^m}\}$;
Update $Seg = \left\{\left[f_{i^j+1}, f_{i^{j+1}-1}\right]\right\}, j = 1, 2, \ldots, m-1$;
$Num\_Seg^j = Count(f_{i^j+1}, f_{i^{j+1}-1})$; //Count the number of frames in each segment.
for $j = 1$ to $m-1$ **do**
  if $11 < Num\_Seg^j < 28$ **then**
    append Seg^j to S_p;
  end if
end for
Update $S_p = \left\{Seg^j_1, Seg^j_2, \ldots, Seg^j_k, \ldots, Seg^j_h\right\}$;
$Num\_S_k = Count(Seg^j_k, Seg^j_{k+1})$; //Count the number of frames between every two segments.
for $k = 1$ to h **do**
  if $Num\_S_k < 8$ **then**
    append Seg^j_k and Seg^j_{k+1} to S;
  end if
end for
Select six consecutive segments from S;
Extract the center frame from each segment;
Output the keystroke frames:
$F_{key} = \left\{F^1_{key}, F^2_{key}, \ldots, F^6_{key}\right\}$;

3.3 Data Processing Module

In this section, we extract features from the selected keystroke images for gaze estimation. In addition, considering that users are not likely to keep a fixed head pose in front of the screen when inputting password, they may have variational head poses that would affect the accuracy of gaze prediction. For example, head may shift around during a conversation while the eyes may still maintain tracking on the screen during password input. In order to ensure the effectiveness of our system, we also need to estimate user's head poses and define relevant features.

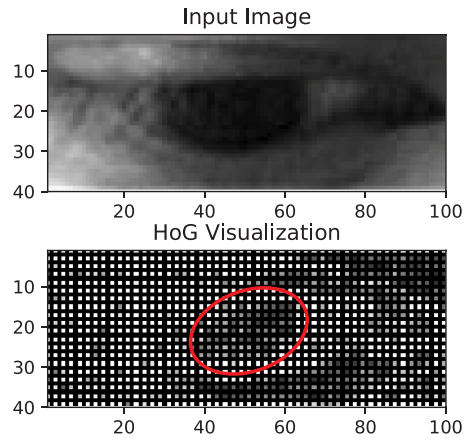


Fig. 6. Feature visualization of an eye image.

3.3.1 Eye Feature Extraction

For appearance-based gaze estimation methods, it is important to choose appropriate method and features to discriminate keystrokes. In our scenario, we simply assume that user has an ordinary smartphone with a front-facing camera. We do not assume users commonly have the latest hardware for smartphones that allows the use of computationally expensive methods, such as convolutional neural networks (CNNs) [40]. As such we use the feature-based estimation technique in our implementation.

Since ambient illumination changes may arise in different usage scenarios, we select Histogram of Oriented Gradients (HoG) as the descriptor which is invariant to the influence of illumination effects [41]. Besides, HoG can distinguish the iris and sclera in low-resolution eye images. As shown in Fig. 6, 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 our eye images. Therefore, we choose HoG features and use the scikit-image tool [42] for gaze estimation, using the following parameters: 9 orientations, 2×2 pixels per cell, and 2×2 cells per block.

HoG features extracted from an eye image would result in high dimension (over 33 k) and suffer from noise. Next, we use Principal Component Analysis (PCA) to reduce the size of the original feature space to a lower dimension. PCA is expected to find the most correlated features and remain the representative information of an image. Fig. 7 shows the result. 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 in our work. To extract features from an eye-pair, we apply the method described above to both of the eye images. As a consequence, we have 8 features in this stage.

3.3.2 Head Pose Estimation

To make our system more flexible in real-world scenario, we measure user's head poses and extract relevant features from the raw face images of the keystrokes. Here the raw

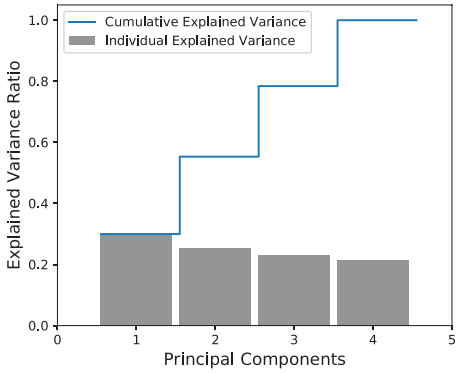


Fig. 7. Principal components selection.

face images refer to the intact video frame, such as the frames in Figs. 3a and 3b.

We use the geometry features to determine the head's orientation. As shown in Fig. 8a, the 3D head movement can be decomposed into three motions, i.e., yaw, pitch, and roll. From a 2D face image captured by the front camera, we intuitively find that the distance between the eyes d_{ee} changes in the case of yaw; the spacing of the nose to the midpoint between two eyes d_{ne} varies in the case of pitch; the position of d_{ne} displaces in the case of roll. Ideally, to calculate the angles of the three motion, users are required to look directly at the front camera for the nominal "0" angles. At this point, we thus have the reference against the subsequent keystroke images to be compared. In our attack scenario, we cannot strictly calibrate the victim's face for the reference. We select a face image which is approximately looking straight ahead from the victim's video as our reference. In what follows, we detail the process of calculating the angles.

As shown in Fig. 8b, yaw can be projected onto the xz plane. Assuming that the eyes move from the reference positions $e_l = (x_l, y_l), e_r = (x_r, y_r)$ to $e_l^{'} = (x_l^{'}, y_l^{'}), e_r^{'} = (x_r^{'}, y_r^{'})$ by rotating β degrees, where e_l, e_r, e_l, and e_r are the pixel positions in the image. Formally, β is calculated as

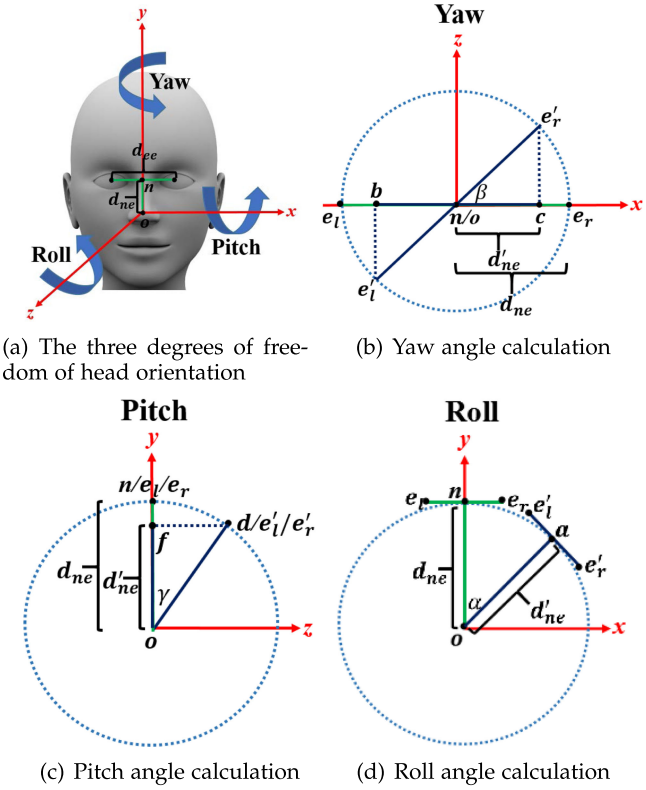
$$\beta = cos^{-1}\left(\frac{x_r^{'} - x_l^{'}}{x_r - x_l}\right). \tag{2}$$

Similarly, in Fig. 8c, pitch can be projected onto the zy plane. When the head rotates from n to d, where the pixel positions of n and d are $(\frac{x_l+x_r}{2}, \frac{y_l+y_r}{2})$ and $(\frac{x_l^{'}+x_r^{'}}{2}, \frac{y_l^{'}+y_r^{'}}{2})$, respectively. γ can thus be calculated as

$$\gamma = cos^{-1}\left(\frac{y_r^{'} + y_l^{'}}{y_r + y_l}\right). \tag{3}$$

By projecting roll to xy plane, the angle α in Fig. 8d represents the rotation for a keystroke eye image. In the image, the nose position o (x_o, y_o) is calculated by nose detection, the positions of the midpoints n and a between the eyes are $(\frac{x_l+x_r}{2}, \frac{y_l+y_r}{2})$ and $(\frac{x_l^{'}+x_r^{'}}{2}, \frac{y_l^{'}+y_r^{'}}{2})$, respectively. Formally α can be derived as

$$\alpha = tan^{-1}\left(\frac{y_r^{'} + y_l^{'} - 2y_o}{x_r^{'} + x_l^{'} - 2x_o}\right) - tan^{-1}\left(\frac{y_r + y_l - 2y_o}{x_r + x_l - 2x_o}\right). \tag{4}$$



(a) The three degrees of freedom of head orientation
(b) Yaw angle calculation
(c) Pitch angle calculation
(d) Roll angle calculation

Fig. 8. Decomposition of head pose.

Next we use the three angles as head pose features for a keystroke image and combine them with the eye features (i.e., a total of 11 features) for gaze estimation.

3.4 Keystroke Inference Module

In this module, we will discuss the details of keystroke recognition based on the extracted features and our enhancement method that aims to improve the inference accuracy.

3.4.1 *Keystroke Recognition*

In our work, keystroke recognition is essentially a ten-class classification problem. Based on the features extracted from eye-pairs and head poses, GazeRevealer estimates the corresponding key number. In our experiment, we use Support Vector Classification (SVC) [43] with Radial Basis Function (RBF) kernel as our classifier. 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

TABLE 1
Average Pupil Center Positions for the Ten Key Numbers

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

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 11 features, some unavoidable factors such as the distance between eyes and smartphone, the touchscreen size, the camera resolution, and even body postures (e.g., standing, sitting, and slouching) are more or less expected to negatively affect the classification. In order to minimize such influence on key number inference, we propose an auxiliary method to the basic classification algorithm to improve the results.

We first measure the average pixel position of pupil center for each of the ten keystroke eye images, denoted as $\{L^i_{avg}(x^i_{avg}, y^i_{avg})\}_{i=0:9}$. As mentioned in Section 3.2.2, two eyes perform similar measurements during movements. To calculate the reference value for each number, we use the 100x100-pixel images of the single eye in our dataset and measure their average pupil center position instead of calculating the positions for the two eyes separately. We report the measurements in Table 1 and take these ten tuples as the references for further use. To infer the corresponding key number of an input eye image, we calculate its pupil center position $L(x, y)$ and the classification probabilities $\{P_i\}_{i=0:9}$. Then, we arrange the ten probabilities in a descending order and select the top n highest estimates numbers as the candidates for the input eye image. Generally, the higher P_i, the more possible that the correct inference number is covered in the candidate set. In our experiment, n is empirically set to 3 (determination of n would be discussed in Section 4.3). 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 candidates as follows:

$$\begin{aligned}&\{D((x, y), (x^i_{avg}, y^i_{avg}))\}_{i=1:n}\\ &= \{\sqrt{(x - x^i_{avg})^2 + (y - y^i_{avg})^2}\}_{i=1:n},\end{aligned} \tag{5}$$

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. The above calculation is a generic compensation for the keystroke recognition in Section 3.4.1 and relies on an assumption that different users may perform similar pupil center positions when looking at the same location on screen. This assumption is proved by the study [44] under a constraint that the distance between the eyes and the screen is relatively unchanged. In the implementation, we collect data at a distance of about 20 cm,[1] resulting in similar pupil center distributions for different users.

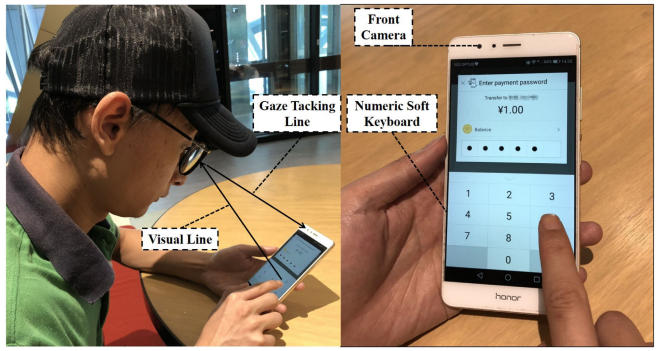


Fig. 9. GazeRevealer setup.

Ultimately, a score for each candidate is calculated as

$$\{Score_i = P_i / D_i\}_{i=1:n}, \tag{6}$$

where P_i represents the classification probability of the candidate and D_i refers to the corresponding euclidean distance. The equation denotes that the higher the score is, the more likely that the candidate is the victim's input key number. GazeRevealer chooses the candidate that has the maximum score as our predicted keystroke number. The above process of inference enhancement is concluded in Algorithm 2.

Algorithm 2. Classification Enhancement

Input: A keystroke eye image img:
$L(x, y)$; //Pupil center location.
Output: Inferred key number n;
1: $P_i|_{i=0:9} = clf.SVC(I)$; //Calculate the probability estimates using SVC classifier.
2: $N_i|_{i=0:9} = SortDescending(P_i|_{i=0:9})$; //Arrange the ten probabilities in descending order.
3: Select the top 3 candidates from $N_i|_{i=0:9}$;
4: Search the corresponding average pupil center location $L^j_{avg}(x^j_{avg}, y^j_{avg})|_{j=0:2}$ for the 3 candidates from Table 1;
5: Measure the distance between $L(x, y)$ and $L^j_{avg}(x^j_{avg}, y^j_{avg})|_{j=0:2}$ using Equation (5);
6: Calculate scores $S_j|_{j=0:2}$ for the 3 candidates using Equation (6);
7: Select number n with the maximum score as the inferred key number;
8: Output the inferred key number n;

4 Evaluation

In this section, we evaluate the performance of our system.

4.1 System Setup

We now move to evaluate GazeRevealer. We conduct our experiments on a popular online payment platform, WeChat Pay offered by the social media application WeChat. The experimental setting is illustrated in Fig. 9. When a payment

1. In general, most people tend to hold smartphones about 20 cm from their eyes when typing. This smartphone habits report is available at: https://www.entrepreneur.com/article/232665.

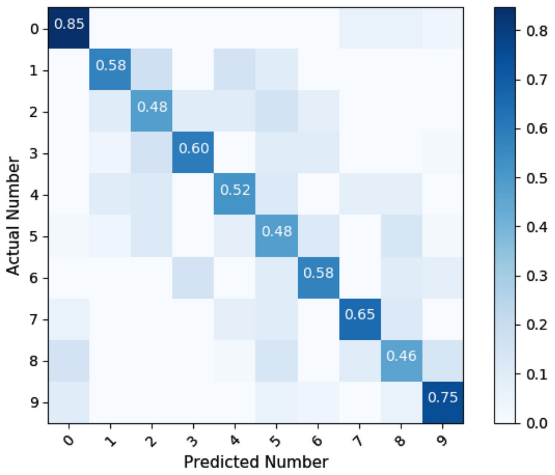


Fig. 10. Confusion matrix of SVC classification on Honor V8.

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, 5.5 and 5.1 inches, respectively. In the setting of front camera for video capturing, we use the frame resolution of 1280 $\times$ 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 26 participants (i.e., 8 females and 18 males) in our experiments, of whom 17 are wearing glasses, and the others are not. They are student volunteers aged between 19 to 33 and they do not have eye problems (e.g., esotropia and exotropia). During data collection, we follow a typical password entering scenario (as illustrated in Fig. 9), a participant sits on a bench in a naturally lit research office where the illumination is in a range between 500-1000 lux. The participants are instructed to hold a smartphone in front of themselves at a distance of approximately 20 cm, and input password by following their own styles (e.g., in their own typing speed on touchscreen, free to blink their eyes as usual during input). We use the above default settings for our experiments unless stated otherwise.

4.2 Data Collection

We first conduct an experiment to evaluate the inference accuracy of entering single key number. Next, we evaluate the inference accuracy of entering 6-digit password. Finally, we discuss the robustness of GazeRevealer against various factors including the distance between eyes and smartphone screen, the ambient illumination intensity, and the motion of victims.

To collect single key numbers, each of the 26 participants is asked to perform 10 cycle samples, where a cycle is defined as a video that records the participant's eye movements when entering the key number from 0 to 9 on soft keyboard. For each type of the three smartphones, we hence obtain a total number of 26 $participants\times$ 10 $cycles\times$10 $numbers$ = 2600 $keypresses$. Keeping the phone at roughly the same position and repeating the same actions may cause fatigue, which could greatly affect the variance across the data. To reduce the impact of fatigue and collect realistic data, we ask the participants to take a break between cycles and stop collection freely once they feel tired.

To collect 6-digit passwords, each of the 26 participants is required to input 50 random passwords produced by a password generator. For each of the three smartphones, the dataset contains 26 $participants\times$ 50 $sets$ = 1300 $samples$. To behave as naturally as possible, for a random password, we ask the participants to keep it in mind. After they claim that they have remembered the random password, they start to enter the password in front of the smartphone. As we can imagine, this data collection process requires the participant to remember the randomly generated password and input it naturally, which may cause extreme fatigue. Similarly, to reduce the impact of fatigue on our data, we take two measures to relieve such negative effects as much as possible. For one thing, the participants are told to get enough rest between random passwords and stop data collection whenever they need a break. For another, we control the data collection time, and ask the participants to complete 5 random passwords on a single day.

In our experiment, collecting the above two types of data costs us more than 20 days. For the data of impact factor experiments, we will present the details in each relevant section.

4.3 Single Key Number Inference

In this section, we apply the inference method described in Section 3 on our collected data and investigate whether the proposed method is effective enough to infer different keystrokes. In real-attack scenarios, it is unlikely for attackers to obtain sufficient data to train a specific model for each victim. To make our attack more realistic, we build a generic model by using data from 26 participants with 5-fold cross validation in the experiment. 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. We first evaluate the performance of SVC classification on our dataset. Fig. 10 shows the confusion matrix of the 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 matrix 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 neighbors.

Based on such observation, we then use our classification enhancement method (which is concluded in Algorithm 1) to improve accuracy. We choose the first 3 highest-probability candidates, and determine the best candidate for each actual input number. In Table 1, we have measured all the average pupil center positions of the ten numbers. We take an example in Table 2 to illustrate the calculation process of our enhancement method. From the table, we observe that although the distance $D_{5,3}$ (i.e., the distance between the average position of candidate 5 and the position of the actual input number 3) is smaller than $D_{3,3}$, our method ultimately chooses the candidate number 3 with the highest score as our inferred number. From the classification results, we can

TABLE 2
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:
$S_3 = P_3/D_{3,3} \approx 0.058$ ✓
$S_5 = P_5/D_{5,3} \approx 0.036$
$S_2 = P_2/D_{2,3} \approx 0.013$

see that the classifier recognizes the input key number correctly (the prediction probability of number 3 is the highest, i.e., 0.3365). It seems unnecessary to add the following distance comparison. In Table 3, we provide another example to explain its necessity. From 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 highest score. As a result, we take number 1 as our final inference for the input key number. Fig. 11 presents the average inference accuracy of each key number on Honor V8. We observe a significant improvement on individual keys when employing the enhancement method to the classifier, i.e., the overall average accuracy of all the ten numbers increases from 59.03 to 77.89 percent.

Fig. 12 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 77.89 percent on V8, 74.26 percent on R11, and 68.64 percent on S5, respectively, for all the ten numbers. In WeChat Pay, the numeric soft keyboard size

TABLE 3
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:
$S_2 = P_2/D_{2,1} \approx 0.033$
$S_4 = P_4/D_{4,1} \approx 0.027$
$S_1 = P_1/D_{1,1} \approx 0.039$ ✓

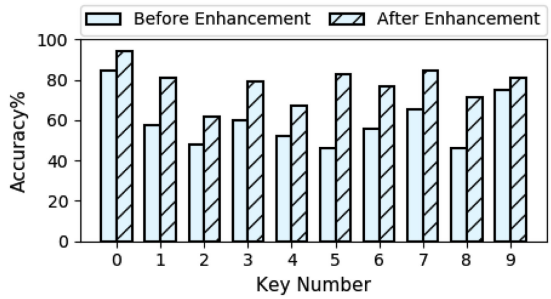


Fig. 11. Inference accuracy per key.

is adaptively adjusted to the size of smartphone screen, users cannot customize the size of the soft keyboard. In other words, the bigger the screen size is, the larger the soft keyboard will be. From the accuracy results, we see that the size of smartphone screen does influence accuracy. The bigger the screen size is, the higher the inference accuracy will achieve. 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.

4.4 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 have collected 1,300 random passwords, which include 7,800 key numbers. The results show that a total of 6,125 key numbers are accurately inferred on V8 (78.53 percent), 5,913 key numbers are recovered on R11 (75.81 percent), and 5,260 key numbers are recovered on S5 (67.44 percent), respectively. In a real-world scenario, for inferring an integral 6-digit password, it fails when any single digit in the password is misjudged. To improve the inference accuracy for 6-digit password, we introduce a premise which is similar to that in [5]. For deducing a 6-digit password, an attacker can implement several attempts to obtain 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

$$S_{overall} = \prod_{i=1}^{6} S_i, \tag{7}$$

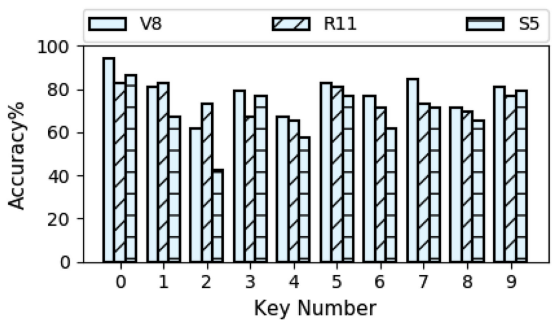


Fig. 12. Inference accuracy on different devices.

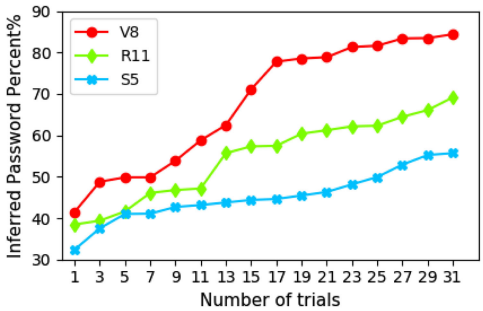


Fig. 13. The impact of different trials on inferring 6-digit password.

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 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. Then, we can estimate how many attempts it needs to infer the actual password. Fig. 13 shows the inference rate of different attempts on the three smartphones. If given only 1 trial, GazeRevealer is able to successfully infer 41.46 percent of the 1,300 passwords on V8, 38.46 percent on R11, and 32.38 percent on S5, respectively. Given 31 trials, the recovery rate can be significantly improved to 84.38 percent on V8, 69.15 percent on R11, and 55.69 percent on S5, respectively. 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, if we try 3 times, the inference rate achieves 48.77 percent on V8, 39.38 percent on R11, and 37.54 percent on S5, respectively. It means that we have at least over one-in-three chance to crack user's password in a practical attack.

As we can see from Table 4, 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 31, 50, and 60 attempts, respectively. The corresponding inference rates are 84.38, 77.53, and 69.00 percent, respectively. The result also demonstrates that, for inferring 6-digit passwords in WeChat Pay, the recovery rate is correlated with the smartphone's screen size. It is much easier to infer passwords on a smartphone with a larger screen size.

4.5 Impact Factors

In this section, we study the impact of various factors on the inference rate of GazeRevealer, including the distance between eyes and screen, the illumination intensity for video recording, the recording angle, the user's motion, and the eyeglasses. We use the default experimental settings (displayed in Table 5), unless stated otherwise.

TABLE 4
Inference Rate on Different Devices

Attempts	31	40	50	60	70
V8	**84.38%**	84.38%	84.38%	84.62%	84.62%
R11	69.15%	74.46%	**77.53%**	77.61%	77.61%
S5	55.69%	57.23%	61.38%	**69.00%**	69.00%

TABLE 5
Experiment Settings

Settings	Parameters
Number of Participants	8 (4 are wearing eyeglasses)
Distance	20cm
Illumination Intensity	Normal lighting (500-1000 lux)
Recording Angle	0 $degree$
Motion	Sitting
Number of Trials	60

4.5.1 Influence of Eyeglasses

As shown in Fig. 14, we generate 3 groups from the 8 participants to study the factor of eyeglasses. Each participant is asked to enter 10 randomly generated 6-digit passwords on each of the three smartphones. Each password is repeated 5 times. The inference rates are shown in Fig. 15. As we can observe, for the same smartphone, the inference rate keeps almost the same. The result means that dividing the participants based on eyeglasses does not have a great influence on inference rate. The factor of wearing eyeglasses does not affect the final inference mainly due to the following two reasons. On the one hand, we apply eye image clipping process to remove potential interferences (e.g., eyeglass frames), as mentioned in Section 3.2.1. On the other hand, in normal lighting conditions, the reflection from eyeglasses is not strong so that it would not produce much noise in the images.

4.5.2 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 [45], people are likely to hold smartphones at a distance between 30 to 40 cm, and people who are under age 25 tend to keep a distance as close as 18 or 20 cm. In this experiment, we use three distances, i.e., 20, 30, and 40 cm, to evaluate the performance of GazeRevealer. For each distance, We ask each participant to record videos for entering 10 randomly generated passwords on each of the smartphones, and each password is repeated 5 times.

Fig. 16 shows the result for this experiment. When the distance increases from 20 to 40 cm, the inference rate decreases from 84.5 percent on V8, 76.25 percent on R11, 66.25 percent on S5 to 66.25, 59.75, and 52 percent, respectively. This indicates that the performance of GazeRevealer can be greatly affected by distance. It is mainly because with a fixed screen size, eye movements become less apparent with longer

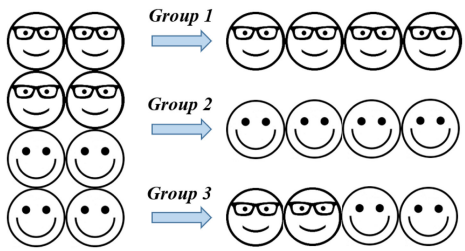


Fig. 14. Diagram of participant grouping for studying the impact of eyeglasses. The 8 participants are partitioned into 3 groups: Group 1, 4 are wearing eyeglasses; Group 2, 4 are not wearing eyeglasses; Group 3, randomly select 4 participants, of whom 2 are wearing eyeglasses, and the other 2 are not.

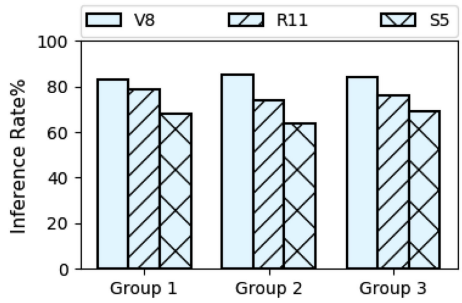


Fig. 15. Impact of glasses on inference rate for different smartphones.

distance. Hence, the recognition of the eye image for different key numbers reduces, leading to decrease in inference rate. Even so, if given enough trials, GazeRevealer can perform an acceptable rate on 6-digit password inference. The results are listed in Table 6. As we can see, if given 300 attempts in this experiment, the recovery rate of V8 in the distance of 20, 30, and 40 cm can reach to 84.75, 82.25, and 74.00 percent, respectively. Similarly, R11 rises up to 80.25, 75.25, and 64.50 percent, respectively, and S5 achieves 69.25, 63.25, and 60.75 percent, respectively. This also demonstrates that GazeRevealer significantly reduces the search space of the potential passwords. In addition, such limitation is expected to be relieved if the attacker uses more advanced video recording techniques. For example, automatic adjustment of zoom settings according to the distance.

4.5.3 Influence of Illumination Intensity

In this section, we investigate the usability of GazeRevealer under three different illumination scenarios: low illumination with less than 50 lux (e.g., twilight and areas with dark surroundings), normal illumination with 500-1500 lux (e.g., office work and library), and high illumination with 10,000-30,000 lux (e.g., full daylight and sunlight). For each scenario, we ask each participant to record videos for entering 10 randomly generated passwords on each of the smartphones, and each password is repeated 5 times.

The result is shown in Fig. 17. As we can observe, when the illumination switches to low and high, the inference rates significantly decrease to less than 21 and 64 percent, respectively. The reason can be explained as follows. Low illumination usually causes blurry and dim images, as illustrated in Fig. 18a. In high illumination environment, especially for people who wear eyeglasses, it causes strong light reflection from eyeglasses as shown in Fig. 18b. GazeRevealer fundamentally

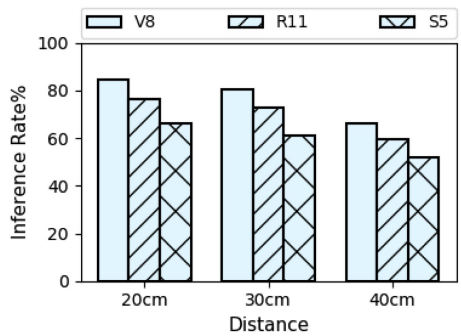


Fig. 16. Impact of distance on inference rate for different smartphones.

TABLE 6
Inference Rates of Different Trials in Different Distances

Device	Attempts	20 cm(%)	30 cm(%)	40 cm(%)
V8	60	84.50	80.25	66.25
	180	84.75	81.75	69.50
	300	84.75	82.25	74.00
R11	60	76.25	72.75	59.75
	180	80.25	74.00	63.25
	300	80.25	75.25	64.50
S5	60	66.25	61.00	52.00
	180	66.25	63.25	53.25
	300	69.25	63.25	60.75

relies on gaze estimation from eye images, unlike the eye image in normal lighting conditions (shown in Fig. 18c), the accuracy of gaze estimation and pupil center detection in such two cases degrades, leading to reduction in password inference.

Noteworthily, we find that screen brightness is automatically adjusted to a higher level in the dark environment (e.g., 0-20 lux), eye images thus can be clearly recorded by the front camera (illustrated in Fig. 18d). We do not consider the case of wearing eyeglasses since it leads to excessively strong light reflection, just as the image in Fig. 18b. We ask the 4 participants who are not wearing eyeglasses to input 10 randomly generated passwords 5 times in this scenario and evaluate the performance. From Fig. 17 we can see that the performance is even higher than that in normal lighting conditions. This is expected because eye contour and pupil center can be clearly captured, thus resulting in better inference rate.

4.5.4 Influence of Recording Angle

Next, we study the impact of angle between eyes and front camera. We evaluate the performance of GazeRevealer on two types of angles, i.e., horizontal angle and vertical angle, they are 0, 10, 20, and 30 degree, respectively. In this experiment, because the distance between eyes and screen is 20 cm, the four different angles can be adjusted by moving the smartphone horizontally and vertically. In each scenario, each participant is asked to perform 10 randomly generated passwords on each of the smartphones. Each password is repeated 5 times.

Figs. 19a and 19b show the results under horizontal angle and vertical angle, respectively. The inference rates in both scenarios drop sharply to less than 35 percent as the angles

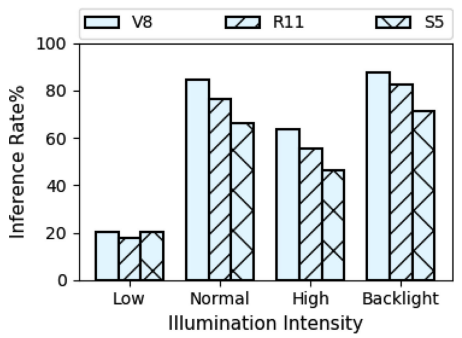


Fig. 17. Impact of illumination on inference rate for different smartphones.

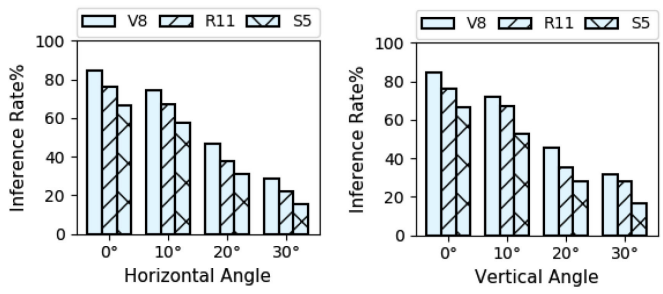
(a) Low illumination (b) High illumination (c) Normal illumination (d) Screen backlight

Fig. 18. Eye images in different illuminations.

increase to 30 degree. It indicates that GazeRevealer is greatly effected by recording angle. This is because of the following two reasons. First, the eye images that are used for training the classifier and calculating the average pupil center are recorded under less than 10 degree. Second, especially for 20 and 30 degree scenarios, part of an eye or an entire eye may not be captured by the front camera. We do not construct the model for one eye gaze estimation, because we find that all the 26 participants tend to hold the smartphone in front of their faces during password input (i.e., the recording angle is less than 10 degree). We do not explicitly ask them to hold the smartphone right in front of their faces before data collection. Gaze estimation under varying head positions (i.e., arbitrary recording angles) is still a challenge in this field [46]. To overcome this issue in our attack scenario (we cannot require the victims to calibrate their head locations before using GazeRevealer), we plan to create several models in our future work. Each model consists of one position and distance of the head. Accordingly, by fitting several models, the system could infer the gaze of users more accurately in varying angles. Furthermore, an increasing number of new smartphones support wide-angle camera, which is easier to capture both eyes in a larger recording angle. We believe that the performance of GazeRevealer could be better on those smartphones with wide-angle camera.

4.5.5 *Influence of Motion*

We now evaluate the impact of user's walking motion on inference rate. In this experiment, we ask each participant to record videos for entering 10 randomly generated passwords 5 times on each device under two states, i.e., static and motion (walking at a speed of about 1.2 m/s). Fig. 20 shows the inference rate of GazeRevealer with different user states. From the result, we see that the inference rates nearly keep the same on the three devices when the user state changes. The result indicates that GazeRevealer still works well when the users are in typical walking. The reason is that users' steady walking motion has little impact on recording of eye patterns and consequently causes little impact on password inference.



(a) Horizontal angle (b) Vertical angle

Fig. 19. Impact of recording angle on inference rate for different smartphones.

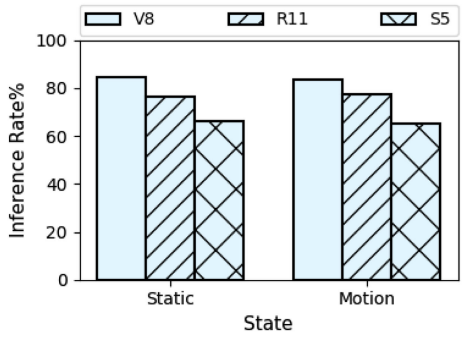


Fig. 20. Impact of motion on inference rate for different smartphones.

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.

1) *Front Camera FPS.* The FPS of front camera directly relates to the number of images in 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 fixation and saccade segments, which would affect the threshold. Consequently, it is difficult to apply GazeRevealer to infer victim's password on 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 several 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.
2) *Lighting Conditions.* Through experiments and observation, the performance of GazeRevealer is not very ideal in low and high illumination scenarios. In high illumination, for people who are wearing eyeglasses, it causes light reflection from eyeglasses, leading to inaccurate gaze estimation and pupil center detection. To solve this issue, we plan to adopt an image inpainting approach [47] to eliminate the reflections in eye images. Low lighting conditions also lead to the failure of GazeRevealer. To alleviate this limitation, some videography tips can be employed. For example, adjusting the ISO sensitivity to a higher level, thereby capturing more light. With further coding effort, GazeRevealer can be more robust in various lighting conditions.
3) *Recording Angle.* In our experiments, we find that the performance of GazeRevealer drops significantly when recording angle is larger than 10 degree. This is because the classification algorithm and the enhancement algorithm we used in the paper assume that the

recording angle is less than 10 degree. This limitation can be addressed by creating several models with various head positions. In addition, user's one eye may not be captured by the camera when recording angle increases. To improve the accuracy of gaze estimation by using features from one single eye, we intend to apply more advanced eye tracking techniques to our system. For example, the approach in [48] effectively estimates eye gaze from single eye image. Besides, this limitation could also be alleviated on smartphones with wide-angle camera as we mentioned in Section 4.5.4.

4) *Gaze Direction*. The study is based on the observation that the gaze directions follow the fingers as they move from one keystroke to another during password entry. This is true in most cases, while some users may not strictly look at each digit, just mechanically moving their fingers because of muscle memory. In this case, as long as the user is not utterly typing without using the sense of sight, we can construct a password dictionary and train probabilistic classifiers to assist our approach. This solution resembles the dictionary-based method in character keystroke recognition to some extent [4], which reduces the complexity of password searching for GazeRevealer.

5.2 Mitigation Strategies

We discuss the mitigation strategies in the following three aspects.

1) *Avoidance*. Since our attack only leverages the user's eye information to infer the sensitive inputs on smartphones, the most direct countermeasure is to hide their gaze information during password input. For example, users can narrow or squint their eyes when inputting password. They can also wear sunglasses to input on touchscreen, so that the gaze information can hardly be acquired by the attacker. Besides, users can mimic the eye movements when typing before entering the real password, the system may thus mistakenly identify those mimicked images as the sensitive images, leading to inference failure.

Another approach against this attack 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.

2) *Elimination*. Preventing data acquisition is also an effective defense against the camera-based side-channel attack. First, 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. Second, users should selectively grant sensor permissions to the apps on their smartphones especially to those that contain payments functionality.

In addition, 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. Users can forget or lose a password, but it is challenging for attackers to steal and forge the personal characteristics.

3) *Workaround*. One easy approach to stay on top of security is to sign up for text alerts for the accounts, e.g., banking and online payment. If the user receives a text notifying that an account has been breached, coupled with considering the credential-stuffing attack, it is imperative to change the password that is also associated with any other accounts. Users should subsequently recheck the apps and remove the suspicious ones on their smartphones.

Besides, taking into account trade-offs in term of security and convenience, payment apps can provide one-time password (OTP) services to users. Under OTP services, small amount payments simply require memorized static passwords; dynamic OTPs are necessitated once the payments exceed users' autonomously preset limits. Because of the one-use nature, OTPs have the potential to secure users even that their keystrokes are captured by attackers.

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's 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 eyeglasses, distance, ambient illumination, recording angle, and motion. In contrast to prior works, our approach only relies on smartphone's front camera without the need of complex and easily perceived external devices.

For our future work, we will investigate new approaches to alleviate the limitations and improve practicability. We also plan to evaluate GazeRevealer with more external factors, such as studying performance under different body gestures (e.g., standing, sitting, and slouching). Furthermore, we will extend GazeRevealer in other application scenarios, such as number dialing, smartphone unlocking, and keystroke inference on a full QWERTY soft keyboard.

Acknowledgments

We would like to thank all the reviewers and editors for their constructive suggestions. We would also like to thank Xiaoluan Zhang and Qianfeng Wang for their supports to this paper. This work is supported by the Key Science and Technology Program Grant (No. 2015GY015) of China and Australian Research Council (ARC) Discovery Project Grants (DP180103932 and DP190101888).

References

[1] K. Ali, A. X. Liu, W. Wang, and M. Shahzad, "Keystroke recognition using WiFi signals," in *Proc. ACM Annu. Int. Conf. Mobile Comput. Netw.*, 2015, pp. 90–102.

[2] L. Zhuang, F. Zhou, and J. D. Tygar, "Keyboard acoustic emanations revisited," *ACM Trans. Inf. Syst. Secur.*, vol. 13, no. 1, 2009, Art. no. 3.

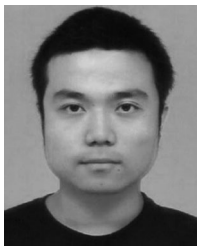
[3] T. Zhu, Q. Ma, S. Zhang, and Y. Liu, "Context-free attacks using keyboard acoustic emanations," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2014, pp. 453–464.

[4] D. Balzarotti, M. Cova, and G. Vigna, "ClearShot: Eavesdropping on keyboard input from video," in *Proc. IEEE Symp. Secur. Privacy*, 2008, pp. 170–183.
[5] M. Li, Y. Meng, J. Liu, H. Zhu, X. Liang, Y. Liu, and N. Ruan, "When CSI meets public WiFi: Inferring your mobile phone password via WiFi signals," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 1068–1079.
[6] J. Sun, X. Jin, Y. Chen, J. Zhang, R. Zhang, and Y. Zhang, "VISIBLE: Video-assisted keystroke inference from tablet backside motion," in *Proc. ISOC Netw. Distrib. Syst. Secur. Symp.*, 2016, pp. 1–15.
[7] F. Maggi, A. Volpatto, S. Gasparini, G. Boracchi, and S. Zanero, "A fast eavesdropping attack against touchscreens," in *Proc. IEEE Int. Conf. Inf. Assurance Secur.*, 2011, pp. 320–325.
[8] Y. Chen, T. Li, R. Zhang, Y. Zhang, and T. Hedgpeth, "EyeTell: Video-assisted touchscreen keystroke inference from eye movements," in *Proc. IEEE Symp. Secur. Privacy*, 2018, pp. 144–160.
[9] L. Cai and H. Chen, "TouchLogger: Inferring keystrokes on touch screen from smartphone motion," in *Proc. USENIX Workshop Hot Topics Secur.*, 2011, pp. 9–9.
[10] E. Owusu, J. Han, S. Das, A. Perrig, and J. Zhang, "ACCessory: Password inference using accelerometers on smartphones," in *Proc. ACM Workshop Mobile Comput. Syst. Appl.*, 2012, Art. no. 9.
[11] R. Schlegel, K. Zhang, X. Zhou, M. Intwala, A. Kapadia, and X. Wang, "Soundcomber: A stealthy and context-aware sound trojan for smartphones," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2011, pp. 17–33.
[12] R. Valenti and T. Gevers, "Accurate eye center location through invariant isocentric patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 9, pp. 1785–1798, Sep. 2012.
[13] Q. Huang, A. Veeraraghavan, and A. Sabharwal, "TabletGaze: Dataset and analysis for unconstrained appearance-based gaze estimation in mobile tablets," *J. Mach. Vis. Appl.*, vol. 28, no. 5/6, pp. 445–461, 2017.
[14] R. Wang, J. Qiu, K. Luo, L. Peng, and P. Han, "Eye gaze tracking based on the shape of pupil image," in *Proc. SPIE Int. Conf. Opt. Instruments Technol.*, 2018, pp. 1–8.
[15] I. Mehta, "500,000 android users downloaded malware made by one developer," Nov. 19, 2018. [Online]. Available: https://twitter.com/EdwardGately/status/1065313844577411072
[16] A. Eshmawi and S. Nair, "Smartphone applications security: Survey of new vectors and solutions," in *Proc. IEEE Int. Conf. Comput. Syst. Appl.*, 2013, pp. 1–4.
[17] X. Lu and S. S. Huang, "Malicious apps may explore a smartphone's vulnerability to detect ones activities," in *Proc. IEEE Int. Conf. Adv. Inf. Netw. Appl.*, 2017, pp. 787–794.
[18] L. Simon and R. Anderson, "PIN skimmer: Inferring PINs through the camera and microphone," in *Proc. ACM Workshop Secur. Privacy Smartphones Mobile Devices*, 2013, pp. 67–78.
[19] A. Kostiainen, I. Oksanen, and H. Dominique, "HTML media capture," Feb. 1, 2018. [Online]. Available: https://www.w3.org/TR/html-media-capture
[20] A. Wulf, "Stealing passwords is easy in native mobile apps despite OAuth," Jan. 12, 2011. [Online]. Available: https://welcome.totheinter.net/2011/01/12/stealing-passwords-is-easy-in-native-mobile-apps-despite-oauth
[21] D. W. Hansen and Q. Ji, "In the eye of the beholder: A survey of models for eyes and gaze," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 3, pp. 478–500, Mar. 2010.
[22] E. Wood and A. Bulling, "EyeTab: Model-based gaze estimation on unmodified tablet computers," in *Proc. ACM Symp. Eye Tracking Res. Appl.*, 2014, pp. 207–210.
[23] Z. Guo, Q. Zhou, and Z. Liu, "Appearance-based gaze estimation under slight head motion," *J. Multimedia Tools Appl.*, vol. 76, no. 2, pp. 2203–2222, 2017.
[24] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling, "Appearance-based gaze estimation in the wild," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 4511–4520.
[25] A. Mayberry, P. Hu, B. Marlin, C. Salthouse, and D. Ganesan, "iShadow: Design of a wearable, real-time mobile gaze tracker," in *Proc. ACM Annu. Int. Conf. Mobile Syst. Appl. Serv.*, 2014, pp. 82–94.
[26] F. Lu, Y. Sugano, T. Okabe, and Y. Sato, "Adaptive linear regression for appearance-based gaze estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 10, pp. 2033–2046, Oct. 2014.
[27] K. Liang, Y. Chahir, M. Molina, C. Tijus, and F. Jouen, "Appearance-based gaze tracking with spectral clustering and semi-supervised Gaussian process regression," in *Proc. ACM Conf. Eye Tracking*, 2013, pp. 17–23.
[28] J. Zhang, X. Zheng, Z. Tang, T. Xing, X. Chen, D. Fang, R. Li, X. Gong, and F. Chen, "Privacy leakage in mobile sensing: Your unlock passwords can be leaked through wireless hotspot functionality," *J. Mobile Inf. Syst.*, vol. 2016, 2016, Art. no. 8793025.
[29] R. Raguram, A. M. White, D. Goswami, F. Monrose, and J. Frahm, "iSpy: Automatic reconstruction of typed input from compromising reflections," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2011, pp. 527–536.
[30] Y. Xu, J. Heinly, A. M. White, F. Monrose, and J. Frahm, "Seeing double: Reconstructing obscured typed input from repeated compromising reflections," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2013, pp. 1063–1074.
[31] Q. Yue, Z. Ling, X. Fu, B. Liu, K. Ren, and W. Zhao, "Blind recognition of touched keys on mobile devices," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2014, pp. 1403–1414.
[32] D. Shukla, R. Kumar, A. Serwadda, and V. V. Phoha, "Beware, your hands reveal your secrets!" in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2014, pp. 904–917.
[33] Z. Xu, K. Bai, and S. Zhu, "TapLogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors," in *Proc. ACM Conf. Secur. Privacy Wireless Mobile. Netw.*, 2012, pp. 113–124.
[34] S. Narain, A. Sanatinia, and G. Noubir, "Single-stroke language-agnostic keylogging using stereo-microphones and domain specific machine learning," in *Proc. ACM Conf. Secur. Privacy Wireless Mobile Netw.*, 2014, pp. 201–212.
[35] T. Ahonen, A. Hadid, and M. Pietikainen, "Face description with local binary patterns: Application to face recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 12, pp. 2037–2041, Dec. 2006.
[36] P. I. Wilson and J. Fernandez, "Facial feature detection using Haar classifiers," *J. Comput. Sci. Colleges*, vol. 21, no. 4, pp. 127–133, 2006.
[37] J. Findlay and R. Walker, "Human saccadic eye movements," *J. Scholarpedia*, vol. 7, no. 7, 2012, Art. no. 5095.
[38] A. Karakosta, M. Vassilaki, S. Plainis, N. H. Elfadl, M. Tsilimbaris, and J. Moschandreas, "Choice of analytic approach for eye-specific outcomes: One eye or two," *Amer. J. Ophthalmology*, vol. 153, no. 3, pp. 571–579, 2012.
[39] I. Sluganovic, M. Roeschlin, K. B. Rasmussen, and I. Martinovic, "Using reflexive eye movements for fast challenge-response authentication," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 1056–1067.
[40] K. Krafka, A. Khosla, P. Kellnhofer, H. Kannan, S. Bhandarkar, W. Matusik, and A. Torralba, "Eye tracking for everyone," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2176–2184.
[41] P. Koutras and P. Maragos, "Estimation of eye gaze direction angles based on active appearance models," in *Proc. IEEE Int. Conf. Image Process.*, 2015, pp. 2424–2428.
[42] Scikit-Image, "Image processing in Python," 2014. [Online]. Available: https://scikit-image.org
[43] C. W. Hsu and C. J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Trans. Neural Netw.*, vol. 13, no. 2, pp. 415–425, Mar. 2002.
[44] J. G. Wang, E. Sung, and R. Venkateswarlu, "Eye gaze estimation from a single image of one eye," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2003, pp. 136–143.
[45] T. Shibata, J. Kim, D. M. Hoffman, and M. S. Banks, "The zone of comfort: Predicting visual discomfort with stereo displays," *J. Vis.*, vol. 11, no. 8, 2011, Art. no. 11.
[46] R. Valenti, N. Sebe, and T. Gevers, "Combining head pose and eye location information for gaze estimation," *IEEE Trans. Image Process.*, vol. 21, no. 2, pp. 802–815, Feb. 2012.
[47] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2536–2544.
[48] S. Park, A. Spurr, and O. Hilliges, "Deep pictorial gaze estimation," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 721–738.

Yao Wang received the BS and MS degrees in software engineering from Xidian University, China. He is currently working toward the PhD degree in computer science and technology at Northwestern Polytechnical University, China. His research interests include privacy protection, mobile computing, and machine learning.

Wandong Cai is currently a professor with the Department of Computer Science and Technology, Northwestern Polytechnical University, China. He is the director of Information Security Institute of Northwestern Polytechnical University, the senior member of China Computer Federation. He authored 16 teaching materials, published more than 220 technical papers, holds more than 10 authorized invention patents. His research interests include complex network and information security.

Tao Gu received the PhD degree in computer science from the National University of Singapore, in 2005. He is currently an associate professor with the School of Computer Science and Information Technology, RMIT University. His research interests lie in the areas of ubiquitous and pervasive computing, mobile computing, wireless sensor networks, big data analytics, and Internet of Things. He is a senior member of the IEEE and a member of the ACM.



Wei Shao received the BS degree in software engineering from Xidian University, China, and the MS degree in software engineering from the University of Hong Kong. He is currently working toward the PhD degree in computer science at RMIT University, Australia. His interest research area focuses on data mining, spatio-temporal data analysis, and device-free activity recognition.