My Google Glass Sees Your Passwords!
(Black Hat USA 2014 White paper)

Qinggang Yue[1], Zhen Ling[2], Xinwen Fu[1], Benyuan Liu[1], Wei Yu[3] and Wei Zhao[4]

University of Massachusetts Lowell, USA[1], {xinwenfu,qye,bliu}@cs.uml.edu
Southeast University, China[2], zhenling@seu.edu.cn
Towson University, USA[3], wyu@towson.edu
University of Macau, China[4], weizhao@umac.mo

Abstract. In this white paper, we introduce a novel computer vision based attack that automatically discloses inputs on a touch enabled device. Our spying camera, including Google Glass, can take a video of the victim tapping on the touch screen and automatically recognize more than 90% of the tapped passcodes from three meters away, even if our naked eyes cannot see those passcodes or anything on the touch screen. The basic idea is to track the movement of the fingertip and use the fingertip's relative position on the touch screen to recognize the touch input. We carefully analyze the shadow formation around the fingertip, apply the optical flow, deformable part-based model (DPM) object detector, k-means clustering and other computer vision techniques to automatically track the touching fingertip and locate the touched points. Planar homography is then applied to map the estimated touched points to a software keyboard in a reference image. Our work is substantially different from related work on blind recognition of touch inputs. We target passcodes where no language model can be applied to correct estimated touched keys. We are interested in scenarios such as conferences and similar gathering places where a Google Glass, webcam, or smartphone can be used for a stealthy attack. Extensive experiments were performed to demonstrate the impact of this attack. As a countermeasure, we design a context aware Privacy Enhancing Keyboard (PEK) which pops up a randomized keyboard on Android systems for sensitive information such as password inputs and shows a conventional QWERTY keyboard for normal inputs.

1 Introduction

Touch enabled devices are ubiquitously used in our daily lives. Nonetheless, they are also attracting attention from adversaries. In addition to hundreds of thousands of malware [1], one class of threats against mobile devices is computer vision based attacks. We can classify those attacks into three groups. Firstly, there are attacks that can directly identify text on a screen or its reflections on objects [2,3]. Secondly, there are attacks that can detect visible features of the keys such as light diffusion surrounding pressed keys [4] and popups of pressed keys [5,6]. Thirdly, there are attacks blindly recognize the text while text or popups are not visible [7] to the adversary.

In this paper, we introduce a novel attack that can blindly recognize inputs on touch enabled devices by estimating the location of touched points from a video associated

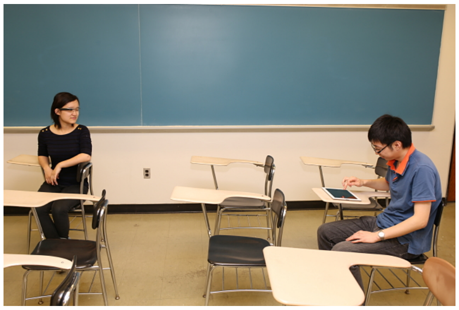

Fig. 1: Google Glass Spying on a Target: Success rate $>$ 90 % in 30 experiments.

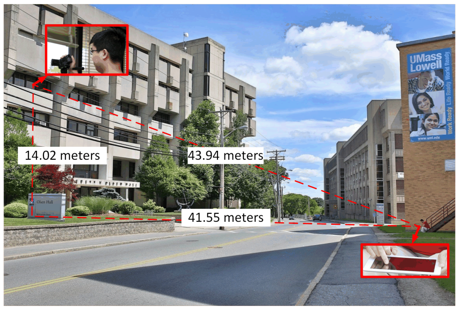

Fig. 2: Remote Attack with Camcorder: Success rate 100 % in 30 experiments.

with people tapping on the touch screen as shown in Figures 1 and 2. In the attack, the optical flow algorithm is used to automatically identify touching frames in which a finger touches the screen surface. We use the intersections of detected edges of the touch screen to derive the homography matrix, mapping the touch screen surface in video frames to a reference image of the software keyboard. Deformable Part-based Model (DPM) and other computer vision techniques are applied to automatically estimate a tiny touched area.

We carefully derive a theory of the shadow formation around the fingertip and use the k-means clustering algorithm to identify touched points in the tiny touched area. Homography can then map these touched points to the software keyboard keys in the reference image in Figure 4. We carried out extensive experiments on the target devices, including iPad, Nexus 7, and iPhone 5. The cameras include a webcam, a phone camera, Google Glass, and even a camcorder (for comparison with existing research efforts). The camera was positioned from different distances and angles. We were able to achieve a success rate of more than 90 % in various scenarios.

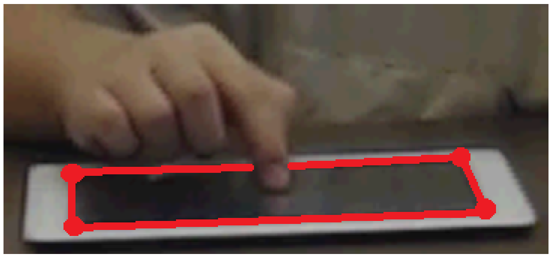

Fig. 3: Touching Frame

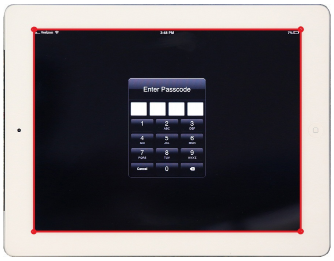

Fig. 4: iPad's Software Keyboard

Our work is substantially different from the most related work by Xu *et al.* [7]. First, we target password inputs whereas Xu *et al.* focused on meaningful text so that they can use a language model to correct the prediction. In comparison with [7] on recognizing passwords, we can achieve a high success rate. Second, we employ a completely different set of computer vision and machine learning techniques to track finger movement and accurately identify touched points. Third, the threat model and targeted scenes are different. We study privacy leakage in scenes such as classrooms, conferences, and other similar gathering places. We use a webcam, smartphone camera, and Google Glass for stealthy attacks whereas single-lens reflex (SLR) cameras with big lens and high-end camcorders with high optical zoom were used in [7] for remote targets. For comparison, we use a Panasonic HDC-SDT750 with 12x Optical zoom for spying on iPads from a distance of more than 43 meters away and achieve a success rate of 100 % in our 30 experiments.

In order to defend against many computer vision based attacks including the one in this paper, we designed a context aware randomized software keyboard for Android, denoted as a Privacy Enhancing Keyboard (PEK). A PEK automatically shows a conventional QWERTY keyboard for normal text input and pops up a randomized keyboard for the input of sensitive information such as passcodes. The first PEK prototype was demonstrated at the ACM Conference on Computer and Communications Security (CCS) Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM) in October, 2012[1]. To the best of our knowledge, the PEK is the first generic software keyboard for a mobile platform while a similar app *CodeScrambler* for iOS [8] appeared in August 2013. PEK is a full-fledged software keyboard whereas CodeScrambler is designed only for unlocking screens and does not provide context-aware functionality.

The rest of the paper is organized as follows. We introduce homography and DPM in Section 2. In Section 3, we introduce the attack. In Section 4, we show experimental design and evaluations. In Section 5, we introduce PEK. We conclude this paper in Section 6.

2 Background

In this section, we introduce the two major computer vision techniques employed in this paper: planar homography and the DPM (Deformable Part-based Model) object detector.

2.1 Planar Homography

Planar homography is a 2D projective transformation that relates two images of the same planar surface [9]. Assume $p = (s, t, 1)$ is any point in an image of a planar surface and $q = (s', t', 1)$ is the corresponding point in another image of the same planar surface. The two images may be taken by the same camera or by different cameras. Then, for q and p, there exists an invertible 3×3 matrix $\mathbf{H}$, denoted as the homography matrix:

$$q = \mathbf{H}p. \tag{1}$$

[1] To the best of our knowledge, our work is the first one towards efforts on PEKs. An early version of this paper is archived at arXiv.org

2.2 Deformable Part-based Model (DPM)

DPM [10] is the state-of-art object detector and contains three main components: a mixture of star-structured part based models, the latent SVM (Support Vector Machine) training process, and an efficient matching process for object detection. It works as follows: First, DPM builds multiple star-structured models for the object of interest from different viewpoints. Each star-structured model has a root model (characterizing the object as a whole) and several (usually six) part models (characterizing each part of the object, particularly the anchor position relative to the root and associated deformation parameters). The models are represented by the Histogram of Oriented Gradients (HOG) [11] feature, which is insensitive to lighting variation. Second, during the training, a bounding box is used to specify the object of interest in each image, where its parts are unlabeled. DPM treats the parts as latent (unknown) variables and employs the latent SVM to train the model. Finally, to detect objects in an image, DPM calculates a score for each possible object sample x:

$$f_{\beta}(x) = \max_{z \in Z(x)} \beta \cdot \Phi(x, z), \tag{2}$$

where $Z(x)$ are the latent values, β is a vector of model parameters, and $\Phi(x, z)$ is the feature vector of x. A high score indicates the location of the object. In order to conduct efficient matching, dynamic programming and generalized distance transforms can be used.

3 Homography Based Attack against Touching Screen

We now introduce the basic idea of the attack and detail each step.

3.1 Basic Idea

Figure 5 shows the flowchart of automatic and blind recognition of touched keys. Without loss of generality, we often use the four-digit passcode input for an iPad as our example.

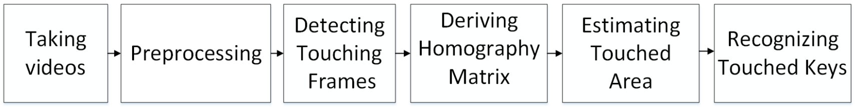

Fig. 5: Work flow of Blind Recognition of Touched Keys

- **Step 1.** Take a video of the victim tapping on a device. We do not assume the video records any text or popup, but we assume the finger movement is recorded.
- **Step 2.** Preprocess the video and keep only the touch screen area showing moving fingers. The type of device is known and we also obtain a high resolution image of the corresponding software keyboard, denoted as *reference image*, as shown in Figure 4.

- **Step 3.** Detect the video frames, denoted as *touching frames*, in which the finger touches the screen surface, as shown in Figure 3.
- **Step 4.** Identify features of the touch screen surface and derive the planar homography matrix between the video frames and the reference image.
- **Step 5.** Estimate the touched area in the touching image. This is a key step to implement automated touched key recognition. We use DPM and various computer vision techniques and obtain a tiny box bounding the touched area.
- **Step 6.** Identify the touched points from the estimated touched area and map them to the reference image through homography.

Through the aforementioned steps, if the touched points can be correctly located, we can disclose the corresponding touched keys. In the following, we introduce the six steps in detail.

3.2 Step 1: Taking Videos

The adversary takes a video of a victim tapping on a device from a distance. Such scenarios include students taking classes, researchers attending conferences, and tourists gathering and resting in a square. Taking a video in such a place with a lot of people around should be stealthy. With the development of smartphones, webcams and various wearable devices, such kinds of stealthy attacks are feasible. For example, iPhone and *Google Glass* cameras have decent resolution. *Galaxy S4 Zoom* has a 16-megapixel (MP) rear camera with a 10x zoom lens, weighing only 208g. Amazon sells a webcam-like plugable 2MP USB 2.0 digital microscope with a 10x-50x optical zoom lens [12]. Many smartwatches also have a decent camera.

Three factors that affect the attack success are angle, distance, and lighting. Recall the success of the attack relies on accurately identifying touched points. The camera needs to adjust the angle to record finger movement and the touch screen. For example, in a conference room, an adversary in the front can use the front camera of his/her phone to record a person tapping in the back row. The camera cannot be too far away from the victim; otherwise, it is hard to recognize the finger's movement on the screen and the touched area. Of course, a camera with a large optical zoom lens can help in such a case. Lighting affects the brightness and contrast of the video and thus the recognition result.

3.3 Step 2: Preprocessing

In the step of preprocessing, we crop the video and keep only the area containing the moving hand on the touch screen. This removes most of the useless background. If the device does not move in the touching process, we need only to locate the area of interest in the first video frame and keep the same area for all the video frames. If the device moves during the touching process, we need to track its movement and crop the corresponding area. There are several tracking methods [13]. We choose the predator in this study [14]. We first draw the bounding box of the target area. The tracker will follow the movement of the device and return its location in every frame.

We are particularly interested in the fingertip area, where the finger touches the key. The resolution of this area is often very poor. Hence, we resize the cropped video frame to add redundancy. Specifically, we resize each cropped frame to four times its original size. We also assume the target device brand is known and that the adversary can get a high quality image of the software keyboard on the touch screen. This image is the "reference image," as shown in Figure 4. The image shows the detailed features of the device, particularly the touch screen surface. For example, for an iPad, we choose a black wallpaper so that the touch screen has a high contrast with its white frame. It is not hard to recognize most tablets and smartphones since each brand has salient features. For example, by walking past the victim, the adversary can know the device brand. The adversary may also identify the brand from the video.

3.4 Step 3: Detecting Touching Frames

Touching frames are those video frames in which the finger touches the screen surface. To detect them, we need to analyze the finger movement pattern in the touching process. It is very common that users normally use one finger to tap on the screen and input a passcode. We use this example to demonstrate the essence of our technique.

During the touching process, the fingertip first moves downwards towards the touch screen, stops, and then moves upwards away from the touch screen. The finger may also move left or right while moving downwards or upwards. We define the direction of moving toward the device as positive and the opposite direction as negative. In the process of a key being touched, the fingertip velocity is first positive while moving downwards, then zero while stopping on the screen, and finally negative while moving upwards. This process repeats for each touched key. Hence, a touching frame is one where the fingertip velocity is around zero. Sometimes the finger moves so fast that there is no frame where the fingertip has a zero velocity. In such a case, the touching frame is the one where the fingertip velocity changes from positive to negative.

The challenge to derive the fingertip velocity is to identify the fingertip. The angle that we use to take the video affects the shape of the fingertip in the video. The fingertip shape also changes when the soft fingertip touches the hard touch screen surface. Users may also use different areas of the fingertip to tap the screen. We find that when a user touches keys with the fingertip, the whole hand most likely follows a similar gesture and moves in the same direction. Instead of tracking the fingertip to identify a touching frame, we track the hand, which has a sufficient number of feature points for automatic tracking.

We adopt optical flow theory [15] to derive the velocity of points on the moving finger or hand. Optical flow computes the motion of an object between two frames. The displacement vector of the points between subsequent frames is called the image velocity or the optical flow at that point. We use the KLT algorithm [16], which can track sparse points. To make the KLT algorithm effective, we select unique feature points, which are often corners in the image. The Shi-Tomasi corner detector [17] is applied to obtain these points. We track several points in case that some points are lost during tracking. Our experiments also show that each touch with the fingertip may produce multiple touching frames. This is reasonable because the fingertip is soft. When a fingertip touches the screen, it deforms and this deforming process takes time. Users may also intentionally stop to make sure that a key is touched. During the interaction between the fingertip and touch screen, some tracked points may also move upward. We use a simple algorithm to deal with this noise: if the velocity of most of the tracked points in one frame moves from positive to negative, that frame is a touching frame. Our experiments show that six points are sufficient to detect all the touching frames.

3.5 Step 4: Deriving the Homography Matrix

In computer vision, automatically deriving the homography matrix H of a planar surface in two images is a well-studied problem [18]. First, a feature detector such as SIFT (Scale-Invariant Feature Transform) [19] or SURF (Speeded Up Robust Features) [20] is used to detect feature points. Matching methods such as FLANN (Fast Library for Approximate Nearest Neighbors) [21] can be used to match feature points in the two images. The pairs of matched points are then used to derive the homography matrix through the RANSAC (RANdom SAmple Consensus) algorithm [22].

Nonetheless, these common computer vision algorithms for deriving homography matrix $\mathbf{H}$ are not effective in our context. Because of the angle of taking videos and the reflection of the touch screen, there are few good feature points in the video frames for the algorithms mentioned above to work effectively. Intuitively, touch screen corners are potential good features, but they are blurry in our context since the video is taken remotely. SIFT or SURF cannot correctly detect these corners.

We derive the homography matrix $\mathbf{H}$ in Equation (1) as follows. $\mathbf{H}$ has 8 degrees of freedom. Therefore, to derive the homography matrix, we need 4 pairs of matching points from the same plane in the touching frame and the reference image. Any three of them should not be collinear [18]. In our case, we tend to use the corners of the touch screen as shown in Figures 3 and 4. Because the corners in the image are blurry, in order to derive the coordinates of these corners, we detect the four edges of the touch screen. The intersections of these edges are the desired corners. We apply the Canny edge detector [23] to extract the edges and use the Hough line detector [24] to derive possible lines in the image. We choose the lines aligned to the edges. Then, we can calculate intersection points and derive the coordinates of the four corners of interest. With these four pairs of matching points, we can derive the homopgraphy matrix with the DLT (Direct Linear Transform) algorithm [18]. If the device does not move during the touching process, this homography matrix can be used for all the video frames; otherwise, we should derive H for every touching frame and the reference image.

3.6 Step 5: Estimating the Touched Area

The complication of lighting and shadow makes accurately estimating the touched area a great challenge. We use the following three steps to this end. First, we apply the object detector DPM to derive a relatively large bounding box of the touched area. Second, within the large bounding box, we locate the finger through the k-means clustering of pixels. Finally, we derive the fingertip direction and train a tiny bounding box at the top of the fingertip as the accurately touched area.

To employ DPM to derive the bounding box of the touched area, we first generate positive data (touched area) and negative data (untouched area) to train a model for the touched area. To obtain positive data, we take videos in various scenarios and obtain the touching frames. For each touching frame, we label the touched area with *an appropriate bounding box centered at the touched key*. Here is how we derive the center of a key in a touching frame. During the training process, we know the touched keys and can compute their position by mapping the area of a key from the reference image to the touching frame with planar homography. DPM needs a bounding box large enough to perform well and we use a large bounding box of 40×30 pixels centered at the touched key. Different images may have a different bounding box size and DPM will resize them to a uniform size for training. To obtain negative data, we use the bounding box around the non-touching fingertip. DPM also generates negative data through data mining and treats a bounding box with less than 50 percent intersection of positive data as negative data.

Since the bounding box derived by DPM is often too large, we further locate the fingertip within the large bounding box. Recall that during the training process, the center C of the large bounding box estimates the center of a touched key. The fingertip is around C as most users tend to touch the center of a key. After DPM is applied for detection, the large bounding box is expected to be centered at the touched key, around the fingertip. We train a small bounding box around C and use k-means clustering over this small bounding box to obtain the fingertip contour. First, we convert the region of this small bounding box into a gray scale image and increase its contrast. Next, k-means clustering is then used to cluster the pixel values into two categories: dark and bright. This region of interest is then transformed into a binary image accordingly. The intuition is that the touching finger is brighter than the area around it. Therefore, we are able to find the contour of the fingertip using the bright areas. Figure 7 shows the contour of the fingertip after we process the small bounding box in Figure 6.

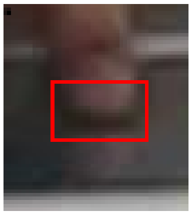

Fig. 6: Small Bounding Box

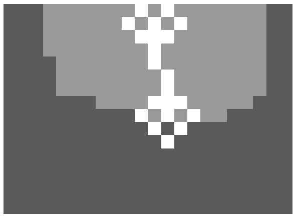

Fig. 7: Fingertip Contour

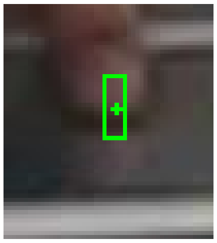

Fig. 8: Touched Area

Once the fingertip is located, we can estimate the top of the fingertip and train a tiny bounding box around this fingertip point as the accurately touched area. To derive the fingertip top, for each horizontal line of pixels of the fingertip contour, we find its central points. We then fit a line over these central points. This is the central line of the finger image and also indicates the finger's direction. The intersection between this line and the fingertip contour produces the top and bottom of the fingertip. Figure 7 shows both the estimated top and bottom of the fingertip and its direction. Figure 8 shows a tiny bounding box we trained around the top of the fingertip.

3.7 Step 6 - Recognizing Touched Keys

Although we have derived the tiny and accurate touching area in Figure 8, such an area is still too large and contains non-touching points. From our analysis and experiments, an actual key area contains only tens of pixels. Our goal in Step 6 is to recognize these actual touched points landed in the key area. Once the actual touched points are located, we can then map them to the reference image in Figure 4. The corresponding points in the reference image are denoted as mapped points. Such mapped points should land in the corresponding key area of the software keyboard. Therefore, we can derive the touched keys. This is the basic idea of blindly recognizing the touched keys even if those touched keys are not visible in the video. The key challenge is to accurately locate the touched points in the tiny bounding box.

We now analyze the brightness of the area around the fingertip. Around because of the lighting over the fingertip and the existence of the fingertip's virtual image, we can have five areas with five different types of brightness: bright fingertip top, gray fingertip middle area, dark fingertip bottom and its virtual image (dark fingertip bottom, dark fingertip bottom of the virtual image), gray fingertip middle area of the virtual image, and bright fingertip top of the virtual image.

We can use clustering algorithms to group these five areas of pixels of different brightness. The k-means clustering algorithm is applied to the pixels in the tiny bounding box in Figure 8. The number of clusters is set as 5. The darkest cluster $\mathcal{C}$ indicates the area where the finger touches the screen surface. We automatically select a pixel in the upper half of the darkest cluster as the touched point. This touched point is then mapped to the reference image and the mapped point shall fall onto the correct key. Basically, the clustering algorithm helps *accurately* identify the touched point. As an example, Figure 10 (a) shows the clustered result of the area in the red and tiny bounding box. The green point is the touched point in the upper half of the darkest area. Figure 10 (b) shows the mapped point (in green) that falls into the frontal part of key 5. Therefore, 5 is the touched key. Please refer to our technical report [25] for the technical details.

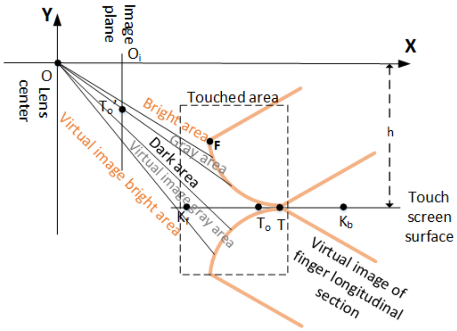

Fig. 9: Five Pixel Groups around Fingertip

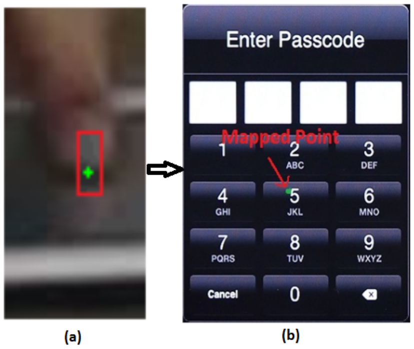

Fig. 10: Clustered Result and Mapped Point

4 Evaluation

In this section, we present the experimental design and results to demonstrate the impact of the blind recognition of touched keys.

4.1 Experiment Design

We have performed extensive experiments on various target devices with different key sizes, including iPads, iPhones, and Nexus 7 tablets. Three cameras are used, including the Logitech HD Pro Webcam C920, the iPhone 5 camera, and Google Glass. Most experiments were performed with the Logitech HD Pro Webcam C920. The last group of experiments is designed for comparing the web camera, iPhone camera, and Google glass. In all experiments, we try to recognize 4-digit passcodes, which are randomly generated. The success rate is defined as the probability that the passcodes are correctly recognized.

In addition to the different cameras and target devices, we consider the impact from the following factors: users, the distance between the camera and target device, and the angle of view of the camera.

- **Users**: Different people have different finger shapes, fingernail lengths, and touching gestures. Five females and six males with the experience of using tablets and smartphones participated in the experiments. They were separated into two groups: 3 people in the first group and 7 people in the second group. These two groups performed the experiments with iPads. The last group helped us to evaluate the success rate versus the distance between the camera and the target. For the first group, we took 10 videos for every person at each angle (front, left, and right of the target device). Thus, 90 total videos were taken. For the second group, five videos were taken for every person per angle and thus, 105 total videos were taken. Combined, 195 videos were taken. During the experiments, users tapped in their own way without any imposed restriction.
- **Angles and Distance**: To measure the impact of the angle, we put the target in front, on the left (3 o'clock), and on the right (9 o'clock) of the camera. In the first two groups of experiments, the camera was 2.1 m to 2.4 m away and around 0.5 m above the device. To test how the distance affects the recognition results, we also positioned the camera (the Logitech HD Pro Webcam C920) in front of the target device (an iPad) at distances of 2 m, 3 m, 4 m, and 5 m away and around one meter above the target.
- **Lighting**: The lighting affects the brightness and contrast of the image. The experiments are performed in a classroom with dimmable lamps on the ceiling. The first group of videos was taken under normal lighting and the second group of experiments was taken under strong lighting conditions. All other experiments were performed under normal lighting. Darkness actually helps the attack as the touch screen is brighter in the dark. We did not consider these easy dark scenes in our experiments.

4.2 Recognizing Touched Keys on an iPad via a Webcam

Table 1 gives the success rate of recognizing touched keys from videos taken at different angles. Recall that the success rate is the ratio of the number of the correctly recognized passcodes (all four digits) to the number of passcodes. For wrong results, we attempt a second try, trying other candidate passcode digits. It can be observed that the overall first-time success rate at different angles is more than 80 %. The second time success rate is higher than the first time success rate and is over 90 %.

Table 1: Clustering Based Matching

	Front	Left	Right	Total
First Time	92.18%	75.75%	79.03 %	82.29%
Second Time	93.75%	89.39%	90.32%	91.14%

The second try is performed thusly. We often see one or two wrong keys in the failed experiments. Some of these wrong keys are caused by a DPM that fails to detect the touched area. Sometimes, even if the touched area is detected, the image can be so blurry that the pixels around the touched area have almost the same color and it is difficult to derive the fingertip contour as shown in Figure 7. Other fingers may also block the touching finger and as a result, incur wrong recognition of the touching fingertip top. Therefore, we often know which key might be wrong and attempt a second try as follows. We may manually select the small bounding box of the fingertip in Figure 6 or the touched area in Figure 8 to correct some errors. From our analysis, for each touch, we may also produce two candidates. Using one of the two choices, we may correct the wrong keys for the second attempt. Hence, the second time success rate is higher than the first time success rate.

Figure 11 presents the results of measuring the impact of the distance between the camera and the target on the success rate. It can be observed that as the distance increases, the success rate decreases. At a distance of 4 m or 5 m, the first time success rate is as low as 20 %. This is because at such a distance, the keys in the image are so small that they are only 1 or 2 pixels wide. Although, it is much more difficult to distinguish a touched key at such a distance, a camera with a high optical zoom may help. Nonetheless, our threat model does not allow for the use of these high zoom cameras.

4.3 Comparing Different Targets and Cameras

To compare the success rate of recognizing touched keys on different devices, we perform thirty experiments on Nexus 7 and iPhone 5 respectively with the Logitech HD Pro Webcam C920 from two meters away and about 0.65 m above the device. To compare the success rate achieved by different cameras, we conducted thirty experiments on iPhone 5 recording passcode inputs on iPad from a similar distance and at a similar height. Thirty experiments using Google glass were performed by recording passcode inputs on iPad two meters away and at a human height. Figure 12 presents the results. It can be observed: (i) in all cases, the first time success rate is more than 80 % and

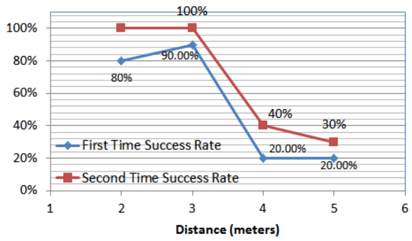

Fig. 11: Success Rate v.s. Distance

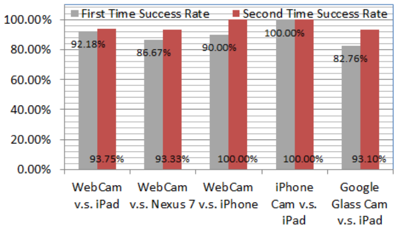

Fig. 12: Success Rate Comparison

the second time success rate is more than 95 %. (ii) iPhone as the spying device produces the best success rate. (iii) When the target is an iPhone instead of an iPad, we can still use the webcam to produce a first time success rate of more than 90 %. These observations further demonstrate the severity of the attack investigated in this paper.

5 Countermeasures

We now discuss countermeasures to computer vision based attacks investigated in this paper and related work. There are a number of authentication approaches immune to these attacks to some extent, including biometric-rich gesture-based authentication [26,27,28] and graphic password schemes [29,30,31]. The idea of a randomized keyboard has been proposed for legacy keypads and touch-enabled devices [32,33,34,35,36,37,38]. We have designed and developed context-aware Privacy Enhancing Keyboards (PEK) for Androids for the first time. PEK pops up a randomized keyboard on Android systems for sensitive information such as password inputs and shows a conventional QWERTY keyboard for normal inputs such as email messages. We have implemented PEK as a third party app and are also able to convert the internal system keyboard into PEK.

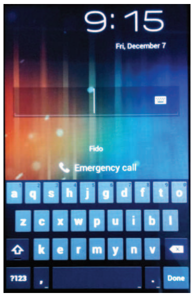

Fig. 13: PEK: Shuffled Keys - the key layout is changed for randomizing keys.

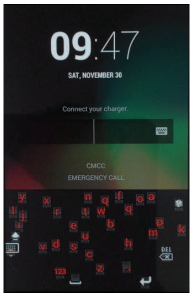

Fig. 14: PEK: Brownian Motion - keys move in a Brownian motion fashion.

6 Conclusion

In this white paper, we presented a computer vision based attack that blindly recognizes inputs on a touch screen from a distance automatically. The attack exploits the homography relationship between the touching images (in which fingers touch the screen surface) and the reference image of a software keyboard. We used the optical flow algorithm to detect touching frames. The Deformable Part-based Model (DPM) and various computer vision techniques were used to track the touching fingertip and identify the touched area accurately. We carefully analyzed the image formation of the touching fingertip and designed the k-means clustering strategy to recognize the touched points. Homography is then applied to recognize the touched keys. We performed extensive experiments and the results showed that the first time success rate is more than 80 % and the second time success rate is more than 90 %. As a countermeasure, we designed a context aware Privacy Enhancing Keyboard (PEK) that pops up a randomized keyboard on Android systems for inputting sensitive information such as passwords. Our future work includes further refinement of the attack and design of alternative authentication strategies for mobile devices.

Acknowledgement

We thank Yang Zhang from University of Minnesota Twin Cities for the implementation of the PEK - Brownian Motion, and Yiqi Bai as the female model.

References

1. Juniper Networks, Inc.: Juniper networks third annual mobile threats report. **http://www.juniper.net/us/en/local/pdf/additional-resources/3rd-jnpr-mobile-threats-report-exec-summary.pdf** (2013)
2. Backes, M., Dürmuth, M., Unruh, D.: Compromising reflections or how to read lcd monitors around the corner. In: Proceedings of IEEE Symposium on Security and Privacy (S&P). (2008) 158–169
3. Backes, M., Chen, T., Duermuth, M., Lensch, H., Welk, M.: Tempest in a teapot: Compromising reflections revisited. In: Proceedings of 30th IEEE Symposium on Security and Privacy (S&P). (2009) 315–327
4. Balzarotti, D., Cova, M., Vigna, G.: Clearshot: Eavesdropping on keyboard input from video. In: Proceedings of the 2008 IEEE Symposium on Security and Privacy (S&P). SP'08 (2008) 170–183
5. Maggi, F., Gasparini, S., Boracchi, G.: A fast eavesdropping attack against touchscreens. In: Proceedings of 2011 7th International Conference on Information Assurance and Security (IAS). (2011) 320–325
6. Raguram, R., White, A.M., Goswami, D., Monrose, F., Frahm, J.M.: ispy: automatic reconstruction of typed input from compromising reflections. In: Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS). (2011) 527–536
7. Xu, Y., Heinly, J., White, A.M., Monrose, F., Frahm, J.M.: Seeing double: Reconstructing obscured typed input from repeated compromising reflections. In: Proceedings of the 20th ACM Conference on Computer and Communications Security (CCS. (2013)

8. Koch, J.: Codescrambler. **http://cydia.saurik.com/package/org.thebigboss.codescrambler/** (2014)
9. Bradski, G.R., Kaehler, A.: Learning opencv, 1st edition. First edn. O'Reilly Media, Inc. (2008)
10. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. IEEE Transactions on Pattern Analysis and Machine Intelligence **32** (2010) 1627–1645
11. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) - Volume 1 - Volume 01. CVPR '05, IEEE Computer Society (2005) 886–893
12. Plugable: Plugable usb 2.0 digital microscope for windows, mac, linux (2mp, 10x-50x optical zoom, 200x digital magnification). **http://www.amazon.com/Plugable-Digital-Microscope-Windows-Magnification/dp/B00AFH3IN4/ref=sr_1_1?ie=UTF8&qid=1382796731&sr=8-1&keywords=optical+zoom+webcam** (2013)
13. Yilmaz, A., Javed, O., Shah, M.: Object tracking: A survey. ACM Comput. Surv. **38**(4) (December 2006)
14. Kalal, Z., Mikolajczyk, K., Matas, J.: Tracking-learning-detection. IEEE Transactions on Pattern Analysis and Machine Intelligence **34**(7) (July 2012) 1409–1422
15. Szeliski, R.: Computer Vision: Algorithms and Applications. 1st edn. Springer-Verlag New York, Inc. (2010)
16. yves Bouguet, J.: Pyramidal implementation of the lucas kanade feature tracker. Intel Corporation, Microprocessor Research Labs (2000)
17. Shi, J., Tomasi, C.: Good features to track. Technical report (1993)
18. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. 2 edn. Cambridge University Press (2003)
19. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision **60**(2) (November 2004) 91–110
20. Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-up robust features (surf). Journal of Computer Vision and Image Understanding **110**(3) (June 2008) 346–359
21. Muja, M., Lowe, D.G.: Fast approximate nearest neighbors with automatic algorithm configuration. In: Proceedings of VISAPP International Conference on Computer Vision Theory and Applications. (2009) 331–340
22. Huber, P.: Robust Statistics. John Wiley & Sons (1981)
23. Canny, J.: A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence **8**(6) (1986) 679–698
24. Matas, J., Galambos, C., Kittler, J.: Robust detection of lines using the progressive probabilistic hough transform. Journal of Computer Vision and Image Understanding **78**(1) (2000) 119–137
25. Yue, Q., Ling, Z., Fu, X., Liu, B., Yu, W., Zhao, W.: My google glass sees your passwords!, **http://www.cs.uml.edu/~xinwenfu/paper/VisionBasedAttack_Fu_2014.pdf**. Technical report (2014)
26. Sae-Bae, N., Ahmed, K., Isbister, K., Memon, N.: Biometric-rich gestures: A novel approach to authentication on multi-touch devices. In: Proceedings of the 30th ACM SIGCHI Conference on Human Factors in Computing Systems (CHI). (2012)
27. Yan, Q., Han, J., Li, Y., Zhou, J., Deng, R.H.: Designing leakage-resilient password entry on touchscreen mobile devices. In: Proceedings of the 8th ACM Symposium on Information, Computer and Communications Security (AsiaCCS). (2013)
28. Kim, D., Dunphy, P., Briggs, P., Hook, J., Nicholson, J.W., Nicholson, J., Olivier, P.: Multitouch authentication on tabletops. In: Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI). (2010)

29. Biddle, R., Chiasson, S., van Oorschot, P.: Graphical passwords: Learning from the first twelve years. In: ACM Computing Surveys. (2012)
30. Suo, X., Zhu, Y., Owen, G.S.: Graphical passwords: A survey. In: Proceedings of Annual Computer Security Applications Conference (ACSAC). (2005)
31. Bulling, A., Alt, F., Schmidt, A.: Increasing the security of gaze-based cued-recall graphical passwords using saliency masks. In: Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI). (2012)
32. Hirsch, S.B.: Secure keyboard input terminal. In: United States Patent No. 4,333,090. (1982)
33. Hirsch, S.B.: Secure input system. In: United States Patent No. 4,479,112. (1982)
34. McIntyre, K.E., Sheets, J.F., Gougeon, D.A.J., Watson, C.W., Morlang, K.P., Faoro, D.: Method for secure pin entry on touch screen display. In: United States Patent No. 6,549,194. (2003)
35. Hoanca, B., Mock, K.: Screen oriented technique for reducing the incidence of shoulder surfing. In: Proceedings of the International Conference on Security and Management (SAM). (2005)
36. Shin, H.S.: Device and method for inputting password using random keypad. In: United States Patent No. 7,698,563. (2010)
37. Lee, C.: System and method for secure data entry. In: United States Patent Application Publication. (2011)
38. Kim, I.: Keypad against brute force attacks on smartphones. In: IET Information Security. (2012)