Seeing Double: Reconstructing Obscured Typed Input from Repeated Compromising Reflections

Yi Xu, Jared Heinly, Andrew M. White, Fabian Monrose and Jan-Michael Frahm
Department of Computer Science, University of North Carolina at Chapel Hill
Chapel Hill, North Carolina, USA
{yix,jheinly,amw,fabian,jmf}@cs.unc.edu

ABSTRACT

Of late, threats enabled by the ubiquitous use of mobile devices have drawn much interest from the research community. However, prior threats all suffer from a similar, and profound, weakness — namely the requirement that the adversary is either within visual range of the victim (e.g., to ensure that the pop-out events in reflections in the victim's sunglasses can be discerned) or is close enough to the target to avoid the use of expensive telescopes. In this paper, we broaden the scope of the attacks by relaxing these requirements and show that breaches of privacy are possible even when the adversary is around a corner. The approach we take overcomes challenges posed by low image resolution by extending computer vision methods to operate on small, high-noise, images. Moreover, our work is applicable to all types of keyboards because of a novel application of fingertip motion analysis for key-press detection. In doing so, we are also able to exploit reflections in the eyeball of the user or even repeated reflections (i.e., a reflection of a reflection of the mobile device in the eyeball of the user). Our empirical results show that we can perform these attacks with high accuracy, and can do so in scenarios that aptly demonstrate the realism of this threat.

Categories and Subject Descriptors: K.4.1 [Computers and Society]: Privacy

General Terms: Human Factors, Security

Keywords: Compromising Emanations; Mobile Devices

1. INTRODUCTION

Gone are the days when mobile phones were used exclusively for voice communication. Today, as these handheld devices have become more sophisticated, they are routinely used for a myriad of everyday activities that include checking email, text messaging, performing financial transactions, and finding directions to a location of interest. Inevitably, as our day-to-day reliance on these devices increases, the sensitive information (e.g., passwords) input on these devices becomes increasingly valuable to prying eyes.

While the academic community has long acknowledged that the ubiquity of these devices provides new opportunities for privacy abuse (as users communicate private data in ways more vulnerable to eavesdropping than ever before), the severity of the threat posed by advancements in computer vision techniques is only now being well understood [3, 34]. As a case in point, both Raguram et al. [34] and Maggi et al. [29] recently showed that modern touch-screen smartphones may offer a greater privacy threat than their traditional counterparts. The increased risk comes from the fact that many touch-screen smartphones utilize virtual keyboards that overcome the perceived lack of tactile feedback by providing users with visual confirmation (a key "pop-out" effect) as a key is pressed. These effects, however, provide strong visual cues that can be exploited by an attacker to help identify the keys tapped on the victim's device.

The techniques used to leverage the so-called compromising reflections in these prior works have raised our collective awareness of the realism of these threats. However, they all suffer from a similar, and profound, weakness — namely the requirement that the adversary is either within visual range of the victim (e.g., to ensure that the pop-out events in reflections in the victim's sunglasses can be discerned [34]) or is close enough to the target to avoid the use of expensive telescopes [3].

In this paper, we push the limits of these attacks by exploiting even more fundamental, and harder to conceal, observable events. That is, unlike prior work, we do not rely on the attacker's ability to capture detail (e.g., a key pop-out event) on the screen, but instead target a common factor in user interaction with mobile devices: the

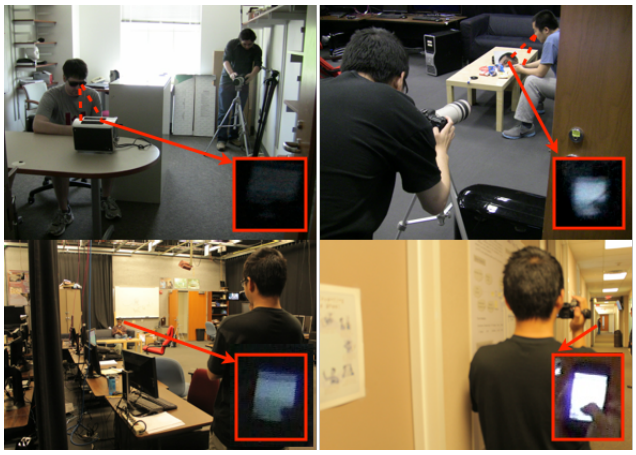

Figure 1: Some example threat scenarios that we investigated. Top left: Through reflection from sunglasses and toaster; Top right: Through reflection from eyeball and mirror; Bottom left: Through reflection from sunglasses; Bottom Right: Direct long-distance view. Note: As with the remaining figures in this work, this image is best viewed in color.

CCS'13, November 4-8, 2013, Berlin, Germany.
ACM 978-1-4503-2477-9/13/11.
http://dx.doi.org/10.1145/2508859.2516709 .

relationship between a user's fingers and the keyboard. By tracking the positions of a user's fingers as they move across the virtual keyboard, we can successfully reconstruct the typed input. In particular, we show that even using inexpensive consumer devices (e.g., small hand-held camcorders), we can successfully perform a reconstruction attack as long as both the device's orientation and the user's fingers are detectable.

Tracking fingers, rather than recognizing displayed images, broadens the class of vulnerable devices to include those without any pop-out effect, such as Apple's iPad. Moreover, our approach is capable of operating at significant distances from the victim (e.g., up to 50 meters away with a camcorder). Perhaps most importantly, our approach operates effectively even for *repeated reflections*, i.e., *reflections of reflections* in nearby objects. This feat allows us to reconstruct typed input from the image of a mobile phone's screen on a user's eyeball as reflected through a nearby mirror (see Figure 1), extending the privacy threat to include situations where the adversary is located *around a corner* from the user.

Our approach operates despite low resolution images (a natural consequence of increasing the distance between target and camera) and high noise levels (due to reduced light levels in reflections of objects as compared to the originals). To overcome these challenges and achieve our goals, we extend a large body of work on object detection and tracking in the area of computer vision. Specifically, we extend existing finger tracking mechanisms to consider spatial context in order to reliably locate fingertips in the images. In addition, we propose a novel method for identifying fingertip trajectories based on robust estimation.

2. RELATED WORK

Techniques for undermining a user's privacy via several forms of compromising emanations have a rich and storied history (e.g., [1–3, 6, 7, 12, 16, 21, 29, 33, 41, 43, 49]). Probably the earliest of these ideas is embodied in the work of van Eck [41] and Highland [16] on compromising signals from electromagnetic radiation. That work was later extended by Kuhn and Kuhn [21], wherein it was argued that a telescope could be used to spy on reflections of a computer screen from afar. Intuitively, the light emitted from the display was modeled as a function of time and space, and then analyzed using signal processing methods. Naturally, the captured light is a disturbed signal of the light emitted by the screen, where disturbances include atmospheric effects, dispersion, attenuation, or lens distortion of the capture device. Regardless of these disturbances, Kuhn and Kuhn [21] show that by utilizing an expensive telescope with a wide aperture, they were able to reconstruct text on a 32×24 cm display from 60 m away.

More recently, Backes et al. [2, 3] overcame the requirement of direct line-of-sight. The key innovation was to exploit reflections to vary the path of light between the target screen and the observer, and showed it was possible to leverage the reflection off a human eyeball (reading a very large 36 pt text font from a distance of 10 m). However, the approach still used a high-powered telescope inheriting the drawbacks of high cost along with limited versatility and ability to go undetected. In addition, the setting of Backes et al. did not have to consider motion (of either the victim's device or the adversary), and also was not concerned with the daunting task of automatically reconstructing text from typed input.

The use of less expensive and more practical equipment was introduced by Raguram et al. [34]. Unlike the approach we present, that method relies on detecting the presence of key pop-outs in virtual keyboards. While that approach worked well for direct line-of-sight attacks, reconstructions involving a single reflection (in this case, off the victim's sunglasses) did not perform as well [34]. A related approach that also relied on the ability to detect pop-outs was proposed by Maggi et al. [29]. However, the approach of Maggi et al. is sensitive to movement of the device and camera and suffers in the presence of occlusions (including the fingers of the device's user). Neither approach could handle reconstructions of low resolution images of reflections of reflections in nearby objects.

Within the computer vision and human computer interaction communities, work on finger tracking for gesture recognition [8, 32], virtual input systems [40, 48], virtual object manipulation [25, 26], and hand writing recognition [19, 46] all share similarities to our application of finger motion analysis. Probably the most germane of these is the work of Iturbe et al. [18] which uses finger movement to control a virtual input system in which a user's finger is modeled as a straight line and its movement is used to determine the button being pointed to by the user. Unfortunately, their approach quickly fails for small mobile devices where the fingers need to bend in order to reach the keys. Similarly, Jin et al. [19] utilized finger input for character writing recognition. In their approach, the user's finger is isolated using a background modeling technique. Next, the path taken by the finger is tracked as the user writes individual letters, effectively recognizing the letter through the trajectory of the finger. Sadly, the approach of Jin et al. [19] can not be directly applied to mobile devices as users do not spell words by forming one character at a time, but instead interact with a keyboard via a series of touch events.

Lastly, Kerdvibulvech and Saito [20] apply a novel technique for tracking the fingers while a user plays a guitar. Instead of trying to uniquely identify each individual finger, the authors use a neural network classifier to recognize known patterns corresponding to different chord formations. While promising, their approach is also not applicable in our setting, as the way users type on mobile devices can differ significantly for the same user (e.g., switching between typing with one, two, or several fingers), and even more among different users, making the learning strategy less practical. Nevertheless, as we show later, we found that by combining many of the strengths of prior works alongside our own enhancements, we are able to achieve a solution that surpasses previous attempts in terms of its practically, range, and robustness.

3. BACKGROUND

Before introducing our approach, we provide pertinent background information that is helpful in understanding the set of challenges that impact how well an adversary can observe reflected objects in a scene.

Obviously, the size of the object in the captured images is of critical importance and naturally depends on the size of the object itself. Another factor is the focal length of the camera. Loosely speaking, the size of the object in the image for direct line-of-sight can be computed as:

$$Size_{Direct} = \underbrace{\frac{Sensor\,Resolution}{Sensor\,Size}}_{\text{pixel scale}} \cdot \underbrace{\frac{Object\,Size}{\frac{Target\,Distance}{Focal\,Length} - 1}}_{\text{size on sensor}} \qquad (1)$$

Intuitively, the observed size is dependent on the physical size of the projection of the object on the sensor and the characteristics of the camera sensor, namely the size and number of pixels (picture elements). The size of the object on the sensor (its projection) is controlled by the focal length, the distance to the object, and the object size. Focal length can be viewed as the amount of magnification of the image, where longer focal lengths (zoom lenses) provide higher magnifications. Thus, by using a lens with a longer focal length, an adversary can gain a better view of the target. The

final size of the image of the device in pixels (given the image's size on the sensor) then depends on the ratio between how many pixels are on the image sensor (*SensorResolution*) and the physical size of that sensor (*SensorSize*). At the same focal length, for example, the size of the object in pixels tends to decrease with full frame sensors found in high-end digital SLRs compared to cheaper digital camcorders with smaller sensors but the same video resolution.

In addition to the physical object size, the size of the object on the image sensor also depends on the presence and shape of any reflecting surfaces between the observer and the object. For instance, if the reflecting object is convex (e.g., a pair of sunglasses or the human eyeball), the size of the observed object will be much smaller than if observed with direct line-of-sight. When an object is viewed via a reflection, the final observed size can be computed as:

$$Size_{Reflection} = Size_{Direct} * \frac{1}{\frac{2\,Distance\,from\,Surface}{Curvature\,Radius} + 1} \quad (2)$$

Thus, the curvature of the reflecting surface is an important factor in the observed size. The more curved the reflecting surface is, the more the light will be bent. For convex surfaces, the bending of the light will result in a smaller observed object size. Lastly, the distance between the reflecting surface and the target object itself (*Distance from Surface*) naturally affects the observed object size.

Takeaway. One way to acquire a larger observed size is to simply reduce the distance to the target object. However, from an adversarial point of view, it is desirable to be as far away as possible from the victim. Hence, a better solution would be to use a lens with a long focal length. For similar reasons, cameras with higher pixel density in their sensors are preferred. Finally, the curvature of any reflecting surface must be taken into account. For example, the human eyeball has a typical curvature of about 8 mm [2]. Hence, when people look at an object 25 cm away, the reflection in their eyeball will be about 60 times smaller than with direct line-of-sight.

Impact of Diffraction. The quality of the acquired image is significantly influenced by the wave properties of the light. When light comes near the edge of the lens, not all light rays traveling from the object pass directly through the lens to the image sensor. Instead, a fraction of the light diffuses and travels in every direction, leading to a blurring of the image. This phenomenon is called diffraction and cannot be eliminated. It presents a physical boundary for the effective resolution of the object in the captured image (commonly referred to as the *Rayleigh Criterion* [2]).

The maximum effective size of the observed object (*Max Size*) can be approximated as:

$$Max\,Size = \frac{Aperture/Wavelength}{1.22\,Target\,Distance} * \frac{Object\,Size}{\frac{2Distance\,from\,Surface}{Curvature\,Radius} + 1} \quad (3)$$

Notice that the actual amount of diffraction is impacted by the wavelength of the light (*Wavelength*). While the adversary has no control over this factor, we include it here for completeness (along with the well-known constant scale factor of $1/1.22$ for circular apertures [37]). However, the adversary can select a lens with an appropriate aperture (i.e., opening of the lens), which lets the desired amount of unobstructed light pass through the lens.

Takeaway. The larger the aperture of the lens, the smaller the amount of diffraction. It is for precisely this reason that prior work (e.g., [2, 3, 21]) resorted to the use of telescopes. However, lenses with large apertures are typically very expensive, costing well over $1,000 per square cm [2], and are difficult to conceal.

Impact of Noise. A very significant factor that affects the quality of the acquired image of the target object is imaging noise. Noise is a random variation in a pixel's intensity, causing the image to appear speckled. As noted by Nakamura [30], there can be several types of background noise in the captured image, each with a constant noise level. To avoid visual impact on the image quality by the noise, the exposure time is typically chosen so that the overall amount of light overwhelms the background noise making it hardly noticeable. However, for video capture, the exposure is naturally limited to the time of a frame, which for darker scenes makes the background noise become more noticeable. The resulting noise causes significant challenges in identifying fine detail.

Typically, cameras with large sensors are more resistant to noise, as their pixels are usually larger and can capture more photons of light. For that reason, the larger sensors provided in digital SLR cameras (as opposed to cellphones or point-and-shoot cameras) are desirable for photography even though they provide a smaller number of pixels on the object.

Taken as a whole, the aforementioned factors present challenges that *severely* limit the use of existing techniques (e.g., [2, 3, 34, 35]) when considering reconstruction of typed input from repeated reflections. For instance, the recently used technique of identifying salient feature points [34, 35] within the image in order to facilitate alignment will fail because the poor image resolution does not provide the required details for the salient feature points. The approach suggested by [29] faces similar challenges. Additionally, the low image quality (e.g., as acquired from a reflection in the eyeball) prevents the detection of fine detail. Therefore, the key pop-out events exploited by previous work will no longer be distinguishable.

4. AUTOMATED TRANSCRIPTION

Our proposed method successfully transcribes the text typed on a keyboard by exploiting video of the user typing (either directly or through up to two reflections from nearby objects). In practice, we must overcome all of the aforementioned challenges of image resolution, diffraction, and noise. To do so, we devise techniques that are resistant to low resolution, blurring, and noise. Figure 2 shows a high-level depiction of our approach.

We take as input a recording of the target device while the user types on its keyboard. First, we roughly estimate the location of the device in the image (Stage ❶). Next, the image of the device is aligned to a known reference template of the device's keyboard layout (Stage ❷). Then, the fingertips are identified in the video and the locations of the fingertips over the video frames are combined into trajectories (Stage ❸). These trajectories are then analyzed to identify the pressed keys (Stage ❹). From these pressed keys we then reconstruct the typed text (Stage ❺). Finally, as an optional post-processing step, we apply a language model in order to improve the readability of the final text (Stage ❻). Aside from some initial input in Stage ❶ to identify the first frame containing the target device, the entire process is fully automated. In what follows, we discuss each step in turn.

Stage ❶: Tracking

With a recording in hand, we first identify the device (phone, iPad, etc.) in the video so that we can focus our remaining analyses on only the relevant parts of the image. Depending on the number of reflections and the distance between the victim and the observer, the device is often only seen in a small region of each video frame (refer to Figure 3 for an example of reflection in an eyeball).

In order to identify the device's location in every frame, we utilize a tracking framework based on AdaBoost [15]. The basic idea is as follows. First, the user selects the region of the first video

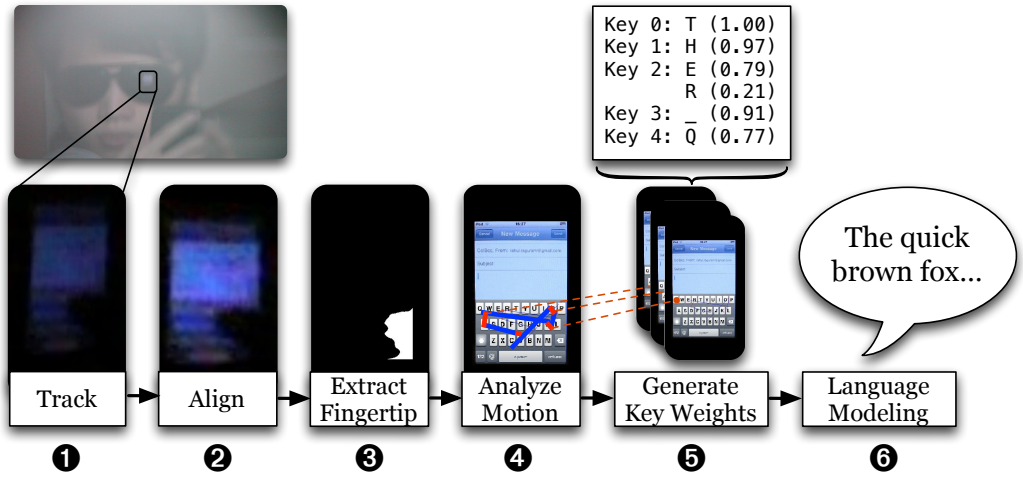

Figure 2: Overall design depicting the stages of our approach. First we track the motion of the mobile device (Stage ❶). Then the mobile device is aligned to a template (Stage ❷). In the stabilized image the finger is extracted (Stage ❸) and its fingertip trajectory is computed (Stage ❹). From the trajectories the likely pressed keys are identified (Stage ❺). As an optional step, we apply a language model to improve the quality of the reconstructed text (Stage ❻).

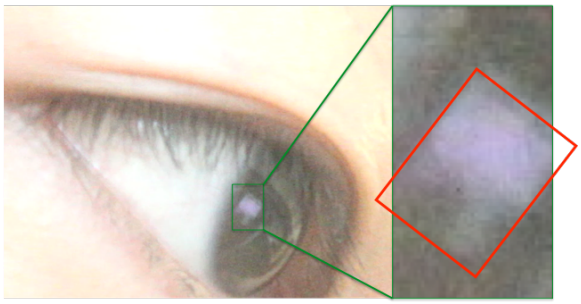

Figure 3: The captured image showing a reflection (of the victim's screen) as observed in the eyeball.

frame (corresponding to the device) that should be localized. Given this initial position, AdaBoost learns the appearance of the target object, and then automatically determines its location in all succeeding video frames. Since AdaBoost tracking works relatively well even in the presence of noise, we can successfully track small or even partially occluded objects in the image. The latter is important because the device is usually occluded by the fingers as the victim types on the keyboard.

That said, despite its robustness to noise, a straightforward application of AdaBoost tracking frequently fails in our setting. This is because tracking relies on the target device maintaining a similar appearance between frames, which is not necessarily the case with the noisy images we must process. In order to mitigate this problem, we average each video frame with the temporally preceding and trailing frame in the sequence. In this manner, each image is the combination of three consecutive frames, which reduces noise as random variation will be smoothed.[1] Although averaging could cause motion blur if there is significant scene motion between the frames, we found that in practice this is not a significant issue because of the high frame rate (30 frames per second) and the small motion of both the device and the user between frames.

[1]Gaussian noise is reduced by a factor of $\sqrt{3}$.

Stage ❷: Alignment

Stage ❶ acquired the rough position of the device in each frame. The task in Stage ❷ is to determine the exact location of each key in the video frame (so that later stages can identify the keys which were most likely pressed). In general, the device will have varying orientation, scale, and image position in the video. In order to determine a mapping between the image in the video and a reference keyboard layout of the device (lifted, for example, from a manual), we must estimate the transformation that will align the device in the video to a reference image of the device's layout.

Prior work [29, 34] faced a similar challenge, and used salient points in the image as visual features to determine the transformation. In our setting, however, those visually distinct salient feature points are no longer visible. Thus we must overcome this challenge in a different manner. The approach we take is to utilize a line detection technique that provides a more robust set of features to achieve the same goal.

Preliminary Alignment: As an initial step, we first reduce the noise present in the images by employing anisotropic filtering [45]. This technique (detailed in Appendix A) can be used to intelligently preserve line structure in the image (otherwise often lost in normal noise reduction techniques), while simultaneously suppressing noise in the remaining parts of the image.

To determine the correct orientation of the device, we transform the video image so that all of the device's lines become either horizontal or vertical. This is accomplished by detecting edges within the image [11], and then using the lines to vote for the device's orientation using a Hough transform. The Hough transform is a voting scheme, where each detected line in the image votes for all possible orientations of the device that conform with the line's own orientation. For robustness, our voting scheme exploits the fact that lines of the same orientation will have a common vanishing point; hence each line is voting for its corresponding vanishing point (see Appendix B). Then, the vertical and horizontal vanishing points

with the most votes represent the dominant vertical and horizontal line directions in the image. With this information at hand, we can successfully transform the device to its proper orientation.

Note, however, that even with the correct orientation, the image is still not aligned with the reference keyboard layout (i.e., the template) as the size and position may still differ. To handle this, we once again turn to a Hough transformation to vote on the translation and scale for aligning the image to the reference image of the device. The voting process relies on correspondences between the detected lines in the current frame and the lines in the template. A descriptor is built for each line in the current frame and the reference image, representing the appearance of each line's surroundings. The lines then vote for possible translation and scaling settings, and their votes are weighted by the similarity of the lines' descriptors. This voting results in a translation and scale in each of the x and y directions. Although the scale should be equal in both directions, we allow different scales (allowing affine transformation) to overcome small artifacts of slightly misaligned rotations. Appendix C details the computed transformation.

Refinement: A result of the Hough transform above is that the voting space will be quantized, leading to small residual misalignments. To provide an alignment that is stable across multiple frames and is as precise as possible we include a final refinement step (termed dense alignment [4, 38]). As opposed to relying on only lines, this step considers all pixels in the image to compute the final alignment. We employ a non-linear optimization in the parameters of the alignment (rotation, scale, translation) using the sum of squared differences between the overlapping pixels of the current and the first frame as an error metric for the residual misalignment. The final alignments are more stable and accurate (to sub-pixel precision) and are thus better suited for further analysis. An example alignment is shown in Figure 4.

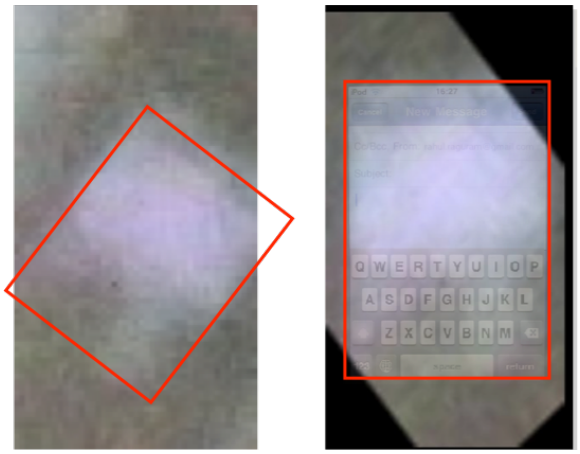

Figure 4: Cropped image (left) and alignment result (right) from Stage ❷ (overlaid with a reference device layout).

Stage ❸: Fingertip Extraction

At this point, the images are now aligned to the reference layout, and we are now primarily concerned with identifying the fingers as they interact with the device.

Similar to Jin et al. [19] and Nguyen et al. [31], we leverage Gaussian modeling of the appearance of the fingers as well as the appearance of the device. Intuitively, at this point, we consider the keyboard area in our input image as consisting primarily of two main colors: a lighter color corresponding to the bright screen, and a darker color corresponding to the user's fingers. Gaussian modeling allows us to represent their respective appearance by two different Gaussian distributions. The idea is to assign each pixel in the keyboard area of the image to either one of the two distributions based on whichever has the highest likelihood for the pixel under consideration. To learn the different distributions, we analyze the pixel colors of the first $n = 50$ frames.

In essence, by assigning each pixel to one of the two Gaussian distributions we convert the input image into a binary image (e.g., the right image in Figure 5). This strategy effectively segments the fingers and the device's screen and allows us to isolate the position and orientation of the fingers. To isolate the position, we simply model the user's finger as an ellipse, and then attempt to fit an ellipse to the boundary of the segmentation. In practice, this means that we must identify points of high curvature along the boundary of the finger's segmentation and then apply an ellipse-fitting regression function to determine the best fit (see Figure 5).

To identify the location of the tip of the finger, we assume that the fingertip corresponds to one of the four apexes of the ellipse. In practice, only three of the four apexes are reasonable fingertip locations because the finger only covers the keyboard from the left, right, or bottom. To determine which of the three cases is present, we consider the intersection of the ellipse with the device's boundary and select the appropriate apex. Specifically, if the ellipse intersects with the left and bottom edges of the device, then we assume that the finger is moving into view from the left; if the ellipse intersects with the right and bottom, it moves in from the right, and if the ellipse only intersects with the bottom, then it is moves into view from the bottom. Notice that by repeating the above process several times (and ignoring already identified fingers) we can detect multiple fingers on the screen and locate the fingertip for each.

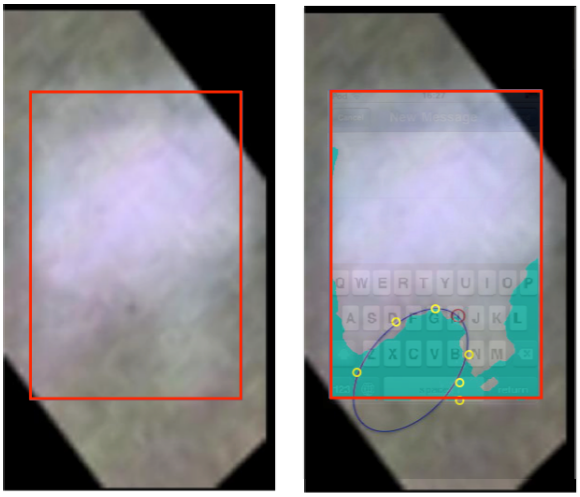

Figure 5: Input aligned image (left) and finger region extraction (right). Shown in the right image is the ellipse fitting (blue ellipse), segmentation (cyan), points with high curvature (yellow), detected fingertip (red circle), and an overlaid reference device layout.

Stage ❹: Fingertip Motion Analysis

We now turn our attention to how we identify the relative position of the fingers when keys are pressed. We take advantage of the insight that when one presses a key, the finger typically hovers at a certain position or suddenly changes direction in order to move to

the next key. That observation allows us to use the location of the fingertip to identify when a key is pressed. To infer the series of likely keys pressed as the victim types, we combine the locations of the fingertips extracted in Stage ❸ into trajectories for each fingertip. The trajectory of a fingertip represents its path through time, and by analyzing this path, we can deduce when a key was pressed.

To accomplish this, we propose a robust method that represents the fingertip trajectories as a curve in 3D space (where the dimensions are the u,v location in the image, corresponding to the x-y plane, and the video frame number). When a fingertip stops moving, the location in the x-y plane will cease to change, but the frame number will still increase. Hence a stopped fingertip describes a line segment in the frame number direction that is perpendicular to the x-y plane and originates at the stopping position u,v.

To find these stopping positions or the direction changes, we approximate the entire 3D curve as a set of line segments. In order to identify the individual line segments, we use a robust model fitting technique called RANSAC [13]. Loosely speaking, the RANSAC algorithm will find a consistent model (a 3D line in this case) in the presence of noisy data even with a small number of correct data points complying with the model. Every time we run RANSAC, we model a part of the 3D curve as a line segment. Next, we remove the previously modeled part of the curve, and focus on the remaining portions. By running RANSAC repeatedly, and removing the modeled parts, we obtain a set of 3D line segments that approximate the original fingertip trajectory.

Given the set of line segments, stopping positions (u,v) are determined by line segments that are nearly perpendicular to the x-y plane. Likewise, sudden changes in direction are detected by finding two adjacent lines with a large change of direction between them. Figure 6 shows an example trajectory and the inferred line segment approximation.

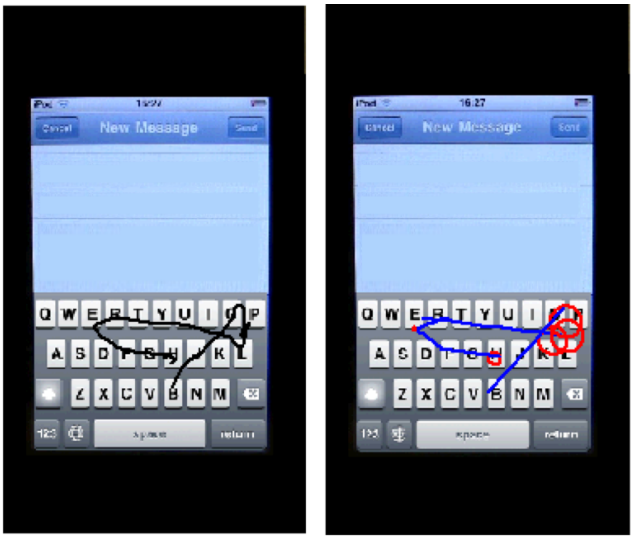

Figure 6: Fingertip trajectory (left) and line modeling (right). The red circles indicate stopping positions for the word "hello", where the size of the circle corresponds to uncertainty in the stopping position.

Stage ❺: Inferring the Keys Pressed

Given the stopping positions inferred in Stage ❹, we now move on to determining what keys were likely pressed. This task is somewhat complicated by the fact that users rarely make contact with the surface of the device with the precise tip of their finger(s). Moreover, when a user types, her fingers will not necessarily be at the same place each time they hit a particular key. Therefore, to determine which key was most likely pressed, we apply image recognition techniques to identify different positions of the user's finger with respect to the keyboard when pressing a key.

To learn the appearance of different key presses, we manually labeled 87 videos, recording the time and position of each key that was pressed. For each key on the keyboard, we collect a sample of its surrounding area as a pixel patch that is 60×100 pixels (matching the aspect ratio of a typical keyboard key). When the key under consideration is pressed, we identify the patch as a positive training sample for that key, and also use it as a negative training sample for all of the other keys on the keyboard. For each of these patches, we extract dense SIFT descriptors [42] and use these to train an AdaBoost classifier. In the end, we have a classifier for each key, where the sole purpose of the classifier is to determine whether or not its associated key has been pressed given an example pixel patch as input.

With this set of classifiers, we then analyze the key-press events detected in Stage ❹. Our analyses show that each event typically lasts anywhere from 10 to 20 frames, and during this time the finger stays at a relatively fixed location. We collect three adjacent frames from the middle of the duration, and extract from each a 60 by 100 pixel patch (corresponding in size to the training patch above) centered at the detected fingertip location. These patches are then fed into each of the nearby classifiers, with each classifier outputting its confidence of a key-press as a value in a range of $[0,1]$ (with 1 being high confidence). By averaging the confidence scores of each classifier (one for each key) from the three chosen frames, we create a set of possible key-presses and their associated confidence levels. If the maximum confidence in the set falls below a predefined threshold ($\delta = 0.5$), we assume the key-press was invalid and discard the event. For a valid event with key-press confidence above δ, we record all the possible keys and their scores. By repeating this process for each key-press event, the final output of this stage is a sequence of key-press sets, where each set has a list of keys and their associated confidence values.

Note also that we do not explicitly detect presses of the "space" button, but instead treat longer pauses between key-press events as the space character. Likewise, shorter pauses are identified as the gaps between different letters in a word. Similar ideas for detecting spaces were utilized by Raguram et al. [34] and Balzarotti et al. [5].

Stage ❻: Language Model

As an optional post-processing step, we can feed the candidate keys and associated confidence values from the previous stage into a language model in order to achieve more accurate and readable results.

Similar to Raguram et al. [34], we view the language modeling task as a *noisy channel decoding* problem and adapt techniques from the speech recognition community to perform this inference (a process known as *maximum likelihood decoding*). In particular, we employ a *cascade* of models, each of which represents a stage in the process of transforming a sequence of candidate character sets with associated confidence values into words and phrases. Each model in the cascade is represented as a *weighted finite state transducer (WFST)*, i.e., a finite state machine with an output tape in addition to the input tape, which maps sequences in one representation to sequences in another. For instance, speech recognition systems often include a component (known as the *lexicon*) which maps sequences of *phones*, the distinct sounds which comprise speech, to (valid) words. A sequence of such component models can then be used to model more complex mappings, such as that from an acous-

tic signal to a sequence of words and phrases, by *composition* of the WFST representations of the component models. With a lexicon as the first component, taking as input a sequence of phones, a second component (often an n-gram language model) maps the resulting sequence of words to likely phrases. The composition of these two models, itself a WFST, maps sequences of phones to likely phrases.

One advantage of the uniform representation of these components as WFSTs is that the *decoding* or inference step (i.e., finding the most likely phrase corresponding to the input sequence of sounds) can be performed on the composition of the component models rather than proceeding sequentially through component models. In other words, traditional speech recognition cascades were effectively Markov chains: the output of each component depended only on the output of the previous component. Often, the resource-intensive nature of the speech recognition task forced the pruning of unlikely sequences, e.g., phones after the application of a component model. Thus a sequence of phones which might be considered unlikely (but not impossible) in the absence of a language model might be pruned in the early stages of such a speech recognition system. A WFST cascade, on the other hand, can be represented as a single WFST, which allows for inference from all components simultaneously and mitigates many problems resulting from pruning.

In our case, the input for the language modeling stage is a sequence of key-press events with character labels and associated confidence values, including marked spaces. We then employ a composite WFST model to smooth over any potential errors in the earlier stages by modifying the output words and phrases to more closely match natural English language. We first apply an edit distance model $\mathscr{E}$ for error correction. Conceptually, the edit distance model produces, for each input string, a set of similar strings each weighted by (keyboard) edit distance from the input string. We then employ a dictionary model $\mathscr{D}$, which prunes those strings which do not result in dictionary words, and an n-gram language model $\mathscr{L}$, which promotes sequences of words that appear more frequently in English and demotes those which appear rarely. Then the cascade model can be represented as $\mathscr{C} = \mathscr{E} \circ \mathscr{D} \circ \mathscr{L}$. To find the most likely sequence of words corresponding to an input sequence of character sets (as output by the previous stage), the input is first represented as a finite state acceptor $\mathscr{I}$. Then the shortest path through the WFST $\mathscr{I} \circ \mathscr{C}$ gives the most likely series of English words given the input sequence of predicted character sets.

Our n-gram language model $\mathscr{L}$ is a unigram model trained on the Brown corpus [14]. The dictionary model $\mathscr{D}$ is based on the 'medium' word list from the Spell Checker Oriented Word Lists[2] with roman numerals, unusual abbreviations, and proper nouns removed. For the keyboard edit distance, we define the distance between two keys as the normalized product of the difference in rows (on a 'vertical' axis through Q and Z) plus one and the difference in keys (on a 'horizontal' axis through A and L) plus one. For instance, the (unnormalized) distance between A and B is $(1+1) * (4+1) = 10$, since A and B are a single row apart vertically and four keys apart horizontally. We offset by one in each case to avoid zero values if the keys are in the same row or column.

5. EVALUATION

Recall that one of the key motivating scenarios driving our design is the ability to reconstruct the typed input from repeated reflections. From an adversarial point of view, we are also interested in understanding the realism of the threat posed by an adversary performing such an attack from as far as possible, yet relying on inexpensive equipment. In what follows, we provide an analysis of our results under these conditions.

[2] `wordlist.sourceforge.net`

Device Name	Focal Length	Aperture	Sensor Size	Resolution
Canon 60D DSLR, 400mm Lens	400mm	F/5.6	22.3 × 14.9mm	5184 × 3456
Canon VIXIA HG21 Camcorder	57mm	F/3.0	4.84 × 3.42mm	1920 × 1080

Table 1: Specifications of the cameras used in our evaluation.

For the case of multiple reflections, we make use of a Canon 60D digital SLR ($700) with 400mm Lens ($1340). For experiments with direct line-of-sight and a single reflection, we use a Canon VIXIA HG21 Camcorder ($1000). These devices are considerably smaller than the telescopes used previously [2, 3, 21] and are also more affordable. The specifications are listed in Table 1.

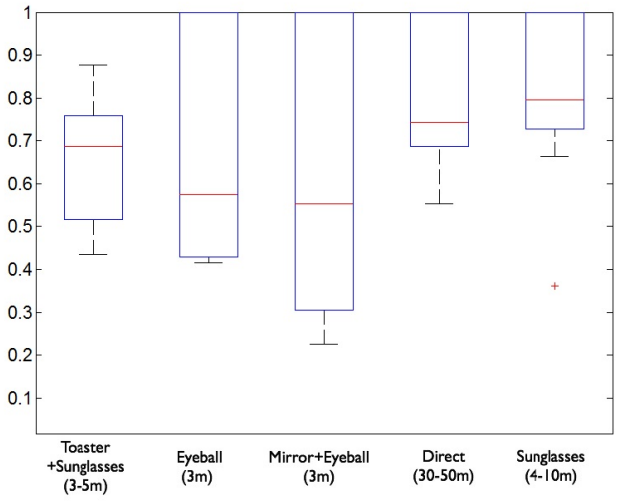

Figure 7: METEOR scores for our approach, broken down by scenario.

For our lab-based evaluation, five different subjects took part in our experiments. To explore different typing conditions, we asked the subjects to perform 3 different styles of typing: typing a specific sentence, providing a quick response to a question (e.g., as one might do when text messaging), and typing a password of their choosing. The chosen sentences are from "The Art of the War" and the questions from SMS messages collected by NUS [9]. In all, we collected 73 responses containing 584 total words. In addition, we collected 15 passwords (each consisting of 5 to 8 lower-case characters).

To gauge our performance, we evaluate the reconstruction results for sentences and passwords using separate metrics. The reason is obvious: the use of a natural language model can easily improve the result of the former, while for passwords a better measure is to examine how many guesses are required to recover the password given our initial hypothesis.

For evaluating the quality of our reconstructed text, we use the METEOR metric proposed by Lavie and Denkowski [24]. Essentially, METEOR scores machine translation hypotheses (i.e., our guesses) by aligning them to one or more reference translations.

Scenario	Distance	Reference Sentence	Reconstructed Results	METEOR Score
Toaster+Sunglasses[C1]	3m	when your round is a short one you take a walk	when your round is a short one he take a walk	0.88
	3m	when it is a long one you take a cab	when a is a long is you the a can	0.43
Mirror+Eyeball[C1]	3m	when it is a long one	when it is a long one	1.00
	3m	i am busy tonight	i at be tonight	0.33
Sunglasses[C2]	4m	if you know your enemy and you know yourself you need not fear the results of a hundred battles	if you know your enemy and you know yourself you need not fear the results of a hundred battles	1.00
	10m	if you know neither the enemy nor yourself you will succumb in every battle	if you his neither the enemy for yourself to will succumb in every battle	0.71
Direct view[C2]	30m	when your round is a short one you take a walk	when your round is a short one you take a walk	1.00
	50m	i plan to stay at home	i men to stay at home	0.74

Table 2: Example reconstructions. [C1]: Using Canon 60D DSLR($700) with 400mm Lens($1340). [C2]: Using Canon VIXIA HG21 Camcorder($1000).

It estimates human perception of the readability of the sentence in terms of the adequacy and fluency of the translation. METEOR assigns scores on a scale from 0.0 to 1.0, with 1.0 indicating a perfect translation. As a guide for interpreting METEOR scores, Lavie [23] suggests that scores of 0.5 or higher indicate understandable translations, while scores of 0.7 or higher indicate good or fluent translations. Examples from our experiments are shown in Table 2.

5.1 Results

The aggregate results of our approach, including confidence intervals, are illustrated in Figure 7. Notice that in every circumstance, our reconstruction results in an average METEOR score above 0.5, indicating an understandable level. Additionally, we reconstructed 23% (17/73) of the sentences perfectly, and 92% (67/73) of all the sentences have a METEOR score above 0.5. All of the captured videos were within the automatic focus and exposure limits of the cameras, leading to no unusable videos which might occur in more challenging environments. We remind the reader that these results are already obtained in situations where previous efforts [29, 34, 35] fail, either due to an increased distance from the target or an additional reflection. These results, as well as the individual examples in Table 2, indicate that our attack is quite stable to changes in the distance to the target and the number and type of reflecting surfaces.

Closer look: Double Reflections

In order to evaluate the performance of double-reflection attacks we conducted the following experiments. First, we position a user with sunglasses at a desk with a nearby toaster. The user then types on an iPhone 4, and an attacker observes the typed input from a concealed location around a corner of 60 to 90 degrees by leveraging the two reflections. For the second setup, the user is once again at a desk, but instead of relying on the reflection from sunglasses, we use the reflection from the user's own eye. To enable this attack, we utilized a small mirror instead of the toaster. The use of the mirror does not influence the image quality but provides the additional ability to attack around the corner. The results without the mirror are similar (Figure 7). These scenarios are illustrated in Figure 8.

The use of multiple reflections naturally limits the information we can acquire from the target. With the double reflection, the main limitation of the system is the contrast between the user's fingers and the device's screen. After two reflections, the device's screen

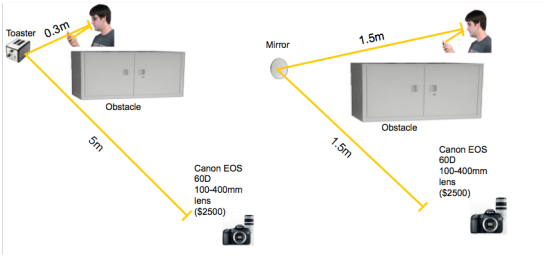

Figure 8: Left: the cellphone image reflects from the user's sunglasses, off the toaster's surface, and is then captured by the camera. Right: the cellphone image reflects from user's eyeball, off the mirror, and then is captured by the camera.

can barely be recognized by a human (see Figure 9). The contrast between the device screen and the background is so low that it is much harder to extract the device screen and finger regions. Yet we are still able to achieve results with an understandable level (0.65). In Figure 10, we illustrate the effect that contrast has on the final result. Reflections significantly reduce the contrast of the image. The RMS contrast (standard deviation of the pixel intensities) takes values in the range $[0.0, 0.5]$, where 0.0 corresponds to an image with uniform color and 0.5 corresponds to a checkerboard pattern of black and white squares. The RMS contrast in our experiments drops from around 0.35 in direct view to 0.03 with two reflections. This leads to difficulties in finger region extraction and therefore lower METEOR scores.

In the reflection off the eyeball (Figure 11), the finger and keyboard are even more difficult to discern. The image is both small and significantly blurred. Backes et al. [2] discuss the theoretical boundary that a telescope with a 62 cm aperture from a distance of 2 m would be required in order to obtain a full resolution image from the reflection of an eyeball. In our setting (i.e., a lens with a 7.1 cm aperture from a distance of 3 m), we achieve understandable reconstruction results with less than $1/10^{th}$ of the full resolution. The main problem with the reflection from the eyeball is the blurring caused by diffraction and noise, as discussed in Section 3, as well as partial occlusion by the eyelashes.

Figure 9: Example double-reflection from sunglasses and a toaster. Left: Captured image; Right: (correctly) identified keypress (`T`, with confidence 0.55). The brightness of the detected image (right) is adjusted for viewing convenience and has been overlaid with a reference device layout.

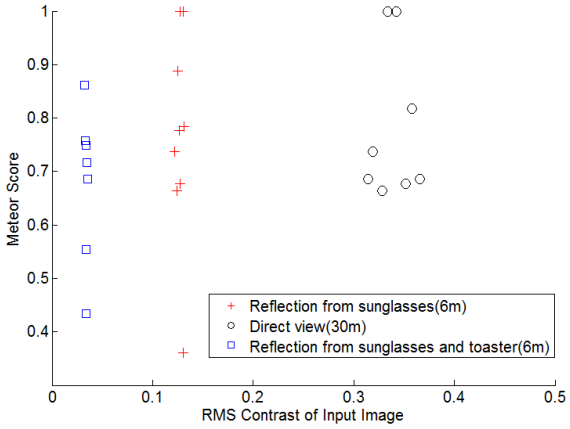

Figure 10: Plot of captured RMS contrast versus METEOR score for the three different scenarios (direct, single-reflection, and repeated reflections).

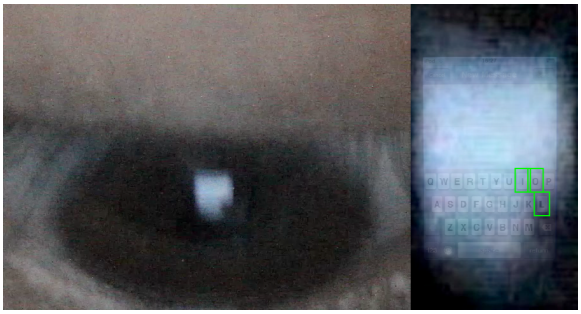

Figure 11: Example double-reflection from an eyeball and mirror. Left: Captured image; Right: candidate keys (`I` with confidence 0.10, `L` with confidence 0.07, and the correct key `O` with confidence 1.00). The right image has been overlaid with a reference device layout.

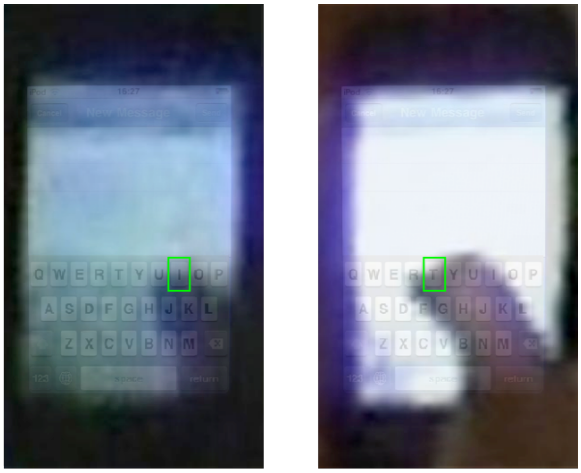

Figure 12: Example single reflection from 10m. Left: Single reflection from 10m (correctly predicting `I` with confidence 1.0). Right: Direct view from 50m (correctly predicting `T` with confidence 1.0). Both images have been overlaid with a reference device layout.

Closer look: Single Reflection and Direct Sight

Compared with the prior state of the art, our attack can be executed from a much farther distance, as illustrated in Figure 12. For example, using the same equipment (Canon VIXIA HG21 Camcorder), the approach of Raguram et al. [35] can only run a direct attack at 24 m and a single-reflection attack at 4 m, while our work increased the distance to be 50 m and 10 m respectively (see Figure 7) while maintaining comparable results.

Devices without Pop-ups

As a final experiment to demonstrate that our design is applicable to a broad range of devices, we include a preliminary study on a device (a first generation iPad) with a virtual keyboard but no pop-out events. In this case, the iPad was placed at a distance of 4 meters from a pair of sunglasses which were in turn situated 4 meters from our camera. As the iPad's screen is considerably larger than the iPhone 4, it is perhaps not surprising that we are able to perfectly reconstruct all 8 sentences tested.

5.1.1 Password Guessing

We also apply our approach to the reconstruction of passwords. While passwords typically chosen by users may follow certain distributions that make them easier to guess, we assume that passwords are chosen with sufficient randomness to mitigate any language modeling, and therefore exclude the optional Stage ❻ from this portion of the analysis. We instead examine the classification results from Stage ❺ directly. One advantage of our approach is that the output of Stage ❺ is a sequence of sets of candidate keys with associated confidence values. We therefore have a natural ordering with which we can prioritize our password guessing strategy.

We implement this ordering with a best-first search, where the 'best' candidate password is the string which maximizes the product of the confidence values for the individual keypresses. We can calculate this product for each child of a given candidate password, where a child is identical to the candidate except that a single character has been replaced with the next most likely character for that position. Beginning with the most likely candidate, i.e., that for which the individual confidence values are maximized, we check if the candidate matches the password for which we are looking. If not, we add each child of the candidate to a priority queue, ordered by the product of the confidence values. Once the children of one

Scenario	Distance	Reference (Typed)	Top Guess	Number of Guesses
Direct Sight	30m	bjtam	bjtam	1
		mstys	matya	47
		dlxmzd	elxmad	64
		zywmxq	atahxq	5967
	50m	wabjta	ewbjta	3
		tatsta	tatata	10
		wdlxmzd	wdiebae	277
Eyeball and Mirror	3m	jrairbf	jfairbf	3
		hvgcjyx	hvccnyx	620
		bgditv	cgcitv	3
Sunglasses and Toaster	3m	sjyfh	sjyfn	2
		aluhe	aauce	14
		kydhwria	kydhwria	1
		jyerpsk	jynraak	21
	5m	hdyeri	hdierx	6

Table 3: Expected number of guesses required to identify user passwords given a ranking, by decreasing confidence, of candidates.

Stage Name	Time Spent (sec/frame)	Percentage
Stage 1: Tracking	0.228	8.4%
Stage 2: Alignment	1.054	38.9%
Stage 3: Finger Extraction	1.042	38.4%
Stage 4: Fingertip Analysis	0.123	4.5%
Stage 5: Key Weight Generation	0.256	9.4%
Stage 6: Language Model	0.007	0.2%
Total	2.708	100.0%

Table 4: Runtime performance of our approach.

candidate are added, the candidate is discarded, the highest priority candidate is drawn from the queue, and the process begins anew.

Letters outside of the candidate set are assigned a weight of 10^{-6}, which is significantly smaller than the weights of any observed keypresses. This accounts for cases where the actual key is not contained within the predicted set. We break ties randomly, i.e., we report the average-case estimates when multiple candidates have the same value for the product of the confidence values.

Table 3 shows our password guessing results. Of the 15 passwords used, 6 were reconstructed in 3 or fewer guesses, 12 in less than 100 guesses, and all 15 in less than 6000 guesses. Random guessing of passwords would result in almost 6 million guesses for 5-letter passwords and almost 155 million guesses for 6-letter passwords, in expectation. Accordingly, our approach results in a speedup of four orders of magnitude. That said, we remind the reader that the passwords used were those chosen by our test subjects. Unfortunately, our test subjects chose passwords which were between 5 and 8 characters long and consisted of only lower-case letters. However, since the mobile devices in which we are interested simply overlay the lower-case keyboard with an upper-case, numeric, or symbolic keyboard as necessary, extending our analysis to cover these cases would not be difficult.

5.1.2 Runtime Performance

The overall performance is 0.37 frames per second, the composition of which is shown in Table 4. The current version of our approach is mainly implemented in MATLAB; only the tracking scheme in Stage ❶ is optimized with C++ and the use of a GPU. As such, re-implementation of the remainder of our approach in C++ and/or on the GPU would likely result in a significant speedup.

6. MITIGATIONS

Our attack invalidates a significant portion of prior defences (e.g., like eliminating key pop-up events (Raguram et al.)). Perhaps the most natural mitigation is to apply a privacy screen/film to the device. This will limit the amount of light emitted, making reconstruction of reflected images (with inherently reduced brightness) extremely difficult. Moreover, it will significantly limit the possibility of direct sight attacks by narrowing the angle of screen observation. However, many materials are diffuse reflectors (reflection in all directions), which may lift the restriction on viewing angle imposed by privacy films by allowing an adversary to exploit the reflection rather than a direct view of the screen.

More esoteric mitigations that are focused on hindering the recovery of sensitive input (e.g., passwords) call for the application of gaze-based passwords [22, 44], shoulder-surfing resistant graphical password schemes (e.g., [17, 36]), and randomized keyboards [39, 47]. Gaze-based password systems, for example, take into consideration the focus of the user's eyes on the screen as the source of input, greatly reducing the ability of an adversary without an unobstructed view of the user's eyes and orientation with respect to the screen [22, 44]. Likewise, randomized keyboards permute onscreen keyboard layouts as the user enters her password [39, 47], effectively thwarting shoulder-surfing attacks during such times.

Clearly, these proposals would hamper the ability to reconstruct entered passwords using approaches similar to ours; in particular, attacking either the graphical password schemes or the randomized keyboard schemes would require the ability to discern detail on the screen of the device. That said, all of these proposals are limited to password entry, and as such, only offer a partial solution to this challenge of limiting recovery of typed input exposed via compromising reflections. Moreover, the real-world usability of these proposals remains unclear.

7. CONCLUSIONS

In this paper, we provide a technique for accurately reconstructing the text typed on a mobile device, with a broader range of credible threats that underscore the realism of the attack. Specifically, we show that it is possible to perform such attacks in a number of challenging scenarios: on devices with any type of virtual or physical keyboard, without direct line-of-sight, and at distances farther away from the victim than previously thought possible. Our attack can even directly use reflections on the eye-ball, which was not possible before due to the noise and physical boundaries of the optics in prior work. Our empirical analysis, which covers a number of scenarios (including direct line-of-sight, single reflections, and repeated reflections) as well as a range of distances (3 m - 50 m), demonstrates the success of our approach and underscores the significance of this privacy threat.

8. ACKNOWLEDGEMENTS

We are thankful to the anonymous reviewers for their insightful comments. This work is supported by a grant from the National Science Foundation, under award number 1148895.

References

[1] D. Asonov and R. Agrawal. Keyboard acoustic emanations. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2004.

[2] M. Backes, M. Durmuth, and D. Unruh. Compromising reflections-or-how to read LCD monitors around the corner. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2008.

[3] M. Backes, T. Chen, M. Dürmuth, H. Lensch, and M. Welk. Tempest in a teapot: Compromising reflections revisited. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2009.

[4] S. Baker and I. Matthews. Lucas-Kanade 20 years on. *International Journal of Computer Vision*, 56(3):221–255, 2004.

[5] D. Balzarotti, M. Cova, and G. Vigna. ClearShot: Eavesdropping on keyboard input from video. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2008.

[6] L. Cai and H. Chen. Touchlogger: inferring keystrokes on touch screen from smartphone motion. In *USENIX Workshop on Hot Topics in Security (HotSec)*, 2011.

[7] L. Cai and H. Chen. On the practicality of motion based keystroke inference attack. *Trust and Trustworthy Computing*, pages 273–290, 2012.

[8] A. Chaudhary, J. Raheja, and S. Raheja. A vision based geometrical method to find fingers positions in real time hand gesture recognition. *Journal of Software*, 7(4): 861–869, 2012.

[9] T. Chen and M.-Y. Kan. Creating a live, public short message service corpus: The NUS SMS corpus. *Language Resources and Evaluation*, 2011.

[10] R. Collins and R. Weiss. Vanishing point calculation as a statistical inference on the unit sphere. In *Proceedings of the Third International Conference on Computer Vision*, 1990.

[11] R. O. Duda and P. E. Hart. Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[12] F. Elibol, U. Sarac, and I. Erer. Realistic eavesdropping attacks on computer displays with low-cost and mobile receiver system. In *Proceedings of the 20th European Signal Processing Conference*, 2012.

[13] M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[14] W. N. Francis and H. Kucera. Brown corpus manual. Technical report, Dept. of Linguistics, Brown University, 1979.

[15] H. Grabner, M. Grabner, and H. Bischof. Real-time tracking via on-line boosting. In *Proceedings of the British Machine Vision Conference*, volume 1, pages 47–56, 2006.

[16] H. J. Highland. Electromagnetic radiation revisited. *Computer Security*, 5:85–93, June 1986.

[17] B. Hoanca and K. J. Mock. Password entry scheme resistant to eavesdropping. In *Security and Management*, 2008.

[18] X. Iturbe, A. Altuna, A. Ruiz de Olano, and I. Martinez. VHDL described finger tracking system for real-time human-machine interaction. In *International Conference on Signals and Electronic Systems*, 2008.

[19] L. Jin, D. Yang, L. Zhen, and J. Huang. A novel vision-based finger-writing character recognition system. *Journal of Circuits, Systems, and Computers*, 16(03):421–436, 2007.

[20] C. Kerdvibulvech and H. Saito. Vision-based detection of guitar players' fingertips without markers. In *Computer Graphics, Imaging and Visualisation*, 2007.

[21] M. Kuhn and C. Kuhn. Compromising emanations: eavesdropping risks of computer displays. Technical report, University of Cambridge, 2003.

[22] M. Kumar, T. Garfinkel, D. Boneh, and T. Winograd. Reducing shoulder-surfing by using gaze-based password entry. In *Symposium on Usable Privacy and Security*, 2007.

[23] A. Lavie. Evaluating the output of machine translation systems. *AMTA Tutorial*, 2010.

[24] A. Lavie and M. J. Denkowski. The METEOR metric for automatic evaluation of machine translation. *Machine Translation*, 23(2-3):105–115, 2009.

[25] B. Lee and J. Chun. Manipulation of virtual objects in marker-less AR system by fingertip tracking and hand gesture recognition. In *Proceedings of the 2nd International Conference on Interaction Sciences*, 2009.

[26] T. Lee and T. Hollerer. Handy AR: Markerless inspection of augmented reality objects using fingertip tracking. In *IEEE International Symposium on Wearable Computers*, 2007.

[27] E. Lutton, H. Maitre, and J. Lopez-Krahe. Contribution to the determination of vanishing points using Hough transform. *Transactions on Pattern Analysis and Machine Intelligence*, 16(4):430–438, 1994.

[28] M. Magee and J. Aggarwal. Determining vanishing points from perspective images. *Computer Vision, Graphics, and Image Processing*, 26(2):256–267, 1984.

[29] F. Maggi, A. Volpatto, S. Gasparini, G. Boracchi, and S. Zanero. A fast eavesdropping attack against touchscreens. In *Information Assurance and Security (IAS)*. IEEE, 2011.

[30] J. Nakamura. *Image sensors and signal processing for digital still cameras*. CRC, 2005.

[31] D. Nguyen, T. Pham, and J. Jeon. Fingertip detection with morphology and geometric calculation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009.

[32] K. Oka, Y. Sato, and H. Koike. Real-time fingertip tracking and gesture recognition. *Computer Graphics and Applications*, 22(6):64–71, 2002.

[33] E. Owusu, J. Han, S. Das, A. Perrig, and J. Zhang. Accessory: password inference using accelerometers on smartphones. In *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*. ACM, 2012.

[34] R. Raguram, A. White, D. Goswami, F. Monrose, and J. Frahm. iSpy: automatic reconstruction of typed input from compromising reflections. In *Proceedings of the ACM Conference on Computer and Communications Security*, 2011.

[35] R. Raguram, A. M. White, Y. Xu, J.-M. Frahm, P. Georgel, and F. Monrose. On the privacy risks of virtual keyboards: automatic reconstruction of typed input from compromising reflections. *IEEE Transactions on Dependable and Secure Computing*, 2013.

[36] L. Sobrado and J.-C. Birget. Graphical passwords. *The Rutgers Scholar*, 4, 2002.

[37] E. Stelzer. Contrast, resolution, pixelation, dynamic range and signal-to-noise ratio: fundamental limits to resolution in fluorescence light microscopy. *Journal of Microscopy*, 189 (1):15–24, 1998.

[38] R. Szeliski. Image alignment and stitching: A tutorial. *Foundations and Trends in Computer Graphics and Vision*, 2006.

[39] D. S. Tan, P. Keyani, and M. Czerwinski. Spy-resistant keyboard: More secure password entry on public touch screen displays. In *Proceedings of the 17th Australia Conference on Computer-Human Interaction*, 2005.

[40] N. Ukita and M. Kidode. Wearable virtual tablet: fingertip drawing on a portable plane-object using an active-infrared camera. In *Proceedings of the International Conference on Intelligent User Interfaces*. ACM, 2004.

[41] W. van Eck. Electromagnetic radiation from video display units: an eavesdropping risk? *Computer Security*, 4: 269–286, December 1985.

[42] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. In *Proceedings of the International Conference on Multimedia*, 2010.

[43] M. Vuagnoux and S. Pasini. Compromising electromagnetic emanations of wired and wireless keyboards. In *Proceedings of the* 18^{th} *USENIX Security Symposium*, 2009.

[44] J. Weaver, K. J. Mock, and B. Hoanca. Gaze-based password authentication through automatic clustering of gaze points. In *IEEE International Conference on Systems, Man and Cybernetics*, 2011.

[45] J. Weickert. *Anisotropic diffusion in image processing*, volume 1. Teubner Stuttgart, 1998.

[46] D. Yang, L. Jin, and J. Yin. An effective robust fingertip detection method for finger writing character recognition system. In *Proceedings of the International Conference on Machine Learning and Cybernetics*, 2005.

[47] Y. Zhang, P. Xia, J. Luo, Z. Ling, B. Liu, and X. Fu. Fingerprint attack against touch-enabled devices. In *Security and Privacy in Smartphones and Mobile Devices*, SPSM '12, 2012.
[48] Z. Zhang. Vision-based interaction with fingers and papers. In *Proceedings International Symposium on the CREST Digital Archiving Project*, 2003.
[49] L. Zhuang, F. Zhou, and J. Tygar. Keyboard acoustic emanations revisited. In *Proceedings of the 12th ACM Conference on Computer and Communications Security*, 2005.

APPENDIX

A. ANISOTROPIC DIFFUSION

Our alignment to the reference layout (Section 4) is based on image edges. Hence we opt for anisotropic diffusion, which is a noise suppression algorithm that maintains edges in the image. Our anisotropic diffusion is in the spirit of the method proposed by Weickert [45], which was designed to preserve all detected edges in the image. In our work however, we only preserve images edges that are along the dominant directions of the boundaries of the mobile device. This not only reduces noise but also effectively suppresses spurious lines, as for example, caused by reflections on the screen. The diffusion process is described by:

$$u' = \begin{cases} \partial_t u = div((1-c(\phi)(1-D(J_\rho(\nabla u_\sigma))))\nabla u) \\ c(\theta) = K_\Sigma(\phi - Dir_1) + K_\Sigma(\phi - Dir_2) \\ D(J_\rho(\nabla u_\sigma)) = U^T \begin{pmatrix} 1/(1+\lambda_1) & 0 \\ 0 & 1 \end{pmatrix} U \end{cases} \quad (4)$$

where u is the pixel value of the original image, u' is the new pixel value, and ϕ is the angle of the eigenvector of the biggest eigenvalue λ_1 of J_ρ (the Jacobian matrix of the image at pixel u). The matrix U is the unitary matrix consisting of the eigenvectors of J_ρ. K_Σ is a Gaussian kernel with parameter Σ, and Dir_1, Dir_2 are the pre-calculated dominant directions. The resulting filter output has the following properties:

1) If the edge is strongly deviating from the main direction (i.e. has a large angle to the dominant direction), it will be blurred since Equation (4) converges to $\partial_t u = div(\nabla u)$.

2) Edges along the dominant directions will be enhanced along the edge since $D(J_\rho(\nabla u_\sigma))$ becomes very small. This means the edge will not be blurred and will maintain its high contrast.

3) In addition, the areas alongside the edges in the dominant directions will be smoothed in the direction parallel to the edge to suppress the noise since in this direction we also have $\partial_t u = div(\nabla u)$.

In summary this will ensure an effective noise suppression while maintaining edges in the dominant directions, which are then used to align the device image to the reference layout.

B. VANISHING POINTS

Vanishing points are points in the 2D projected image where lines that are parallel converge. There is a large body of work on estimating vanishing points. Most of the methods are based on premarked points and mainly aim to solve problems when there is significant skew in the directions of the lines, such as [28]. However, these methods fail in our application because the vanishing points are so far away that the data matrix becomes singular. Alternative methods are based on transformations such as the Hough transform [27] and Sphere representation [10].

Due to its ability to detect multiple vanishing points at once we chose to use a Hough transform to calculate the vanishing points. The Hough transform is a voting scheme in the parameter space where every pixel along a line segment votes for all compatible parameterizations of its line. The detected lines are then chosen as the local maxima in the voting space. We use the angular representation $xcos\theta + ysin\theta = \rho$ of line segments for the voting, which avoids the singularity of vertical lines in the slope and bias parameterization of lines. The Hough transformation then provides the θ_i, ρ_i of all lines l_i in the image. These lines are then used to compute the vanishing point $v = (x, y)$ through solving:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \rho \frac{d\theta}{d\rho} \cos\theta - \sin\theta \\ \rho \frac{d\rho}{d\theta} \sin\theta + \cos\theta \end{pmatrix} * \frac{1}{d\theta/d\rho} \quad (5)$$

for each of the lines. To combine the information from all lines along each of the major directions we use regression to compute $d\theta/d\rho$ and θ. This is then used in solving for the vanishing points (x, y) with Equation (5), which results in the horizontal vanishing point v_x and the vertical vanishing point v_y. Assuming v_x, v_y are normalized to satisfy $\|v_x\| = \|v_y\| = 1$ the camera rotation matrix R is given by

$$R = \begin{pmatrix} v_x \\ v_y \\ v_x \times v_y \end{pmatrix}$$

Applying the inverse rotation ensures the device screen plane and the coordinate systems x-y-plane are parallel, effectively removing any rotation-related distortion from the image.

C. TRANSLATION ALIGNMENT

The transformation of a given image to the template can be viewed as rotation of the camera followed by translation and zoom. Our method uses the Hough transformation to extract the camera's rotation, translation, and scale as explained in Section 4. Here, we introduce how to compute the resulting transformation given estimated parameters.

The vanishing points reveal the rotation information. For instance, all the horizontal lines on the device correspond to a vanishing point (x, y) on the image plane, which is calculated as described above. The direction of these lines can be calculated as the direction from the center of the camera to the pixel (x, y). For a pinhole camera, this 3D direction *Direction* can be represented as: $(x - c_x, y - c_y, f)^T$ with (c_x, c_y) being the coordinate of the center of the image and f representing the focal length of the camera normalized by the physical size of the pixels. These are intrinsic parameters modeling the imaging parameters of the camera sensor and lens.

With the vanishing points, we acquire the two dominant directions of the edges of the device as $Direction_h = (x_h, y_h, z_h)$ and $Direction_v = (x_v, y_v, z_v)$. The edges of the device can be transformed to become horizontal and vertical with following equation:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \frac{\begin{pmatrix} \frac{x-c_x}{f} x_h + \frac{y-c_y}{f} y_h + z_h \\ \frac{x-c_x}{f} x_v + \frac{y-c_y}{f} y_v + z_v \end{pmatrix} * f}{\det \begin{pmatrix} x_h & y_h & z_h \\ x_v & y_v & z_v \\ \frac{x-c_x}{f} & \frac{y-c_y}{f} & 1 \end{pmatrix}} + \begin{pmatrix} c_x \\ c_y \end{pmatrix}$$

Here, point (x, y) in the image is transformed to (x', y'). This equation virtually rotates the camera in the 3D space so that the coordinate systems x-y-plane is parallel with the device's boundary. As discussed in Section 4, the translation parameters are calculated using a Hough transform using the weighted line voting. Considering the parameters acquired by the line matching (translations t_x, t_y and scaling s_x, s_y in x-direction and y-direction respectively), the transformation performing translation and scale alignment is given as: $(x'', y'')^T = (s_x x' + t_x, s_y y' + t_y)^T$. This maps the previous transformation result (x', y') to the point (x'', y''). In this way, we successfully transform the device screen's image into the template's coordinate system.