On the Privacy Risks of Virtual Keyboards: Automatic Reconstruction of Typed Input from Compromising Reflections

Rahul Raguram, Andrew M. White, Yi Xu, Jan-Michael Frahm, Pierre Georgel, and Fabian Monrose

Abstract—We investigate the implications of the ubiquity of personal mobile devices and reveal new techniques for compromising the privacy of users typing on virtual keyboards. Specifically, we show that so-called compromising reflections (in, for example, a victim's sunglasses) of a device's screen are sufficient to enable *automated* reconstruction, from video, of text typed on a virtual keyboard. Through the use of advanced computer vision and machine learning techniques, we are able to operate under extremely realistic threat models, in real-world operating conditions, which are far beyond the range of more traditional OCR-based attacks. In particular, our system does not require expensive and bulky telescopic lenses: rather, we make use of off-the-shelf, handheld video cameras. In addition, we make no limiting assumptions about the motion of the phone or of the camera, nor the typing style of the user, and are able to reconstruct accurate transcripts of recorded input, even when using footage captured in challenging environments (e.g., on a moving bus). To further underscore the extent of this threat, our system is able to achieve accurate results even at very large distances—up to 61 m for direct surveillance, and 12 m for sunglass reflections. We believe these results highlight the importance of adjusting privacy expectations in response to emerging technologies.

Index Terms—Privacy, security, side-channel attack, human factors, compromising emanations, mobile devices

1 Introduction

The ability to obtain information without the owner's knowledge or consent is one which has been sought after throughout human history, and which has been used to great effect in arenas as diverse as war, politics, business, and personal relationships. Accordingly, methods and techniques for compromising—and protecting—communications and data storage have been studied extensively. However, the ubiquity of powerful personal computing devices has changed how we communicate and store information, providing new possibilities for the surreptitious observation of private messages and data. In particular, mobile phones have become omnipresent in today's society, and are used on a daily basis by millions of us to send text messages and e-mails, check bank balances, search the internet, and even make purchases. And while some of us may be concerned with—and take steps to prevent—shoulder-surfing and direct observation of the text we input into these devices (see Fig. 1), few of us take notice of the person facing us across the aisle on our morning bus ride nor consider what our sunglasses might reveal.

In this work, we show that automated reconstruction of text typed on a mobile device's virtual keyboard is possible via *compromising reflections*, e.g., those of the phone in the user's sunglasses. Such *compromising reflections* have been exploited in the past for reconstructing text displayed on a screen [2], [3] using expensive, high-powered telescopic lenses. Our approach operates on video recorded by inexpensive commodity cameras, such as those found in modern smartphones. The low resolution of these cameras makes visual analysis difficult, even for humans, and severely limits the possibility of directly identifying on-screen text. What makes this threat practical, however, is that most modern touchscreen smartphones make use of a *virtual keyboard*, where users tap keys on-screen. In the absence of tactile feedback, visual confirmation is typically provided to the user via a key *pop-out* effect, as illustrated in Fig. 1. Note that although the on-screen text is essentially unreadable, the pop-out event provides a strong visual cue to help identify the letter that was tapped. The approach we take in this paper exploits this effect to recover the text typed by the user, and offers much promise in two real-world threat models, namely: 1) *direct surveillance*, wherein we assume the adversary is able to direct the camera toward the screen of the mobile device (e.g., over the victim's shoulder), and 2) *indirect surveillance*, wherein the adversary takes advantage of indirect views of the virtual keyboard obtained, for example, via "compromising reflections" of the phone in the victim's sunglasses. In both cases, we assume only inexpensive, commodity video cameras, without any telescopic lenses or high-end equipment. In addition, we make no limiting assumptions about the capture setup, the motion of the phone or of the camera, nor the typing style of the user. Thus, the only input we assume is a video, captured either directly or indirectly, of a user typing on a virtual keyboard. We then apply techniques from computer vision to process the recorded video, identifying, for each frame, potential keys that were

• *The authors are with the Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, NC 27514. E-mail: {rraguram, amw, yix, jmf, georgel, fabian}@cs.unc.edu.*

Manuscript received 22 Oct. 2012; revised 12 Feb. 2013; accepted 6 Mar. 2013; published online 13 Mar. 2013.
For information on obtaining reprints of this article, please send e-mail to: tdsc@computer.org, and reference IEEECS Log Number TDSC-2012-10-0258. Digital Object Identifier no. 10.1109/TDSC.2013.16.

Fig. 1. Example threat scenarios that we investigated. Video was recorded in both indoor and outdoor environments, using various consumer video cameras. Top: shoulder surfing. Bottom: reflection surfing. Observe the key *pop-out* events in the inset images.

pressed. This visual detection, coupled with a language model, enables us to achieve surprisingly accurate retrieval results, even under challenging scenarios.

An earlier version of this work was originally presented in [1]. For the present paper, we demonstrate more exhaustive results on a wider range of threat scenarios. One of the questions we explore relates to understanding the range of scenarios under which our attack is practical; for instance, given a capture device (e.g., a pocket camera), from how far away can an attacker effectively eavesdrop on a victim? We investigate this issue by considering a variety of capture devices, ranging from compact pocket cameras to higher-end digital SLRs. In addition, while our original work focused mainly on recovering complete typed sentences, we now explore the issue of recovering passwords. In other words, we analyze the efficacy of our system without the language modeling steps, and benchmark its accuracy for this task. Finally, we also propose and investigate an additional attack that uses still images processed with optical character recognition (OCR) techniques. Our experiments show that while this type of attack is simpler, and viable under certain operating conditions, it is still far less robust than the video based approach, which operates over greater distances.

Our ability to reconstruct text typed on virtual keyboards from compromising reflections underscores the need to continually reevaluate our preconceptions of privacy—or the lack thereof—in modern society. Even cryptography and secure devices are of little use when, across the aisle, someone who appears to be reading e-mail on their phone is in fact surreptitiously recording every character we type.

2 Related Work

By now, it is well understood that electronic, electro-optical and electromechanical devices give off some form of unintentional electromagnetic signals that can inadvertently leak sensitive information. The risks from these so-called "*compromising emanations*" were noted over half a century ago, and led to the introduction of emission-security tests standards to control leakage from digital electronics [4]. Although the nature of these emissions has changed with technology, side-channel attacks continue to surface [5], [6], [7], [8], [9], [10].

More recently, both visual emanations (e.g., from reflections on curved surfaces of close-by objects such as tea pots) and acoustic emanations (e.g., from key presses on a keyboard or from the sounds made by dot-matrix printers) [11], [12], [13] have been used to undermine the confidentiality of information displayed or entered into commodity devices. More closely related is the work of Backes et al. [2], [3] on "*compromising reflections*" that presents eavesdropping techniques for exploiting optical emanations using telescopic equipment. There, the authors show that an adversary is able to successfully spy from as far as 30 meters away and, in certain cases, can even read large text reflected in the eyeball of the victim. In this work, we focus on a related but different problem, namely, exploring the feasibility of automatic generation of transcripts from low resolution, indirect footage captured using inexpensive, and ubiquitous, consumer electronics.

The work most closely related to ours is that of Maggi et al. [14], who consider similar automated attacks on touchscreen devices. However, their work considers only direct surveillance and lacks any error correction, such as performed in the edit distance and language modeling portions of our approach. In addition, Maggi et al. conclude that reconstruction when both the device and camera are "jiggled" is infeasible with their system, whereas our experiments on a moving bus demonstrate the effectiveness of our approach even under such adverse conditions.

Also germane to this paper is the work of Balzarotti et al. [15] that explores the idea of automatically reproducing text from surveillance video—albeit from a camera mounted directly above a terminal—that captures a user's typing as she inputs data at the keyboard. Similar to Balzarotti et al., we apply the noisy channel model to help recover sequences of words from streams of frames with guessed labels. However, the error model employed by Balzarotti et al. only accounts for the deletion of identified characters and the substitution of one character for another. In contrast, our model allows for insertions, deletions, and substitutions, with substitutions weighted according to the distance between the two characters on the keyboard. Moreover, unlike Balzarotti et al., our frame parsing model handles spacing, allowing for the insertion and removal of spaces. An additional challenge in our setting is the need to overcome significant instability in the captured footage, as well as operate at a far lower resolution. The instability comes from the fact that in our case, both the phone and the camera are free to move and can be positioned arbitrarily with respect to each other.

A more distantly related problem is that of extracting captions in broadcast news to provide search metadata for digital archives. In these works, the low resolution of characters within the video makes the problem of segmenting characters quite challenging, so much so that video OCR typically does not perform well without significant text enhancement [16].

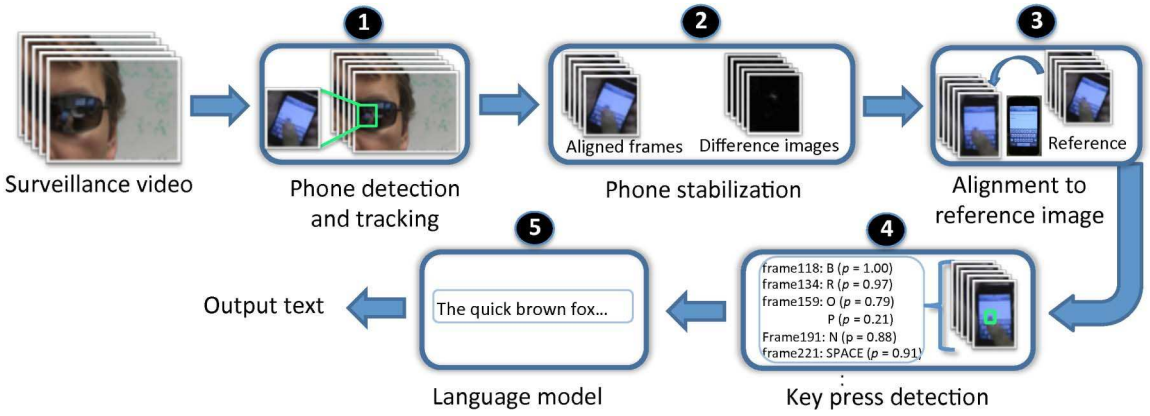


Fig. 2. Overview of our approach for typed input reconstruction from video.

3 Our Approach

Our approach consists of a number of stages (refer Fig. 2), each a difficult problem requiring the application of advanced computer vision and machine learning techniques. At a high level, the approach we take can be summarized via the following steps:

- *Stage* ①. We first detect and track the phone across the frames of a video sequence.
- *Stage* ②. We extract distinctive feature points from the tracked phone region in each frame, which are used to compute stabilizing image transformations that compensate for camera and phone motion.
- *Stage* ③. The stabilized video frames are also aligned to a reference image of the phone (obtained, for instance, from the user manual of the device of interest).
- *Stage* ④. Pop-out events for each key are now localized to specific regions of the keyboard, and we train classifiers to detect each key pop-out.
- *Stage* ⑤. To account for missed and spurious detections, we use a language model to refine the output of the computer vision modules.

Note that we apply the above steps to both threat models, i.e., direct surveillance and sunglass reflections. In the latter case, the images are simply flipped to account for lateral inversion, and then processed as above. In the following sections, we discuss each component in more detail.

3.1 Phone Detection and Tracking (Stage ①)

Given a surveillance video, one of the most basic challenges we face is in determining the location of the phone in the video. It is often the case that the phone occupies only a small spatial region of the image, with the remaining portion being unrelated background clutter (e.g., the phone in Fig. 3 only occupies 1.8 percent of the total image area). Indeed, the visual features on the background are invariably "distracting" for subsequent stages of our approach, because they vastly outnumber the visual features on the phone itself. Determining the location of the phone in the video, thus, enables us to focus specifically on the object of interest, eliminating all irrelevant background information.

The domains of *object detection* and *object tracking* have received widespread attention in computer vision [17], [18], [19], [20], [21]. In certain applications, such as frontal-view face detection, modern techniques are capable of providing very accurate results. In general, however, object detection and tracking are still challenging problems, in part due to the tremendous variability in the appearance of objects when captured in arbitrary configurations, i.e., from different angles, distances, and under different lighting conditions.

In this paper, we formulate the tracking problem as one of binary classification [20], [21], [22]. The intuition is to train binary classifiers to distinguish the appearance of the object being tracked from that of the background. This training is either performed *offline* (using a dedicated training phase prior to the tracking algorithm), or *online* (where the appearance of the object is learned *during* the tracking process). In the former case, tracking is typically very fast, since the classifiers have been pretrained beforehand. The latter, while slower, is capable of adapting on-the-fly to changes in appearance. Since the appearance of the phone can vary considerably, we elect to perform online training, learning the appearance of the phone during tracking.

We base our phone tracker on the techniques proposed in [19], [20], which describe an online AdaBoost [23] feature selection algorithm for tracking. At a high level, *boosting* is a classification scheme that works by combining a number of *weak learners* (e.g., a threshold on a feature value) into a more accurate *ensemble* classifier. A weak learner may be

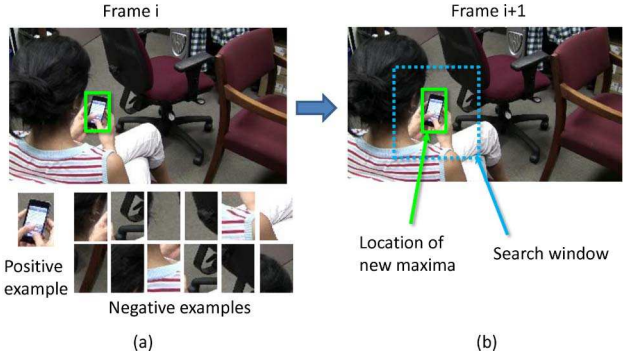


Fig. 3. Phone tracking. (a) User-selected bounding box highlighting our positive example for training. (b) In the next frame, the classifier is evaluated within a search window, and the position of the maximum response defines the phone's new location.

thought of as a "rule of thumb" that only has to perform slightly better than chance—for example, in a binary classification problem, the error rate must be less than 50 percent. The intuition is that a combination of these "weak" rules will often be more accurate than any individual rule. Given a set of training images, where each image is labeled as positive (containing the phone) or negative (not containing the phone), we obtain a training set $\chi^L = \{\langle \mathbf{x}_1, y_1 \rangle, \ldots, \langle \mathbf{x}_{|\chi^L|}, y_{|\chi^L|} \rangle\}$, where $\mathbf{x}_i$ is an m-dimensional feature representing the ith training image, and $y_i \in \{+1, -1\}$ is the corresponding label. As suggested by Grabner et al. [19], we use three types of features, concatenated to form the vector $\mathbf{x}_i$: Haar features [18], orientation histograms [24], and local binary patterns [25].

Initially, each training example is given a uniform weight $p(\mathbf{x}_i) = 1/|\chi^L|$. During the first round of training, we select a weak learner that has the lowest weighted classification error, given χ^L and $p(\mathbf{x})$, and add it to our ensemble. Following this, the weights $p(\mathbf{x}_i)$ of the misclassified training examples are increased, while the weights of the correctly classified samples are decreased. At each subsequent training iteration, we select a weak learner h_i that does well on the training examples that were hard for the previous weak learners. The final ensemble classifier is computed as a linear combination of the selected weak learners, with the weight of each learner being proportional to its accuracy.

The online boosting variant assumes that one training example is available (for instance, by drawing a bounding box in the first video frame). This image region becomes the positive training sample, and negative examples are extracted from the surrounding background regions. Given this data, multiple training iterations of the online boosting algorithm are performed as above. In the next frame, the ensemble classifier is evaluated at a number of possible image locations (e.g., in a search window surrounding the object's position in the first frame), with each location being assigned a confidence value. The target window is then shifted to the new location of the maxima, and the classifier is updated using new training examples so as to become discriminative to variable object/background appearance. The operation of the tracker is illustrated in Fig. 3. The output of this module is the phone's location in each frame, allowing all further processing steps to focus on the phone.

3.2 Phone Stabilization (Stage ②)

Given the location of the phone in each frame, the next step is to compensate for the effects of phone and camera motion. As discussed earlier, we do not impose any constraints on the motion of either the camera or the user. While this enables us to operate in a wide range of real-world threat scenarios, it also results in a tremendous degree of variation in the appearance of the phone within each frame. Explicitly compensating for this motion would allow us to effectively reduce one dimension of variability, resulting in a more "stable" set of images to work with.

Before presenting the details of our stabilization algorithm, we introduce the notion of a *homography* [26]. In computer vision parlance, a homography is a 2D projective transformation that relates two images of the same planar surface (in our setting, the phone represents a (mostly) rigid, planar object). The images that we capture—specifically, the portions of the image that contain the phone—are, thus, related to each other via a 2D homography. Note that in the case of reflections, the image of the phone can be distorted due to the curvature of the sunglasses. We do not explicitly model this distortion in the current system, but rather assume that the sunglasses are approximately locally planar. Since the phone occupies only a small area of the sunglasses, this approximation proves to be sufficient.

The 2D homography has a number of important practical applications, one of which is image stabilization. In particular, if we were given access to the homography $\mathbf{H}$ between a pair of neighboring video frames, then we could *warp* them into alignment, thus removing the effects of phone and camera motion. In short, the basic idea is to compute pairwise homographies between the video frames, chaining them together to align all phones in the frames to a common reference frame. The problem now reduces to that of automatically determining the transformation $\mathbf{H}$, given two neighboring images I_t and I_{t+1}.

The approach we take involves two key steps. In our *feature extraction and matching* step, we extract stable, repeatable, and distinctive *feature points* in the two images, with the intuition being that we would like to identify matching points in the captured images that correspond to the same 3D point on the phone. For this, we use the Scale Invariant Feature Transform, or SIFT [27]. Each SIFT feature consists of a 2D image location, scale, orientation vector, and a 128-dimensional *feature descriptor* that represents a histogram of gradient orientations centered around the extracted feature. The main point is that a pair of features in two images that correspond to the same point in 3D space will have similar SIFT descriptors. The popularity of SIFT stems from its ability to tolerate a wide range of scale and illumination changes, as well some degree of viewpoint variation. For this task, we use a fast in-house GPU implementation of SIFT,[1] running at $\approx$12 frames per second on a standard graphics card.

For our *robust homography estimation* step, we compute the homography $\mathbf{H}$ from N "true" feature matches between the two images using the normalized Direct Linear Transformation (DLT) algorithm [26]. One limitation here, however, is that doing so requires a minimum of $N = 4$ correct feature matches between the two images. Additionally, the DLT algorithm is sensitive to outliers, or mismatched features. To combat this problem, we turn to the field of *robust statistics* [28], where the problem of estimating quantities from data that have been corrupted by noise and outliers has been well studied. Perhaps the most popular of these *robust estimators* is Random Sample Consensus (RANSAC) [29], which is a randomized, data-driven approach that is capable of tolerating a high level of data contamination. In our approach, we apply a fast, real-time variant of RANSAC, called Adaptive Real-Time Random Sample Consensus (ARRSAC) [30], which simultaneously estimates the 2D homography, as well as returning the set of "true" feature correspondences. The resulting homography can then be used to align the phone images together, thus nullifying the effects of scene and camera motion.

1. Available online: http://cs.unc.edu/~ccwu/siftgpu.

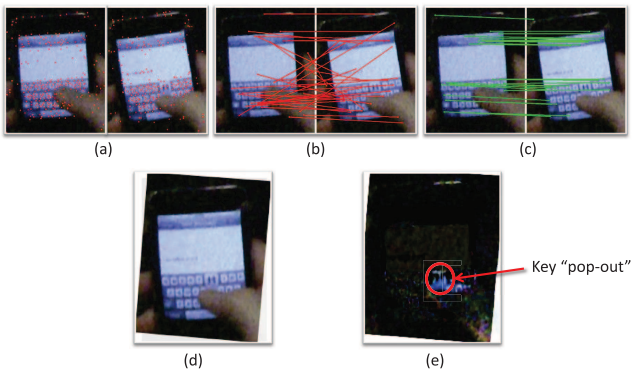


Fig. 4. Automatic phone stabilization. (a) Two video frames, with extracted SIFT features displayed on the images. (b) SIFT features are matched between the two images, to obtain a set of "tentative" point correspondences, shown in red. (c) ARRSAC-based robust estimation, which returns a homography, in addition to a set of "true" correspondences, shown in green. (d) The two video frames are aligned using the computed homography. (e) Pixel-wise difference following image alignment. Note that the main area of difference is in the vicinity of the key pop-out.

The two steps outlined above are illustrated in Fig. 4. Figs. 4a, 4b, and 4c denote the process of SIFT feature extraction, matching, and robust homography estimation, respectively. Notice that the incorrect feature matches present in Fig. 4b are "cleaned up" by ARRSAC, which selects the set of true correspondences (shown in Fig. 4c) out of the potential correspondences (Fig. 4b). These true correspondences are then used to estimate the homography between the two frames. Fig. 4d shows the two frames aligned with respect to each other, and Fig. 4e represents the pixel-wise difference between the images after alignment. In the difference image, dark pixels represent areas of low image difference, and lighter pixels represent areas of high difference. Note that the difference image consists mainly of dark pixels, which is an indication that the homography-based alignment has accurately aligned the images to each other.

3.3 Alignment to Reference Image (Stage ③)

In the previous section, we showed how one could compensate for the effects of scene and camera motion by aligning the video frames using a robust homography estimation procedure. While this results in a stabilized video, one other aspect of appearance variation that remains unaccounted for is the relative positioning between the surveillance camera and the user. Note that we do not assume that the camera has a clean, frontal view of the screen of the device; rather, the surveillance camera can be oriented arbitrarily with respect to the user. We now reduce the difficulty of our problem further by aligning the stabilized video to a *reference phone image*. This image can be obtained easily, in a number of ways, for example, by taking a single frontal-view photograph of the phone or using a photo from a reference manual.

Given a reference image, the process of aligning the stabilized video to this reference image can be carried out in much the same way as before; that is, by detecting features in the reference image and the video frames, matching them, and computing a robust homography estimate that

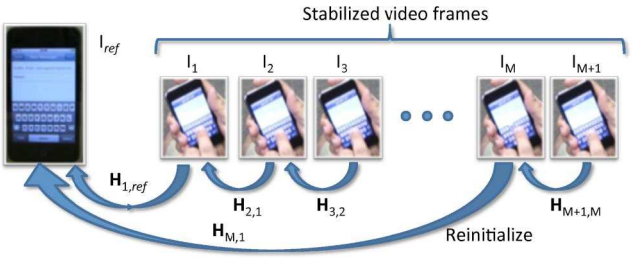


Fig. 5. Iterative procedure to align video frames to the reference image (I_{ref}). To prevent long-term drift, the frames are periodically reinitialized with respect to the reference image.

can then be used to warp all the video frames to the reference. In principle, we actually need to align only a *single* video frame to the reference image, because the frames of the video sequence have already been aligned to each other by pairwise homographies. More specifically, let $\mathbf{H}_{j+1,j}$ be the homography that transforms video frame I_{j+1} to frame I_j. Assuming that I_1 denotes the first frame, the transformation between I_{j+1} and I_1 can be computed by chaining together all previous pairwise transformations:

$$\mathbf{H}_{j+1,1} = \prod_{k=1}^{j} \mathbf{H}_{k+1,k}. \quad (1)$$

In theory, given a single transformation $\mathbf{H}_{1,ref}$ that aligns frame I_1 to the reference image I_{ref}, then by extension, one can align the entire video to the reference image. However, because the transformations are chained together, even a small error in the estimation of homography $\mathbf{H}_{j+1,j}$ propagates to all subsequent transformations. For a reasonably long video sequence, this invariably leads to "drift," where the alignment progressively deteriorates as more frames are aligned to the reference image.

To combat this effect, we instead perform a more careful alignment process, depicted in Fig. 5. We begin by aligning frame I_1 to the reference image I_{ref}, via a robust homography $\mathbf{H}_{1,ref}$, estimated using the techniques introduced in Section 3.2. We then align subsequent frames of video by chaining together pairwise homography transformations—the difference being that every M frames ($M = 50$ in our experiments), we reinitialize our transformation with respect to the reference image by recomputing the video-to-reference image homography. We use the newly estimated homography as the base transformation for the next window of M frames.[2] This process of interframe homography estimation is much more accurate, as well as far more efficient, than performing alignment to the reference image. The reason is that the change in appearance of the phone between two successive video frames is often very small, while the change in appearance between the phone image in a random video frame (captured under arbitrary pose and lighting conditions) and the reference image is much larger.

2. In particular, we perform a nonlinear minimization of the estimation error [26] with respect to the reference image. Reinitialization is a common trick often used in practice to prevent drift. In our case, periodic reinitialization also helps prevent catastrophic failure in the event of failed alignments during Stage ②.

3.4 Key Press Detection (Stage ④)

Thus far, we have focused primarily on accounting for sources of appearance variability: phone and camera motion and the spatial relationships between the user and the surveillance system. The net effect of the operations performed thus far is to convert an arbitrary, free-form video sequence into one that has been aligned to a stable, *known* reference frame. There is a significant advantage to doing so: By aligning the video to a known coordinate frame, we know precisely which regions to inspect to find key pop-out events. More specifically, once the video frames have been aligned to the reference image, we can isolate the key pop-out event of each key on the virtual keypad to a *specific spatial location* (derived, e.g., by overlaying 2D boxes on the reference image).

Although we have greatly simplified the problem, we are still faced with challenges. For one, because we are operating at a fairly low resolution, coupled with the fact that the appearance of the keys is often "blurred out," one cannot readily apply OCR techniques to recover the characters in the isolated frames. Moreover, in several cases the pop-out events are occluded. Yet another complication is that the 2D boxes constituting the keypad grid are overlapping—in other words, the key pop-out events for neighboring keys have a nonnegligible area of overlap. To address this, we do not make any final decisions at this stage; rather, for each frame, we inspect each key location independently and assign a score to each key, which may be interpreted as the probability of the key having been pressed in that frame. These scores, along with their key labels, are then used in the final stage.

3.4.1 Training a Key Press Classifier

The basic idea we use to identify key press events is to exploit the fact that we have a known, regular grid and to train a binary classifier for each key on the keypad. The classifier for each key focuses on a specific bounding box on the reference keypad, and aims to distinguish between a key pop-out event and the "background." We again make use of AdaBoost classifiers, introduced in Section 3.1, to perform this classification. In addition, because we have explicitly compensated for multiple sources of appearance variation, we can, at this stage, use an offline training procedure to have a classifier that is capable of rapid classification when processing each frame of video. Since we are operating on small sections of images, known as *patches*, and some illumination variation remains, we use dense SIFT descriptors as the features for each patch (i.e., a SIFT descriptor is extracted for each pixel in the patch, and concatenated to form a feature vector).

For each key on the keypad, we train a binary classifier, by providing positive and negative examples of key pop-out events. This data are obtained by running a representative collection of 10 training videos through Stages ①-③, subsequently labeling each aligned frame with the key pressed for the frame. For example, to detect the tapping of the letter "C," we extract a positive training patch from the 2D box corresponding to that letter, and negative patches for all other letters, at their respective locations. On average, we obtain 200 positive exemplars and 1,000 negative exemplars for each key. Each classifier is then trained offline using the acquired samples.

3.4.2 On Detecting Keyboard Layout Mode

While the above discussion focuses primarily on the alphabet keys, the same principles apply for special characters and numbers. On a smartphone, there is usually a special key that allows one to toggle between alphabet and numeric/special character mode. There are a couple of strategies one could adopt to detect keyboard toggle: 1) train a classifier that inspects the entire keyboard area to detect when the keyboard layout has been toggled, and then use the classifiers for the appropriate keys in each layout or 2) at each key pop-out location, run the classifiers for all keys that could potentially pop out at that location, and select the classifier that yields the highest score. In this work, we chose to pursue the latter option, and have used that approach to successfully detect numbers and special characters interspersed with alphabet characters.

3.4.3 Testing the Classifier

Given a test video, and a pool of trained key press classifiers, we run the test video through Stages ①-③. Then, for every frame of video, each classifier inspects its respective image patch and outputs a classification score (the probability of that key having been pressed in the frame). We reject detections that score less than 0.5. Note that each classifier is run independently, and so there could potentially be multiple keys that pass this threshold. For each frame, we store all potential key labels and scores. Once a key pops-out on the keypad, it typically stays in this state for a fixed amount of time (e.g., about 0.25 s on the `iPhone`); this fact can be used to parse the detected sequence of keys to identify character breaks.

The observant reader would have noticed by now that we have yet to discuss the issue of the "space" bar. On many popular smartphones we examined (e.g., the `iPhone` and `NexusOne`), there is no pop-out event for the space bar. However, it is still possible to obtain a reasonable estimate of the locations of the spaces in typed text, by performing some straightforward post-hoc analysis. Given a sequence of identified key press events, we determine the median time interval t between successive key presses. If we now reinspect our key press detections, we can label the frames lying between two widely separated[3] key presses as potential space events. Additionally, the visual classifier we use to determine the space key event inspects a larger region of the keyboard, with the intuition being that when users press the space, a large portion of the keyboard is visible. This is by no means foolproof: a few spaces may be missed, and false spaces may be inserted when the user pauses between keystrokes. However, coupled with an image-based classifier, this analysis still provides useful information.

3.5 Parsing and Language Modeling (Stage ⑤)

Once we have identified key labels for each frame in a video, along with potential character breaks and spaces, the issue of identifying typed words still remains. We can view this task in terms of the *noisy channel* problem, often encountered in speech recognition: given a sequence of observations (labeled frames), find the most likely sequence

3. A threshold of $1.5t$ appears to be effective in practice.

of intended words. This process is often referred to as *maximum-likelihood decoding* [31].

The noisy channel problem is often formulated in a Bayesian framework. Let $P(w \mid o)$ represent the probability of a word sequence w given an observation sequence o. The decoding problem is that of finding $\hat{w} = \operatorname{argmax}_w P(w \mid o)$. Using Bayes' rule, this can be reformulated as

$$\hat{w} = \operatorname*{argmax}_w \frac{P(o \mid w)P(w)}{P(o)} = \operatorname*{argmax}_w P(o \mid w)P(w),$$

where $P(o \mid w)$ represents the probability of observing a sequence o given that the sequence w was intended. The prior $P(w)$ represents the probability of observing word sequence w in the language of interest. The denominator can be safely omitted as it does not depend on w.

To solve the noisy channel decoding problem, speech recognition systems have long employed *cascades* of component models, where each model represents one conceptual stage in the task of transcribing speech. For instance, one such cascade might consist of three models: 1) an *acoustic model*, which transforms an acoustic signal into component sounds, 2) a *pronunciation model*, which converts sequences of sounds to individual words, and 3) a *language model*, which governs the combination of words into phrases and sentences. In this case, $P(w)$ represents the language model. The likelihood $P(o \mid w)$ can be further decomposed into submodels, such as the acoustic and pronunciation models, representing intermediate stages. Commonly, these submodels are assumed to be independent.

We draw on a large body of work on speech recognition cascades that has proven to be very useful in our context. However, in traditional speech recognition systems, the interaction between components often cannot be modeled explicitly, i.e., each step is performed independently. Pereira and Riley [32] proposed an elegant solution to this problem, representing each model and submodel as a *weighted finite-state transducer (WFST)*, thereby allowing for decoding to range over the entire cascade simultaneously. A finite-state transducer is a finite-state machine with both an input and an output tape and, thus, represents a mapping between sequences from two alphabets; applying weights to each arc then allows for scoring each path through the transducer. A finite-state *acceptor* can be viewed as the special case, where the input and output tapes are identical. Many of the traditional components of speech recognition systems can be represented as WFSTs, including n-gram language models, pronunciation dictionaries, and Hidden Markov models (HMMs). HMMs are themselves one method of approximating solutions to the noisy-channel decoding problem; however, the WFST framework can not only duplicate the functionality of an HMM-based approach but also streamlines the construction of a recognition cascade by allowing for different component models with a uniform representation. By representing system components uniformly as WFSTs, we take advantage of the fact that multiple finite-state automata can be combined in various ways: for example, a speech recognition cascade can be represented as the *composition* of individual transducers for each stage. The resulting cascade can then be composed with an acceptor representing an input sequence, which transforms the decoding problem into that of finding the shortest path through the WFST.

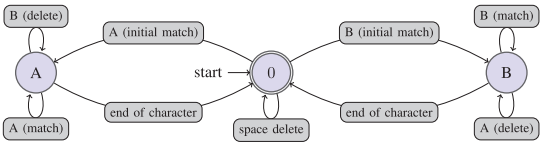


Fig. 6. Simplified frame parsing WFST, with only two characters, that maps a sequence of labeled frames to characters.

In what follows, we apply WFSTs to solve the noisy channel decoding problem in a manner similar to that of speech recognition cascades. That is, we utilize a language model and define a number of component models which, when combined, provide a probabilistic mapping from frame label sequences to word sequences. More specifically, we first apply a frame parsing model, $\mathcal{F}$, which maps sequences of frame labels to character strings. We then apply an edit distance model, $\mathcal{E}$, which maps each character string to a (weighted) set of similar character strings and helps account for errors made by the typist, the recognition algorithm, and the frame parser. Next, a dictionary model $\mathcal{D}$ is applied, discarding those paths resulting in invalid words. Finally, the language model $\mathcal{L}$ is applied, accounting for unlikely words and sequences of words in English.

Each component model is represented as a weighted finite-state machine, and the application of each is performed by composition. The resulting cascade WFST is then composed with the input sequence, represented as acceptor $\mathcal{I}$, resulting in a weighted transducer that maps from the input sequence to word sequences. We then search this transducer for the shortest path, which corresponds to the most likely sequence of words given the input sequence. We use the OpenFST library[4] to construct, combine, optimize, and search our model cascade.

The frame parsing WFST (Fig. 6) allows for character break and space insertion as well as frame, break, and space deletion and is parametrized by weights on each of these actions. Each path starts at the zero state, representing the beginning of the frame label sequence. Upon encountering a frame labeled with a character, a transition is made to a character-dependent state (A or B, in this example) and that character is output. Subsequent frames with the same label are ignored freely, while those with a different character label are dropped with a penalty. In a typical path, once a character break appears in the frame stream, the transition back to the start state is made, from which the string can end or a new character can begin. Thus, an input stream "AA | BB" would be output as "A | B" in a typical path. Other paths are possible, however, which insert or drop characters breaks and spaces with certain penalties; the same input would also generate, among other outputs, "A" and "A | A | B," albeit with a lower probability.

Our edit distance is based on the distance between keys on the keyboard and is intended to correct any misclassifications by the recognition algorithm. It can also automatically correct typing mistakes. The distance between two characters is straightforward. If both keys are in the same row, then the distance is the number of keys between them. If the keys are not in the same row, we calculate the distance as though they were in the same row (with rows

4. http://www.openfst.org.

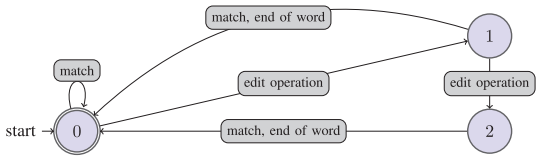


Fig. 7. Simplified edit distance WFST mapping sequences of characters to similar sequences.

aligned on the left-hand edge) and apply a multiplicative penalty for each row the two keys are apart. This distance is then normalized to be between zero and one; we take the additive inverse to obtain a probability estimate, which is weighted with a parameter. Similarly, the insertion and deletion costs are represented as probabilities and weighted with parameters, which allow for tuning the effects of the edit distance on the overall cascade. For efficiency, we limit the number of contiguous edits to 2.

A simplified view of the edit distance WFST appears in Fig. 7. For the edit distance WFST, the most likely path is a loop in the start state, in which case the output string is identical to the input string. However, an edit (substitution, insertion, or deletion) can be made, with a penalty, at any position in a word, resulting in a path which transitions to state 1; a second edit can then be made, transitioning to state 2. From either of these states, a match—or the end of a word—is required to return to the zero state, which limits to 2 the number of edits that can be made in a row.

The dictionary used is based on the medium-sized word list from the Spell Checker Oriented Word Lists (SCOWL),[5] from which we removed roman numerals and the more obscure abbreviations and proper nouns. Finally, the language model used is a unigram model, i.e., simple word frequencies, trained on the well-known Brown corpus [33].

4 Evaluation

4.1 Evaluating Output Quality

We now turn our attention to how we measure the quality of the reconstructions produced by our system. The problem we face here is similar to that found in both the automated speech recognition and machine translation (MT) communities. One common metric used in these domains is the *word error rate* (WER). The WER of a transcription is based on the normalized Levenshtein edit distance between the hypothesis and the reference, where the basic unit of comparison is the word. While WER has been used historically for many tasks, it has several failings [34] that make it ill-suited for our goals. Consider four hypotheses for the phrase "the art of war":

1. *the art war*,
2. *the art of painting*,
3. *the art of of war*, and
4. *art of war*.

Each of these hypotheses has exactly the same WER (1/4), even though they are quite different in quality, particularly as it relates to human understanding: notice that the last two sentences convey the appropriate meaning, while the first two do not.

5. http://wordlist.sourceforge.net.

For more appropriate metrics, we turn to the MT community, which has addressed many of the challenges associated with scoring the output of such systems [35], [36]. While humans are the target audience for MT systems (and thus the ultimate arbiters of output quality), evaluations using human judges pose several obstacles; for example, using *experts* can be prohibitively expensive and time consuming; conversely, hiring nonexperts leads to issues with reliability and inconsistency. Automated evaluation, on the other hand, allows system designers to quickly test new ideas while providing a consistent basis for comparing multiple approaches. Ideally, such automated evaluations would produce results similar to human experts, who typically assess the *adequacy*, or how well the appropriate meaning is conveyed, and *fluency* of a translation. Similarly, state-of-the-art automated MT evaluation techniques score a *hypothesis* (i.e., the MT) by comparing it with one or more *reference* (i.e., expert) translations. The performance of these automated techniques is judged according to how well the assigned scores correlate with those assigned by experts.

4.1.1 Scoring Our Inferences

Before proceeding further, we note that automated MT evaluation remains an area of active research, with entire conferences dedicated to the topic. Nevertheless, one widely adopted metric for producing scores at the segment level is the Metric for Evaluation of Translation with Explicit ORdering (METEOR) [37]. METEOR accounts for position-independent matching of words (i.e., to model adequacy) and differences in word order (i.e., to model fluency). More specifically, the METEOR metric is the combination of a weighted f-score and a *fragmentation penalty*. The f-score is defined as the harmonic mean of unigram *precision* p and *recall* r. In this context, precision is the ratio of the number of (nonunique) words that occur in both the reference and the hypothesis to the total number of (nonunique) words in the hypothesis. Recall is the ratio of the number of words present in both hypothesis and reference to the number of words in the reference.

Denkowski and Lavie [38] have extensively explored the space of tunable parameters, and have identified different sets of values that correlate well with human evaluations on different tasks; we use the Human-Targeted Edit Rate parameter set with synonym matching disabled. As a guideline for METEOR scores, Lavie [39] suggests that scores of 0.5 and higher indicate *understandable* hypotheses, while scores of 0.7 and higher indicate *good* or *fluent* hypotheses.

4.2 Results

4.2.1 Experiment #1: Full System Analysis

Recall that our primary goal is to explore the feasibility of exploiting compromising reflections using low-cost consumer devices, and to impose very few constraints on the capture environment. Toward this end, our first experiment uses capture devices ranging from low cost, handheld devices (Kodak PlayTouch and Sanyo VPC-CG20, costing $90 and $140, respectively) to mid-range consumer grade cameras (a Canon VIXIA HG21 Camcorder, retailing

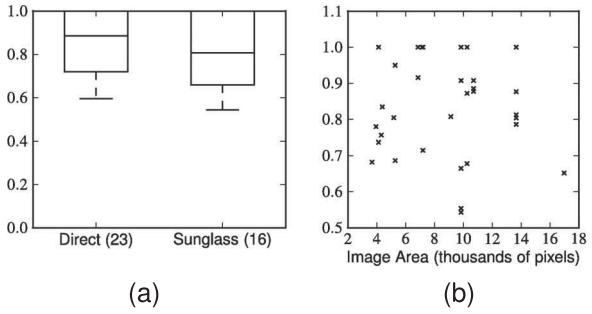


Fig. 8. (a) METEOR scores for direct surveillance and sunglass reflections, with the number of sentences listed in parentheses. (b) Plot of input area (i.e., image resolution) versus METEOR scores.

for about $1,000). Note that these devices have small form factors (see Fig. 9), thus allowing for unobtrusive capture in real-world settings. Our capture settings for this experiment cover both static and dynamic camera positioning, ranging from cameras mounted near the ceiling of an indoor office environment, to hand-held capture performed outdoors—for example, we recorded video footage at a bus stop as well as on a moving bus. In the indoor setup, the distance between the user and the camera was approximately 4.5 meters, while the outdoor capture was done at distances ranging from 1.2-2.2 meters (for instance, looking over a person's shoulder while sitting on a bus). At these distances, the pixel dimensions of the phone in the captured video ranged from about 49×75 to 114×149 ($width \times height$). While low resolution is in itself often problematic for computer vision systems, these data sets present numerous other challenges: unstable video, motion blur, reflections from other objects, and so on.[6] In total, we collected 18 videos (containing 39 sentences) from 10 different users typing on the `iPhone`. Our experiments covered a number of practical use-cases designed to elicit a variety of typing styles (and speeds), and includes scenarios, where subjects 1) typed short passages of text from *The Art of War* and David Kahn's *The Codebreakers*, 2) simply typed whatever came to mind, and 3) typed responses to text messages (e.g., "*What time shall we meet?*") sent to the phone. In each case, subjects were instructed to use the phone as they normally would. All subjects routinely use smartphones.

We evaluate our system at two levels: the sentence (or segment) level, and the system level. In the former case, we are interested in the ability of our system to reconstruct appropriate transcripts of individual sentences. This is particularly important as it allows an attacker to have confidence in the output of the system for individual sentences. The latter case provides a characterization of our system's performance as a whole.

Sentence-level accuracy. A boxplot of the METEOR scores for our reconstructions of the sentences typed in our collected videos is provided in Fig. 8a. Notice that in both the direct and indirect cases, more than 35 percent (8/23 and 6/16, respectively) of our hypotheses achieve *perfect* scores, and none score below the 0.5 threshold representing "understandable" translations. We provide a few examples of our hypothesized transcripts in Table 1, where we also list the input as actually typed by the user and the reference text used for scoring.

System-Level Analysis. Interestingly, while the basic unit for comparison is the segment or sentence, it is also instructive to consider evaluations at the level of entire corpora of documents, i.e., the system level. System-level analysis offers a different perspective and, in particular, smooths the dependency of the scoring on the length of the sentence. For instance, even a single mistake in a short sentence can lead to a relatively low METEOR score, as in Table 1 (Sentence 3). System-level analysis does not depend as strongly on the length of individual sentences and can therefore alleviate this issue to some extent. The formulas are the same as at the sentence-level, but instead, 1) the system-level precision is calculated as the ratio of the sum of the counts of matched words over all sentences to the total number of words over all hypothesis sentences, and 2) the fragmentation penalty is calculated based on the total number of contiguous subsequences and unigram matches over all sentences. To better judge how well the system-level scores generalize, we also provide confidence intervals based on *bootstrap resampling*, a common statistical technique for estimating the distribution of a quantity, which consists of sampling (with replacement) from the set used to derive a statistic and calculating a *bootstrap statistic* based on the new sample. This process is repeated many times, resulting in an empirical distribution over the statistic of interest. For direct surveillance, we achieve a system-level METEOR score of 0.89, with a bootstrapped 95 percent confidence interval of $[0.84, 0.93]$. In the indirect surveillance case, we achieve a lower, yet still respectable, system score of 0.77, with a bootstrapped 95 percent confidence interval of $[0.70, 0.86]$.

Impact of input resolution. To gain a deeper understanding of the influence of the various input resolutions (of the phone's screen) on our ability to reconstruct the typed text, we plot the area (in pixels) of each input, against the ultimate METEOR score (Fig. 8b). The figure shows no correlation, as evidenced by a correlation coefficient (Pearsons's r-value) of 0.07. This indicates that our system performs robustly over a wide range of input resolutions.

Operating range. Finally, one of our goals in this work is to gain an understanding of the range of scenarios under which this kind of attack is practical. For instance, should you be concerned about someone sitting across the aisle from you on a bus, equipped with a low-cost hand held camera? What about an attacker standing on the second floor of a building, recording people through a window? Stated differently: How sophisticated (or expensive) does a capture device need to be, to operate under realistic threat models in real-world operating conditions?

To answer these questions, we "stress-tested" our system on a range of devices: a pocket camera (Kodak PlayTouch), mini camcorder (Sanyo Xacti), point-and-shoot (Nikon Coolpix s9100), HD camcorder (Canon Vixia HG21), and a mid-range digital SLR with a zoom lens (Canon EOS 60D with a 100-400-mm lens) (see Fig. 9(bottom)). These devices were selected to cover a range of form-factors, prices, and levels of optical zoom. For each device, we

6. See http://cs.unc.edu/ispy for some examples.

Fig. 9. Operating ranges for five different capture devices. Top: Direct surveillance. Middle: Sunglass reflections. Bottom: Cameras used, along with their approximate price (as of June 2012). For each device, we determine the maximum distance at which we can capture video that would then be successfully processed by our system (where "successful" is defined by a METEOR score >0.5).

TABLE 1
Example Hypotheses from Our Reconstruction Process

	Sentence		Scenario	METEOR Score
1	Typed:	to be prepared beforehand for any contingency is the greatest of virtues	Sunglasses Canon camcorder 66x104	0.92
	Reference:	to be prepared beforehand for any contingency is the greatest of virtues		
	Hypothesis:	to be prepared beforehand for any contingency __ the greatest of virtues		
2	Typed:	i freaked out by the *possibilitu* of someone else *reafing* this	Direct, on-bus Kodak 114x149	0.65
	Reference:	i freaked out by the possibility of someone else reading this		
	Hypothesis:	i *created* out by the possibility of *committee* else reading *the*		
3	Typed:	i can levitate birds	Sunglasses Sanyo 92x107	0.54
	Reference:	i can levitate birds		
	Hypothesis:	i can *hesitate* birds		

Under "Scenario" is given the details of the capture scenario for each hypothesis, including camera type and phone image resolution. Note that Sentence 2 was captured on a bus.

estimated the maximum distance at which we could capture video and still be able to accurately reconstruct the typed text; more specifically, we determine, for each device, the distance at which the METEOR score of our system drops below the 0.5 threshold of being "understandable". The results for this experiment are illustrated in Fig. 9, which shows the case of direct surveillance (top) as well as sunglass reflections (middle).

The main takeaway from these figures is that for both direct and indirect threat models, *very low cost, small form-factor* capture devices are sufficient to enable accurate reconstruction of typed text under realistic settings. For the direct case, for instance, note that even simple pocket cameras are capable of operating at approximately 3 meters, which is a very realistic setting for a shoulder-surfing attack. The corresponding range for the case of sunglass reflections is lower, because the size of the phone in a reflection is much smaller. Thus, for this case, while the lowest class of device (i.e., a pocket camera) is relatively impractical, a simple point-and-shoot camera can be used from about 2.5 meters away—a distance that may not be noticed by an unobservant victim.

Of course, an attacker with a larger budget can use higher end devices—such as an SLR camera with a zoom lens—which enables eavesdropping at much larger distances. For the direct case, we were able to achieve accurate results at distances of $\approx$61 meters, which could now represent a hypothetical attacker standing on an upper floor of a building and recording people through a window (as in Fig. 1). The range for sunglass reflections is again lower, but still very realistic; when using an SLR camera, we achieved a maximum distance of $\approx$12 meters.

4.2.2 Experiment #2: Isolated Word-Unit Matching

Next, we consider a more specific scenario, where an adversary might not wish to apply the dictionary matching and language modeling stages. One such example is the case of recovering passwords, which are often nondictionary strings, with numbers and special characters. In this context, it would, thus, be informative to analyze the raw accuracy of our system *without* the language modeling steps.

For this experiment, we provided users with randomly sampled passwords from the Sony BMG Netherlands

database,[7] and used the Canon VIXIA HG21 video camera to capture a total of 10 surveillance videos, including both direct and indirect threat models, using the same capture settings as in Experiment 1 (Section 4.2.1). We ran each recorded video stream through our system, but did not apply the edit distance, dictionary matching and language modeling stages. In other words, the WFST in Stage ⑤ of our system now consists of only the frame parser module.

For this experiment, the passwords we used consisted of a combination of letters, numbers, and special characters (for instance, "des8gn@H"). Most on-screen keyboards have an option to toggle the keyboard layout between alphabets and numbers/special characters. As noted in Section 3.4, there are various ways in which this scenario can be addressed in our system. We choose a simple strategy in this work: we simply train classifiers for each key on each keyboard mode as before, and when running the classifiers on a test frame, we output the key (across all keyboard modes), which gathered the strongest response. In other words, this simple approach relies on the hypothesis that often, for a specific pop-out location on the keyboard, the keys in the various keyboard modes are sufficiently different in appearance in order for the classifier to tell them apart.

In terms of evaluation, note that for this experiment, the METEOR score is not well suited to evaluating accuracy, because we are interested in the ability to reconstruct isolated *word units*, i.e., sequences of contiguous nonspace characters, rather than phrases. For this reason, we record precision and recall scores, based on the number of password characters that match between what was actually typed, and our reconstructed text. For the 10 passwords tested in this experiment, we achieved an average precision of 0.97, with an average recall of 0.92. In the context of recovering passwords, these high precision/recall scores imply that the search space for any subsequent algorithm is significantly reduced [40], thus making it much easier for an attacker to accurately recover passwords.

As an additional experiment, we can also perform the same word-unit-based analysis on the data sets captured for Experiment #1. In other words, we can process the same sets of sentences as in Experiment #1, but without the application of edit distance, dictionary, or language models. In this analysis, we achieve precision and recall, respectively, of 0.75 and 0.78 for direct surveillance and 0.64 and 0.65 for indirect surveillance—in all cases, the accuracy is high enough to recover more than half of any typed words. In addition, our single-character precision and recall scores are 94 and 98 percent, respectively, in the direct case, and 92 and 97 percent in the indirect case—again demonstrating that our system is certainly accurate enough for password guessing, particularly given that we have a reasonable prior distribution over characters to drive password space exploration.

4.2.3 Experiment #3: OCR

While our main focus in this work has been to exploit the "pop-out" characteristics of on-screen keyboards via a video stream of the typing activity, it is worth comparing our video-based attack against an alternate, and perhaps more obvious, attack: using OCR techniques to directly read the text displayed on the screen of the devices. This alternate attack has some notable advantages: 1) It does not require the capture of video sequences; rather, a single *still image* of the screen would be sufficient to reconstruct the displayed text; and 2) capturing a still image typically yields a higher resolution image than a video recording; thus, for example, it might be possible to use digital SLR cameras coupled with zoom lenses, to capture the screen of the device from a much larger distance. Note that Backes et al. [2], [3] exploit essentially this kind of attack, using high-end telescopic lenses to capture reflections of computer monitors. However, while their work focuses more on the *acquisition* of these images, we are more interested in whether these images can be *automatically* interpreted using OCR-based techniques.

TABLE 2
Synthetic Results: METEOR Scores for Each OCR Engine at Varying Image Resolutions

OCR Method	Image Resolution				
	450×300	300 × 200	225 × 150	180 × 120	129 × 86
OnlineOCR	1.0	0.0	0.0	0.0	0.0
Google Docs	0.30	0.30	0.0	0.0	0.0
FreeOCR	0.51	0.0	0.0	0.0	0.0
Tesseract OCR	1.0	0.0	0.0	0.0	0.0
ABBYY FineReader	1.0	0.55	0.21	0.0	0.0
OmniPage	0.48	0.0	0.0	0.0	0.0

As a simple "synthetic" experiment, we first capture a screenshot of passage of text displayed on the screen of the device, and downsample the resulting image to mimic the effect of decreasing resolution (see Table 2, which lists the resolutions used). Note that this provides the best possible input to the OCR system—i.e., images with no camera imaging noise, occlusion, illumination variation, or perspective distortion. In other words, running an OCR engine on these synthetic images should provide a baseline for their accuracy for this task, under the most favorable operating conditions.

We compare the performance of six OCR engines that are freely available on the web: OnlineOCR, Google Docs, FreeOCR, Tesseract OCR, ABBYY FineReader, and OmniPage. These engines were selected to be a representative sample of the current state of the art in OCR. Each of these engines was run on the synthetic images at a range of resolutions (see Table 2). Note that the accuracy of even the best OCR engine drops dramatically for a moderate reduction in pixel resolution. This suggests that in a practical capture scenario, the accuracy of the OCR attack will depend significantly on the size of the phone in the captured images. Note that the video-based attack does not display this strong correlation; as shown in Fig. 8b, our attack operates robustly over a wide range of input resolutions.

To further evaluate the applicability of OCR techniques under realistic capture conditions, we now capture still images of the screen of the phone, from a clear, fronto-parallel viewpoint. It is worth noting that we could also capture images from an oblique viewpoint and run them through Stage ③ of our system (alignment to reference image) to obtain a fronto-parallel image. However, because our main focus in this experiment is to test the accuracy of OCR, we opt to directly capture a frontal view. While our

7. See WIRED's "Sony Hit Yet Again; Consumer Passwords Exposed."

TABLE 3
OCR Results for Direct Surveillance Using a Digital SLR Camera with Zoom Lens

Focal length (mm)	Distance (m)	Reconstructed sentence	METEOR score
70	3.0	the multiple human needs and desires that demand privacy among two or more people in the midst of social life must inevitably lead to cryptology wherever men thrive and wherever they write	1.0
70	4.0	the multiple human needs and desires that demand privacy among two or more people m the midst of social irfe must mevitabty lead to cryptology wherever men thrive and wherever they wnte	0.51
70	5.0	-	0.0
400	8.0	the multiple human needs and desires that demand privacy among two or more people in the midst of social life must inevitably lead to r.ryptrjloqy wherever men thrivn and whmnvi'i fhoy wrlln	0.49
400	9.0	the multiple human nee<ls and desires that demand pnvacy among rwo or moce people m the midst of sooa Âğ*e must inevitably lead to cryptology men thrive and wherever th#?v *rite	0.33
400	10.0	-	0.0

synthetic experiments involved artificially downsampling the images, in this case, we now alter the distance between the phone and the camera to capture this effect. For this experiment, we used a Canon EOS 60D camera, with the focal length of the lenses used varying between 70 and 400 mm. This, thus, represents a recording device that provides very high-resolution ($5{,}184 \times 3{,}456$) images. For this experiment, we use the best performing OCR engine from the synthetic experiments (ABBYY FineReader). A representative selection of the results are shown in Table 3.

The main take-home message from these results is that while the OCR attack is indeed simpler, it is viable only up to a point, beyond which it quickly becomes far too inaccurate. For the case of direct surveillance, for instance, using an OCR-based attack, an attacker with a digital SLR camera and 400-mm zoom lens can be up to 8 m away from the user and be able to accurately (i.e., with a METEOR score > 0.5) reconstruct the on-screen text. These results are far weaker than with our proposed approach, where we achieve a corresponding maximum distance for direct surveillance, using exactly the same equipment, of 61 m—almost 8 times farther than the OCR attack.

The main reason for the relative inaccuracy of OCR is due to the fact that, particularly at low resolutions, it is often hard to distinguish between different letters. Some examples are shown in Fig. 10; these letters were cropped from one of the videos we recorded for the experiments in Section 4.2.1. Note that it is difficult, even for humans, to identify these letters, making it an impossibly hard task for an OCR engine. The reason our system is able to handle this case is due to the fact that we do not attempt to recognize these letters in isolation; rather, we take spatial context into account. More specifically, because we align each frame of video, and train a different classifier for each keyboard location, our system does not ever need to distinguish *between* characters. Each trained classifier in Stage ④ only needs to distinguish between a popped-out letter and its background, which is a much simpler problem, as shown in Fig. 11. Stated differently, we transform a 26-class classification problem (considering letters a-z) in the case of OCR, into a 2-class classification problem (that of distinguishing between a letter and its background). In this sense, our system can actually be viewed as a highly specialized OCR system that is able to leverage spatial information, via the location of the keys on the keyboard, to recognize key presses. This fact enables us to operate effectively over a much wider range of distances compared to the OCR attack.

Fig. 10. Low-resolution images of some characters from the iPhone keypad. Can you recognize these characters? Note that at these low resolutions, it is an exceedingly difficult task for humans—let alone OCR engines—to recognize the character corresponding to each image. In the absence of additional information, this is the main reason OCR systems often fail completely when applied to this task. (Answer: reading left to right, C D G O Q).

5 Summary and Limitations

We explore the feasibility of automatically reconstructing typed input in low resolution video, of, for example, compromising reflections, captured in realistic scenarios. While our results are certainly disconcerting, it is prudent to note that there are some important issues that remain open. Low-pixel resolution of the phone image is one of the key problems we encountered. It can be caused by a variety of

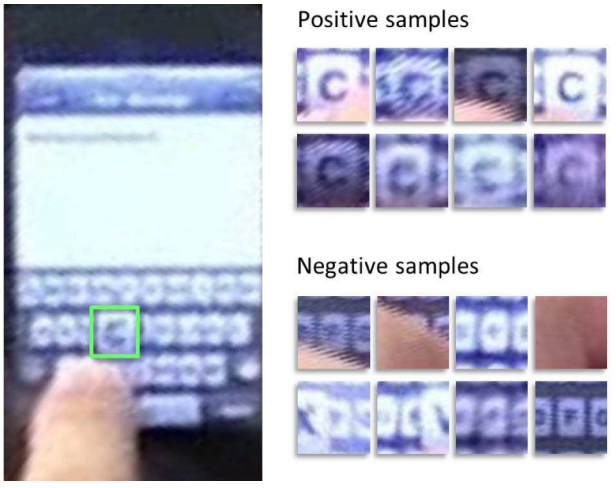


Fig. 11. Left: Aligned iPhone keypad, with the "popped-out" letter C highlighted. Right: Extracted patches, corresponding to positive and negative training samples for our key-press classifier for the letter C. Note that this two-class classification problem is much easier to solve than the problem illustrated in Fig. 10.

factors, including camera aperture, wide angle lenses, and large effective capture distance. While capturing data on the bus, we sometimes encountered motion blur artifacts, caused by excessive camera jitter. All of these make the phone's appearance so blurry that no reliable features can be extracted, and so our phone stabilization (Stage ②) and alignment (Stage ③) methods fail in certain cases. We believe this could be addressed by using more sophisticated (and expensive) capture techniques, as in [3], which addresses the allied problem of *capturing* clear images from reflections.

Finally, there are potential defenses against the attacks proposed in this work. One, in the indirect case, is the application of an antireflective coating, such as is common on modern eyeglasses, on the reflection surface. Reducing the brightness of the screen would also have a detrimental effect on any reconstruction. Finally, one might disable the visual key press confirmation mechanism that we leverage in this work. Obviously, our approach is not applicable to situations where there is no visual key press confirmation. Hence, devices that lack this effect—for instance, tablets, or devices that use drag-based input mechanisms (e.g., Swype)—are not vulnerable to our attack. How to effectively handle these kinds of devices is an interesting direction to explore. Lastly, as suggested by Backes et al. [3], one could use secondary reflections in the environment when direct line-of-sight to the target is infeasible.

Nevertheless, the fact that we can achieve such high accuracy underscores the practicality of our attack, and aptly demonstrates the threats posed by emerging technologies.

Acknowledgments

This work was supported in part by the US National Science Foundation (NSF) grant CNS-0852649. An earlier version of this paper appeared in the 2011 ACM Conference on Computer and Communications Security.

References

[1] R. Raguram, A.M. White, D. Goswami, F. Monrose, and J.-M. Frahm, "iSpy: Automatic Reconstruction of Typed Input from Compromising Reflections," *Proc. ACM Conf. Computer and Comm. Security,* pp. 527-536, 2011.

[2] M. Backes, M. Dürmuth, and D. Unruh, "Compromising Reflections-or-How to Read LCD Monitors around the Corner," *Proc. IEEE Symp. Security and Privacy,* 2008.

[3] M. Backes, T. Chen, M. Duermuth, H. Lensch, and M. Welk, "Tempest in a Teapot: Compromising Reflections Revisited," *Proc. IEEE Symp. Security and Privacy,* 2009.

[4] National Security Agency, "TEMPEST: A Signal Problem," *Cryptologic Spectrum,* vol. 2, no. 3, 1972.

[5] H.J. Highland, "Electromagnetic Radiation Revisited," *Computer Security,* vol. 5, pp. 85-93, June 1986.

[6] W. van Eck, "Electromagnetic Radiation from Video Display Units: An Eavesdropping Risk?" *Computer Security,* vol. 4, pp. 269-286, 1985.

[7] J. Loughry and D.A. Umphress, "Information Leakage from Optical Emanations," *ACM Trans. Information and System Security,* vol. 5, pp. 262-289, Aug. 2002.

[8] M.G. Kuhn, "Optical Time-Domain Eavesdropping Risks of CRT Displays," *Proc. IEEE Symp. Security and Privacy,* 2002.

[9] M. Kuhn, "Electromagnetic Eavesdropping Risks of Flat-Panel Displays," *Proc. Fourth Workshop Privacy Enhancing Technologies,* 2004.

[10] M. Vuagnoux and S. Pasini, "Compromising Electromagnetic Emanations of Wired and Wireless Keyboards," *Proc. 18th USENIX Security Symp.,* 2009.

[11] D. Asonov and R. Agrawal, "Keyboard Acoustic Emanations," *Proc. IEEE Symp. Security and Privacy,* 2004.

[12] L. Zhuang, F. Zhou, and J.D. Tygar, "Keyboard Acoustic Emanations Revisited," *ACM Trans. Information and System Security,* vol. 13, article 3, 2009.

[13] P. Marquardt, A. Verma, H. Carter, and P. Traynor, "(sp)iPhone: Decoding Vibrations from Nearby Keyboards Using Mobile Phone Accelerometers," *Proc. ACM Conf. Computer and Comm. Security,* 2011.

[14] F. Maggi, A. Volpatto, S. Gasparini, G. Boracchi, and S. Zanero, "A Fast Eavesdropping Attack against Touchscreens," *Proc. Seventh Int'l Conf. Information Assurance and Security (IAS),* 2011.

[15] D. Balzarotti, M. Cova, and G. Vigna, "ClearShot: Eavesdropping on Keyboard Input from Video," *Proc. IEEE Symp. Security and Privacy,* 2008.

[16] K. Jung, K.I. Kim, and A.K. Jain, "Text Information Extraction in Images and Video: A Survey," *Pattern Recognition,* vol. 37, no. 5, pp. 977-997, 2004.

[17] P.A. Viola and M.J. Jones, "Rapid Object Detection Using a Boosted Cascade of Simple Features," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* 2001.

[18] P.A. Viola and M.J. Jones, "Robust Real-Time Face Detection," *Int'l J. Computer Vision,* vol. 57, no. 2, pp. 137-154, 2004.

[19] H. Grabner, M. Grabner, and H. Bischof, "Real-Time Tracking via On-Line Boosting," *Proc. British Machine Vision Conf.,* vol. 1, pp. 47-56, 2006.

[20] S. Stalder, H. Grabner, and L.V. Gool, "Beyond Semi-Supervised Tracking: Tracking Should Be as Simple as Detection, but Not Simpler Than Recognition," *Proc. Workshop On-Line Learning for Computer Vision,* pp. 1409-1416, Sept. 2009.

[21] Z. Kalal, K. Mikolajczyk, and J. Matas, "Forward-Backward Error: Automatic Detection of Tracking Failures," *Proc. Int'l Conf. Pattern Recognition,* 2010.

[22] H. Grabner, C. Leistner, and H. Bischof, "Semi-Supervised On-Line Boosting for Robust Tracking," *Proc. European Conf. Computer Vision,* pp. 234-247, 2008.

[23] Y. Freund and R.E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *Proc. Second European Conf. Computational Learning Theory,* pp. 23-37, 1995.

[24] K. Levi and Y. Weiss, "Learning Object Detection from a Small Number of Examples: The Importance of Good Features," *Proc. IEEE CS Computer Vision and Pattern Recognition,* 2004.

[25] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns," *IEEE Trans. Pattern Anal. Machine Intelligence,* vol. 24, no. 7, pp. 971-987, July 2002.

[26] R.I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision.* Cambridge Univ. Press, 2000.

[27] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *Int'l J. Computer Vision,* vol. 60, no. 2, pp. 91-110, 2004.

[28] P.J. Huber, *Robust Statistics.* John Wiley & Sons, 1981.

[29] M.A. Fischler and R.C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Comm. ACM,* vol. 24, no. 6, pp. 381-395, 1981.

[30] R. Raguram, J.-M. Frahm, and M. Pollefeys, "A Comparative Analysis of RANSAC Techniques Leading to Adaptive Real-Time Random Sample Consensus," *Proc. European Conf. Computer Vision: Part II,* pp. 500-513, 2008.

[31] D. Jurafsky and J.H. Martin, *Speech and Language Processing.* Prentice Hall, 2008.

[32] F.C.N. Pereira and M. Riley, "Speech Recognition by Composition of Weighted Finite Automata," *Computing Research Repository,* vol. cmp-lg/9603001, 1996.

[33] W.N. Francis and H. Kucera, "Brown Corpus Manual," technical report, Dept. of Linguistics, Brown Univ., 1979.

[34] I.A. McCowan, D. Moore, J. Dines, D. Gatica-Perez, M. Flynn, P. Wellner, and H. Bourlard, "On the Use of Information Retrieval Measures for Speech Recognition Evaluation," Idiap-RR 73-2004, Idiap Research Institute, 2004.

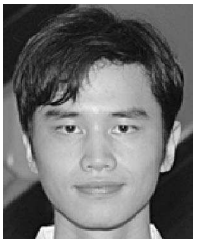
[35] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: A Method for Automatic Evaluation of Machine Translation," *Proc. 40th Ann. Meeting Assoc. Computational Linguistics,* pp. 311-318, 2002.

[36] G. Doddington, "Automatic Evaluation of Machine Translation Quality Using N-Gram Co-Occurrence Statistics," *Proc. Second Human Language Technologies Conf, (HLT '02),* pp. 128-132, 2002.

[37] A. Lavie and M.J. Denkowski, "The METEOR Metric for Automatic Evaluation of Machine Translation," *Machine Translation*, vol. 23, pp. 105-115, Sept. 2009.
[38] M. Denkowski and A. Lavie, "Choosing the Right Evaluation for Machine Translation: An Examination of Annotator and Automatic Metric Performance on Human Judgment Tasks," *Proc. Conf. Assoc. Machine Translation in the Am. (AMTA)*, 2010.
[39] A. Lavie, "Evaluating the Output of Machine Translation Systems," *Proc. Conf. Assoc. Machine Translation in the Am. (AMTA)*, 2010.
[40] Y. Zhang, F. Monrose, and M.K. Reiter, "The Security of Modern Password Expiration: An Algorithmic Framework and Empirical Analysis," *Proc. ACM Conf. Computer and Comm. Security*, pp. 176-186, 2010.
[41] L. von Ahn, B. Maurer, C. Mcmillen, D. Abraham, and M. Blum, "RECAPTCHA: Human-Based Character Recognition via Web Security Measures," *Science*, vol. 321, no. 5895. pp. 1465-1468, 2008.

Rahul Raguram is working toward the PhD degree in the Department of Computer Science at the University of North Carolina at Chapel Hill. His research interests include robust estimation, large-scale structure from motion from video and internet photo collections, 2D/3D scene analysis and segmentation, applications of computer vision in security/privacy, and image and video compression.

Andrew M. White received the degrees in computer science and mathematics from the University of Richmond in 2008, the master's degree in 2011, and is working toward the PhD degree in the Department of Computer Science at the University of North Carolina at Chapel Hill. In 2008, he received of the Mary Church Kent and Joseph F. Kent Computer Science Prize. His research interests include network and distributed system security and privacy and machine learning.



Yi Xu is working toward the PhD degree in the Department of Computer Science at the University of North Carolina at Chapel Hill. His research interests include computer vision and its applications in the security field, such as the secure design of CAPTCHA and the privacy problem of mobile device screen.

Jan-Michael Frahm received the PhD degree in computer vision from the Christian-Albrechts University of Kiel, Germany, in 2005. He is currently an assistant professor at the University of North Carolina at Chapel Hill. His research interests include structure from motion, real-time multiview stereo, camera-sensor systems for 3D scene reconstruction, robust estimation methods, high-performance feature tracking, and the development of data-parallel algorithms for commodity graphics hardware.

Pierre Georgel received the PhD degree in computer science from the Technical University Munich, Germany, in 2011. He is a principal engineer at Deko Inc. His interests include 3D reconstruction from images, augmented reality, and image based rendering.

Fabian Monrose received the PhD degree in computer science from the Courant Institute of Mathematical Sciences, New York University, in 1999. He is a professor of computer science at the University of North Carolina at Chapel Hill. His interests span the fields of networking and security and includes such diverse topics as traffic classification, computer forensics, user authentication, and privacy.