POSTER: Fast, Automatic iPhone Shoulder Surfing

Federico Maggi
Alberto Volpatto
Politecnico di Milano
volpatto, fmaggi
@elet.polimi.it

Simone Gasparini
INRIA Grenoble
Rhone-Alpes
simone.gasparini
@inrialpes.fr

Giacomo Boracchi
Stefano Zanero
Politecnico di Milano
boracchi, zanero
@elet.polimi.it

ABSTRACT

Touchscreen devices increase the risk of shoulder surfing to such an extent that attackers could steal sensitive information by simply following the victim and observe his or her portable device. We underline this concern by proposing an automatic shoulder surfing attack against modern touchscreen keyboards that display magnified keys in predictable positions. We demonstrate this attack against the Apple iPhone—although it can work with other layouts and different devices—and show that it recognizes up to 97.07% (91.03% on average) of the keystrokes, with only 1.15% of errors, at 37 to 51 keystrokes per minute: About eight times faster than a human analyzing a recorded video.

Our attack, described thoroughly in [2], accurately recovers the sequence of keystrokes input by the user. The attack described in [1], which targeted desktop scenarios and thus worked with very restrictive settings, is similar in spirit to ours. However, as it assumes that camera and target keyboard are both in fixed, perpendicular position, it cannot suite mobile settings, characterized by moving target and skewed, rotated viewpoints. Our attack, instead, requires no particular settings and even allows for natural movements of both target device and shoulder surfer's camera. In addition, our attack yields accurate output without any grammar or syntax checks, so that it can detect large context-free text or non-dictionary words.

In summary:

- We are the first studying the practical risks brought forth by mainstream touchscreen keyboards.
- We design a practical attack that detects keystrokes on modern touchscreen keyboards: The attacker requires not to stand exactly behind the victim nor to observe the screen perpendicularly.
- Our attack is robust to occlusions (e.g., typing fingers), thanks to our efficient filtering technique that validates detected keys and reconstructs keystroke sequences accurately.

1 Threat Model & Requirements

The attacker can point a camera toward the target touchscreen while the victim enters a text. No visibility of typed text is required.

Requirement 1: The virtual keyboard must display a partially visible *magnified key* upon each keystroke, or at least in one frame after each keystroke. The attack works even when fingers partially cover the magnified keys, as it typically happens while typing. Key magnification is often enabled by default in popular touchscreen phones and sometimes cannot be cannot be deactivated (except in some older versions of Android OS).

Requirement 2: The attacker must know:

- *Screen template*: Screenshot of the device's virtual keyboard.
- *Key template*: Appearance of each magnified key.
- *Magnified layout*: Coordinates of the magnified key centers.

This requirement is easily met by building an offline database of magnified keyboard layouts (e.g., by taking screenshots) of any target device.

2 Automatic Shoulder Surfing Attack

Our system processes a stream of images frame by frame as depicted in Figure 1.

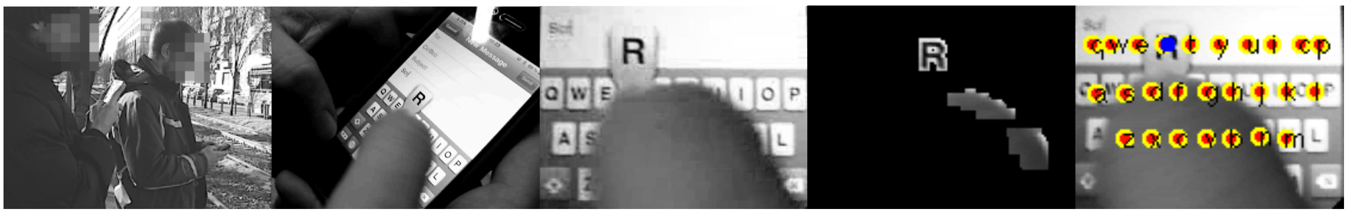
Phase 1	**Screen Detection and Rectification**
Input	Current frame.
Task	Detect the touchscreen image by leveraging a template of the target screen. When a match is found in the current frame, rectify and crop the screen area.
Output	A rectified, cropped and scaled image of the device screen in the current frame. This is close to the image that a fixed camera had acquired when the device is at a fixed distance, with its screen parallel to the camera.
Phase 2	**Magnified Keys Detection**
Input	Rectified image of the target screen.
Task	Isolate magnified-key candidates, i.e., high-contrast areas of the rectified image that are different from the template and previous frames.
Output	A segmented image (i.e., a map of the image areas) identifying the magnified-key candidates (blobs). Typically, there is more than one blob per frame.
Phase 3	**Keystroke Sequence Recognition**
Input	A set of magnified-key candidates.
Task	Filter out wrong candidates, by matching them with the corresponding template of the magnified key, thus identifying the best-matching key.
Output	The symbol, if any, of the best-matching key.

We implemented a working prototype of both Phase 1 (with OpenCV) and Phase 2–3. We recorded a video demonstration of the intermediate outputs of these phases and published it at `http://www.youtube.com/playlist?list=PL81F91E404B928833`.

3 Evaluation

We spied on six "victims" with a low-resolution camera (i.e., 640 by 480 pixels @ 25fps), each while typing:

- *Context-free text:* 63 English words, such that attackers cannot simply guess the words (`http://sqze.it/qMNwy`).
- *Context-rich text:* 65 words from the lyrics of Dream Theater's *"Regression"* song.
- *Brief text:* Used to evaluate specific features and limitations:



(a) Envisioned attack settings. (b) Input of Phase 1. (c) Input of Phase 2. (d) Input of Phase 3. (e) Output (i.e., the 'R' key).

Figure 1: Intermediate outputs captured in a sample attack (a). Phase 1: We detect the device screen in each input frame (b), crop and rectify it, obtaining (c). Phase 2: We select the magnified-key candidates within the foreground (i.e., image areas shown in (d)). Phase 3: According to the coordinates of the magnified keyboard layout (e), we compare each candidate to its template to identify the typed key. The template of 'R' exhibits high similarity with one of the candidates, as show by the local maximum in Figure 2 around frame 985.

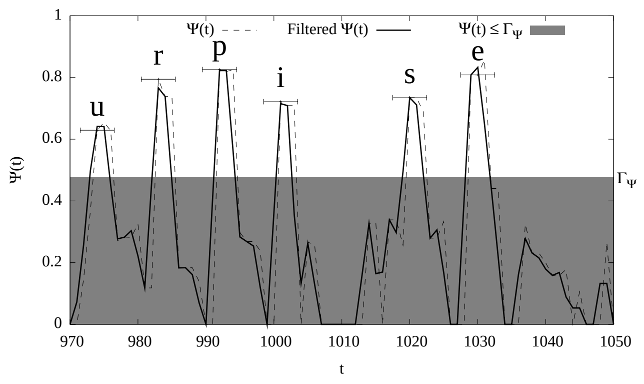


Figure 2: Keystroke sequence recognition: Key magnifications typically last longer than one frame, and there are frames that contains no magnified keys, thus it is insufficient to identify the best-matching key at each time step. We first low pass the key similarity measure of each best matching key and then search for local maxima. Frames yielding low similarity are discarded. The horizontal bars above each local maximum indicate the minimum distance between two local maxima, which can be considered as the minimum number of frames the key magnification lasts (e.g., five frames). This choice only influences the *maximum* typing speed handled by our system: Even when victims type *slower* than one stroke per five frames, we recognize magnified keys correctly. However, one stroke per five frames is a very high typing speed at 25fps.

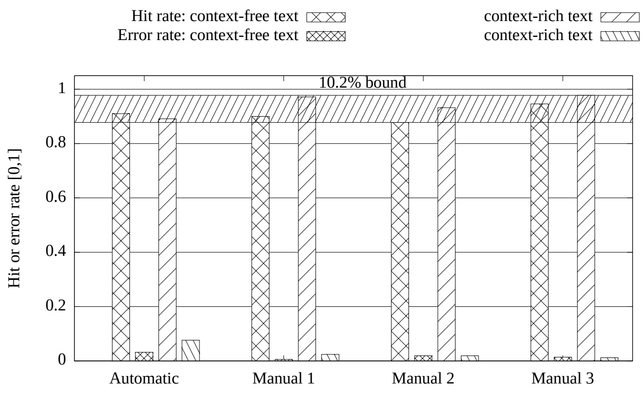
"Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas".

We had to perform recording because we needed repeatable and comparable experiments, although our attack works perfectly online, on streams of images. Unless differently stated, we kept the *handheld* camera at an angle such that our system can recognize the screen.

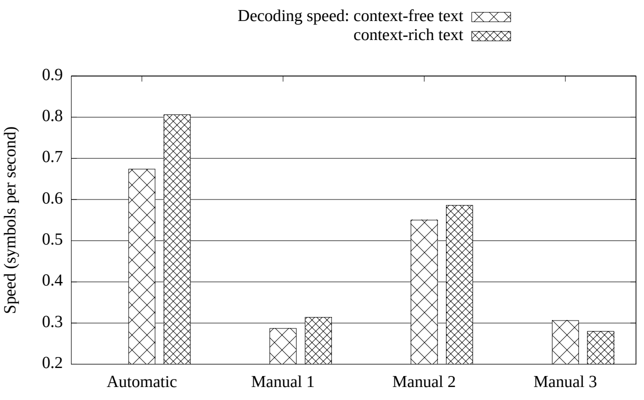
3.1 Experiment 1: Precision and Speed

We performed 3 sessions on context-free text and 3 on context-rich text, each with a different victim. In particular, we asked the victims to type naturally, as they would do in their daily activities. Each typing session was processed by our system and by a *different* attacker for each victim. Without any a priori knowledge, each attacker had to recognize the keystrokes by stopping, rewinding or slowing-down the recorded stream as needed.

As shown in Figure 3b, regardless of the text's context, manual recognition is notably slower than our system. For example, our system can recognize, on average, up to 0.803 keystrokes per second, about one third of the maximum typing speed, and 0.864 in the best case, about half of the average typing speed. Only twice (over 18 trials) the attackers were able to beat such speeds. As expected, human sight can recognize symbols with slightly higher precision than our system. This is more evident in the context-rich text experiment, where our system is outperformed by about 8 percentage points. On



(a) Hit and error rate.



(b) Recognition speed (keystrokes per second).

Figure 3: Comparison of average precision (a) and speed (b) of our automatic detector versus human attackers. Precision (i.e., hit and error rates) remain within shallow bounds, whereas the speed of manual recognition is significantly low with respect to our automated attack.

the other hand, even with no spell correction, on context-free text for example, our system is just 3 percentage points less accurate than the attacker with the highest average precision. Hence, our system is at least effective as offline inspection of a video, although remarkably faster and, more importantly, tireless. Indeed, all the volunteers described this session as an extremely tedious task, certainly not doable for extended periods of time.

These results are more interesting if we take the typing speed into account. For example, one of the victims typed at slow speed and, as expected, humans were able to recognize keystrokes very efficiently. In case of faster typing, manual analysis is more error prone, whereas our system is not influenced by typing speed. Recall that we set a minimum distance of 5 frames between adjacent keystrokes, although this choice only influences the *maximum* typing

MEASURES	IDEAL	REAL-WORLD (AVG.)	DIFFERENCE
	PRECISION %		
Hits %	95.12	91.03	4.09
Errors %	1.01	3.16	-2.15

Table 1: Ideal conditions vs. real-world conditions.

speed handled by our system. Therefore, even when victims type *slower* than such speed, magnified keys are recognized correctly. It is worth noticing that one stroke every 5 frames is a very high typing speed at 25fps.

The state-of-the-art system [1], even under more relaxed hypotheses, works at 0.101 keystrokes per second on average, with a maximum precision of 82%, only achievable in context-sensitive text. Our attack needs no linguistic context to recognize 97.77% of the keystrokes at 0.803 keystrokes per second.

3.2 Experiment 2: Ideal Versus Real World

In the previous experiment we assessed the feasibility of the attack in natural conditions: The camera was *handheld* and the victim was allowed to *move naturally*. The goal of this second experiment is instead to evaluate Phase 1 and 2–3, separately, thus allowing us to measure the maximum achievable performance of Phase 2–3 in ideal conditions. We positioned the recording camera on a tripod and fixed the touchscreen on a table, with its screen perfectly aligned and parallel to the camera's sensor. Then, we run our system with Phase 1 *disabled* on the video depicting the victim while typing the context-free text. Hit and error rates are summarized in Table 1.

As one may expect, under ideal conditions Phase 2–3 alone reach remarkable hit rate and error rate: Magnified keys are never occluded (e.g., the victim's finger is always below each magnified key), a less common case in real-world settings (e.g., because of the camera's skewed viewing angle). However, in Section 3.1 we showed that, in real-world settings, Phase 1–3 together can reach up to 97.07% hit rate.

3.3 Experiment 3: Resilience to Disturbances

Our goal was to stress the robustness of Phase 1. To this end, we performed a series of brief typing sessions and included several significant disturbances. In practice, we asked a victim to type the brief text under the following conditions: (1) we attached a piece of gray tape diagonally on the screen to emulate a permanent occlusion, (2) we asked the victim to jiggle the device while typing, (3) we jiggled the camera during recording, and (4) both the camera and the target device were jiggling during recording.

Table 2 shows that Phase 1 was able to rectify parts of the video and thus recognize the screen at some point of the video. In the best case we were able to recognize 96% of the symbols with 4% errors. In the worst case, we are able to detect 44.44% of the keystrokes, although the errors caused by the difficulty of dealing with objects that cover the screen permanently are quite high. However, users seldom hold touchscreen devices with permanent occlusions, especially while typing on the go. Our system can thus handle sudden movements of either camera or device, whereas Phase 1 fails when both camera and device move excessively, causing intra-frame, motion-blurring side effects that affect feature extraction.

DISTURBANCE	PHASE 1	PHASE 2–3	
		Hits %	Errors %
(1) Permanent occlusion	difficult	44.44	33.33
(2) Shake device	feasible	67.74	8.70
(3) Shake camera	feasible	96.00	4.00
(4) Shake device + camera	unfeasible	0.00	-

Table 2: Detection results under different working conditions.

4 Countermeasures

The usability of modern touchscreen keyboards is exploitable to such an extent that we were able to implement an efficient and precise automatic shoulder surfing attack. The definitive countermeasure consists in disabling key magnification, disallowed in iPhone as well as in newer versions of Android and BlackBerry OSs. Users of older Android versions can switch this feature on and off, yet they typically leave it enabled. The reason is because on mechanical keyboards the user's focus is on the text that appears on the display, whereas on virtual keyboards even very small shifts in the position of the fingers can cause typing errors, reduced by key magnification.

Mechanical keyboards are less privacy leaking with about the same degree of usability of magnifying touchscreen ones. For instance, we recorded a typing sessions on a BlackBerry[1], using a long English text with no linguistic context (context-free text described in Section 3). Six volunteers who analyzed the video reported that most of the keystrokes were not actually visible on the BlackBerry. Also, even with "unlimited" time and the possibility of stopping, slowing down and restarting the videos as needed, only one volunteer reconstructed a negligible (1.35%) portion of the text. The remainder two volunteers gave up for excessive fatigue. From this experiment we argue that spying on non-touchscreen mobile keyboards is hard. First, the compactness of the keyboard forces the victim to small and almost unperceivable fingers movements. Second, keys are very small and, typically, a user covers multiple keys at once while typing making it impossible even for a human to distinguish among them. On touchscreen keyboards instead, shoulder surfers were able to recognize between 95% and 100% of keystrokes, with at most 2.6% of errors, as discussed in Section 3.1.

5 Portability and Limitations

Our system's main limitation revolves around the fact that keystrokes are recognizable as long as visual feedback such as magnified keys is displayed. Therefore, if some keys are not magnified our current implementation would not detect them. Similarly, special characters and numbers on some devices, including the iPhone, are selected on different magnified layouts. Analogous observations apply to alternative layouts (e.g., landscape). Dealing with these peculiarities on our system would require minor modifications. Specifically, instead of using only one screen template (e.g., portrait-alphabetical), Phase 1 would cycle through several alternative layouts (e.g., portrait alphabetical, portrait numerical, landscape numerical) and choose the best-matching one before. Similarly, Phase 2 would need additional key templates.

6 Conclusions

Keyboards that employ key-magnification feedback are unsuitable for high-privacy applications and, given that the most popular touchscreen devices display such feedback, the risk brought forth by our system is remarkable.

7 References

[1] BALZAROTTI, D., COVA, M., AND VIGNA, G. ClearShot: Eavesdropping on Keyboard Input from Video. In *Proc. of the IEEE Symposium on Security and Privacy* (Oakland, CA, May 2008).

[2] MAGGI, F., VOLPATTO, A., GASPARINI, S., BORACCHI, G., AND ZANERO, S. Don't touch a word! a practical input eavesdropping attack against mobile touchscreen devices. Tech. Rep. TR-2010-59, Politecnico di Milano, 2010.

[1]We recorded this session: `http://www.youtube.com/watch?v=JxrqYA56A48`