TouchLogger: Inferring Keystrokes On Touch Screen From Smartphone Motion

Liang Cai
Univ. of California, Davis

Hao Chen
Univ. of California, Davis

Abstract

Attacks that use side channels, such as sound and electromagnetic emanation, to infer keystrokes on physical keyboards are ineffective on smartphones without physical keyboards. We describe a new side channel, motion, on touch screen smartphones with only soft keyboards. Since typing on different locations on the screen causes different vibrations, motion data can be used to infer the keys being typed. To demonstrate this attack, we developed TouchLogger, an Android application that extracts features from device orientation data to infer keystrokes. TouchLogger correctly inferred more than 70% of the keys typed on a number-only soft keyboard on a smartphone. We hope to raise the awareness of motion as a significant side channel that may leak confidential data.

1 Introduction

Keyboard is perhaps the most common human input device. We use keyboard to input a variety of information, some of which are highly valuable, such as passwords, PINs, social security numbers, and credit card numbers. It came as no surprise that keystroke logging is a favorite tool of trade by attackers. The attacker can install a Trojan program on the victim computer to log keystrokes, or use out of band channels to infer keystrokes. Acoustic key logger, for example, can infer keystroks from acoustic frequency signatures[2], timings between two keystroks[4], or language models[11]. Electromaganetic emanations of keyboards are also studied for keylogging [8].

Touch screen smartphones have changed the paradigm of user interaction. Most touch screen smartphones have no physical keyboard. Instead, the user types on the software keyboard on the screen. Since there is neither sound nor electromagnetic emanation from a virtual keyboard, the attacker can no longer infer keystrokes based on these signals. Moreover, many smartphone operating systems, such as Android and iOS, restricts privileges granted to applications. In most cases, an application cannot read keystrokes unless it is active and receives the focus on the screen. It seems that key loggers, at least the traditional ones described above, are facing severe obstacles on touch screen smartphones.

In this paper, we investigate a new avenue for keystroke logging on touch screen smartphones. Most of these phones are equipped with a variety of sensors for detecting sound, image, location, and motion. Our insight is that motion sensors, such as accelerometers and gyroscopes, may be used to infer keystrokes. When the user types on the soft keyboard on her smartphone (especially when she holds her phone by hand rather than placing it on a fixed surface), the phone vibrates. We discover that keystroke vibration on touch screens are highly correlated to the keys being typed. In our preliminary evaluation, we were able to infer correctly more than 70% of the keys typed on a number-only soft keyboard on a smartphone.

1.1 Threat model

Currently, to read from the motion sensors, the key logging application needs to be installed on the victim smartphone. Given the increasing number of malware applications on the smartphone market [5] and the prevalence of potentially untrusted third-party ad code incorporated in applications, we do not believe that this assumption is over-optimistic. The user also needs to grant the key logging application the privilege to read from motion sensors. We believe that most users would have no qualm of granting this privilege, as it seems much less risky than other sensor privileges, such as the microphone or camera.

The assumption that most users would not treat motion data as highly sensitive is not just our wishful thinking. W3C has recently published *DeviceOrientation Event Specification* [6] to allow web applications to access accelerometer and gyroscope sensors through Javascript, which both Android 3.0 and iOS4.2 will support. This suggests that our motion-based key logger can be delivered from a website, without requiring the user to install any application.

1.2 TouchLogger

The primary goal of this paper is to raise the awareness of the sensitivity of motion sensor data. To demonstrate the attack, we introduce TouchLogger, which infers keystrokes on touch screen smartphones with motion sensors. Once the user installs TouchLogger and grants it the motion sensor privilege, it starts to monitor motions and infer keystrokes.

2 Modeling and capturing typing-induced smartphone motion

2.1 Modeling typing-induced motion

Since the commercial success of the iPhone, typing on the soft keyboard on smartphones' touch screen has become the most prevalent means of input. Compared to an earlier input method that uses a stylus to touch the screen, typing with a finger causes stronger motion of the smartphone. When we type, we observe that the reflection of distant objects on the screen shifts, and the shift is consistent for each key. This suggests that we may infer keystrokes by the motion of smartphones.

The motion of a smartphone during typing depends on several factors: 1) the striking force of the typing finger; 2) the resistance force of the supporting hand; 3) the landing location of the typing finger; and 4) the location of the supporting hand on the smartphone. The first two factors mainly affect the shift of the phone, while the latter two mainly affect the rotation. We observe that the first two factors likely depend on the user, while the latter two are likely to be user-independent because (1) on each soft keyboard configuration, each key is at a fixed location, and (2) a user typically holds her smartphone in a consistent way. Therefore, we would like to extract the rotation components while filtering out the shift components from motion sensor data.

Most modern smartphone operating systems fire at least two types of events when certain motion is detected: accelerometer event and orientation event[1]. Initially we focused on the accelerometer event because it has higher frequency than orientation event. However, we discovered that data in accelerometer event reflect both shift and rotation, while data in orientation event reflect only rotation. Since we observe that typing-induced rotation is more user independent than shift, we have been using data in orientation events for the rest of our study.

2.2 Device orientation

Data in device orientation measure angles of the device along three axes. On Android, a change in the device orientation triggers an orientation event, which reports a set of intrinsic Tait-Bryan angles (α-β-γ) and the event time t [1].

- α: When the device rotates along the Z-axis (perpendicular to the screen plane), α (*azimuth* angle) changes in $[0, 360)$.
- β: When the device rotates along the X-axis (usually parallel to the shorter side of the screen), β (*pitch* angle) changes in $[-180, 180)$.
- γ: When the device rotates along the Y-axis (usually parallel to the longer side of the screen), γ (*roll* angle) changes in $[-90, 90)$.

On Android, an application reads the motion data by registering a motion sensor event listener, so motion data do not arrive at a constant interval. Both Android and iOS provide three accuracy levels based on event frequencies. The intervals of the motion data also depend on the hardware. For example, at the highest accuracy level, the average interval of device orientation events on an HTC Evo 4G phone is about 30ms, while that on a Motorola Droid phone is about 110ms.

3 Inferring keystrokes via device orientation

We designed and implemented TouchLogger, an Android tool to infer keystrokes on the soft keyboard of smartphones from the device orientation. More precisely, TouchLogger infers the landing locations of the typing finger based on the device orientation and then looks up the corresponding keys based on the current soft keyboard configuration.

3.1 Set up

TouchLogger collects device orientation data when user types on the soft keyboard. The raw device orientation data consists of tuples $(t_i, \alpha_i', \beta_i', \gamma_i'), i = 1 \ldots N$, where t_i is the time of the orientation event, and α_i', β_i' and γ_i' are the azimath, pitch and roll angles of the device, respectively, and N is the number of orientation events in the entire typing session.

For training and testing, we also developed an application to collect key touch events. They consist of tuples $(L_i, t_i^s, t_i^e), i = 1 \ldots M$, where L_i is the label of the key, t_i^s and t_i^e are the starting and ending time of the touch event, respectively, and M is the number of keystrokes in the session.

3.2 Preprocessing

TouchLogger preprocesses the raw device orientation data before it infers keystrokes. First, it discards the azimath angle (α) since rotation caused by typing mainly

[1] Data in orientation events are mainly derived from the accelerometer sensor output and are different from the data in gyroscope events.

affects pitch (β) and roll (γ) angles. Second, it normalizes the angles by eliminating the average angles ($\bar{\beta}'$ and $\bar{\gamma}'$), which represent the initial orientation of the device and are therefore irrelevant to the keystrokes. Finally, to identify the starting and ending time of keystrokes, TouchLogger calculates the Peak-to-Average ratios of the β and γ angles, as these ratios are much larger during typing. Then, during each keystroke (TouchLogger detects the duration of each keystroke based on orientation events rather than keystroke events), TouchLogger converts the raw device orientation data into to a series of tuples $(t_i, \beta_i = \beta'_i - \bar{\beta}', \gamma_i = \gamma'_i - \bar{\gamma}')$. We call each tuple the *motion signal* of a keystroke.

3.3 Basic feature extraction

TouchLogger infers keystrokes based on *features* in motion signals. A good feature should be consistent among motion signals caused by the same keystroke while being distinctive between motion signals caused by different keystrokes.

Because we observed that keystrokes affect the rotation angle of the screen, a naive feature would be the ratio of maximum pitch angle over the maximum roll angle $max(\beta)/max(\gamma)$. However, our experiments showed that this feature is inconsistent among motion signals for the same keystroke. We found that pitch angle and roll angle do not reach their peaks simultaneously. Figure 1 illustrates the paths of the pitch and roll angles as the device vibrates during typing. Each path exhibits a pattern with two lobes, each on one side of the horizontal axis. During a keystroke, the pitch and roll angles move from the center of the pattern to the vertex ($max(\sqrt{\beta^2 + \gamma^2})$) on the upper lobe ($\beta > 0$) through one path, then to the vertex on the lower lobe ($\beta < 0$) via another path, and finally back to the center of the pattern.

We observed that the lobes of the patterns produced by the same key point to similar directions, while the angles of the lobes vary for different keys. Figure 1 shows that the upper lobes of the patterns point up left for keys 1, 4, and 7, straight up for keys 2, 5, and 8, and up right for keys 3, 6, and 9. This observation is consistent with the position of these keys on the soft keyboard (Figure 3). The directions of the lower lobes also demonstrate similar patterns. Therefore, we use lobe direction as the feature for inferring keystrokes.

Each lobe consists of two path segments, one from the horizontal axis up to the vertex, and the other one from the vertex down to the horizontal axis. To measure the direction of a lobe, TouchLogger searches for the *dominating edge* on each path, and the direction of the lobe is the bisector of the angle between the two dominating edges.

For each pattern, we extract two features: the angle between the direction of the upper lobe and the x-axis (AUB, or *Angle of Upper Bisector*), and the angle between the direction of the lower lobe and the x-axis (*ALB*, or *Angle of Lower Bisector*) as shown in Figure 1.

3.4 Classification

We use supervised learning to infer keystrokes from features extracted in Section 3.3.

Training During the training phase, we provide TouchLogger with a data set that consists of motion signals with their corresponding keys. Assuming that the features of the same key have a Gaussian distribution, TouchLogger calculates the mean (μ^k_{AUB}, μ^k_{ALB}) and standard deviation (σ^k_{AUB}, σ^k_{ALB}) for each key k.

Classification During classification, TouchLogger extracts *AUB* and *ALB* from each motion signal and calculates the probabilities that the signal corresponds to each key using the *probability density function* for Gaussian distribution:

$$p^k_{AUB} = \frac{1}{\sigma^k_{AUB}\sqrt{2\pi}} exp(-\frac{(AUB - \mu^k_{AUB})^2}{2\sigma^k_{AUB}{}^2}) \tag{1}$$

$$p^k_{ALB} = \frac{1}{\sigma^k_{ALB}\sqrt{2\pi}} exp(-\frac{(ALB - \mu^k_{ALB})^2}{2\sigma^k_{ALB}{}^2}) \tag{2}$$

$$p^k = p^k_{AUB} \times p^k_{ALB} \tag{3}$$

TouchLogger determines the key as $\arg\max_k p^k$. Our evaluation shows an accuracy rate of 50.6%.

3.5 Advanced feature extraction

Beside lobe direction, we observed that the width of the lobes can also be used to distinguish keystrokes. Therefore we added two more pairs of features to improve the accuracy of keystroke inference.

The first pair of features are the angles of the two dominating edges (*AU* and *AL* in Figure 1). We calculate the means (μ^k_{AU}, μ^k_{AL}) and standard deviations (σ^i_{AU}, σ^k_{AL}) in the training phase. During classification, TouchLogger determines the key as $\arg\max_k P^k$ where

$$P^k = p^k_{AUB} \times p^k_{ALB} \times p^k_{AU} \times p^k_{AL} \tag{4}$$

Our evaluation shows that these features increase the classification accuracy to 64.8%.

The second pair of features are the average width of the upper (and lower) lobe, defined as the area of the polygon formed by the upper (and lower) lobe and the horizontal axis divided by the pitch angle (y-axis value) of the upper (or lower) vertex. Combining all these three

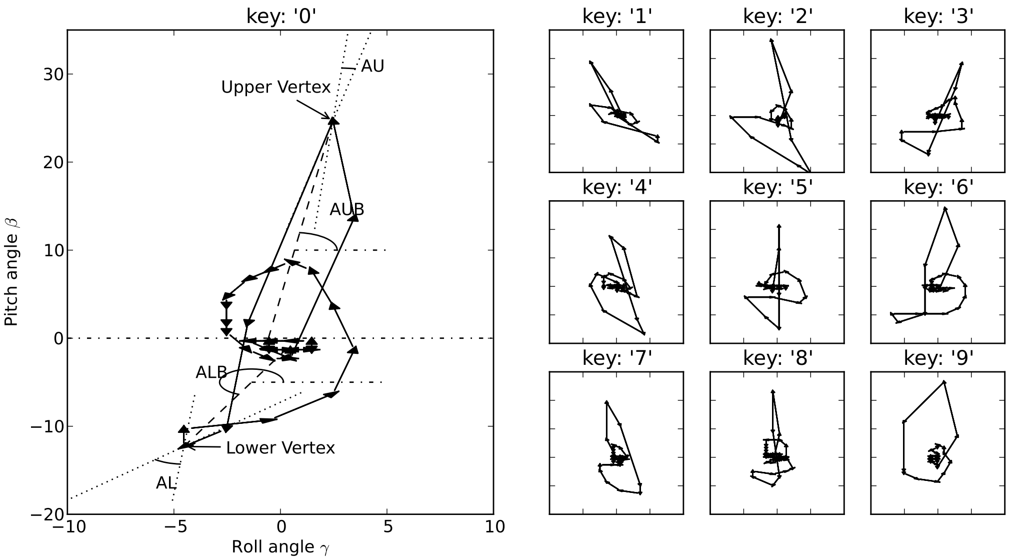

Figure 1: Typical patterns of patch angles and roll angles when different digit keys are pressed. We extract features from these patterns.

pairs of features (two from this section and one from Section 3.3), TouchLogger successfully inferred 71.5% of all keystrokes.

Figure 2 shows the means and standard deviations of all three pairs of features. The x-axis represents features associated with the upper lobe while the y-axis represents features associated with the lower lobe. The boxes represent the range of features in the training data, where the horizontal and vertical centers of each box represent the means of each feature pair while the width and height represent half of the standard deviations. The distances between the boxes reflect the quality of the features. For example, Figure 2 shows that the first pair of features are a good discriminator between keys 1, 2 and 3 or keys 4, 5 and 6, but not between keys 1 and 4 or keys 3 and 9. The other two pairs of features are inferior to the first pair on most keys (because the boxes are much closer to each other), but they complement the first feature pair as they can distinguish keys 1 and 4 or keys 3 and 9 better.

4 Evaluation

We conducted a preliminary evaluation of TouchLogger on an HTC Evo 4G smartphone. Figure 3 shows the user interface of the data collection application. We collected three datasets of keystrokes on a number-only soft keyboard in the landscape mode. Each dataset includes multiple sessions containing from 4 to 25 consecutive keystrokes. The datasets cover all the 16 keys on the soft keyboard, but we used only the digit keys (0 $\cdots$ 9) to train and evaluate TouchLogger.

TouchLogger achieves an accuracy rate of over 70% on each dataset. On the largest data set with 449 strokes of digit keys, TouchLogger correctly inferred 71.5% of them. Table 1 shows the inference results on each key. The keys with the highest inference accuracy are digits 1 and 9, both located on a corner of the soft keyboard. This is consistent with Figure 2, which shows that the feature boxes for keys 1 and 9 are separated further than those for the other keys. Among all the 90 false inference rates (all the rates not on the diagonal in Table 1), 12 of them are larger than or equal to 10%. Out of these 12 worse cases, in 9 cases the inferred key is in the same column as the actual key, and in 7 cases the inferred key is next to the actual key in the same column. This suggests that physical proximity decreases inference accuracy.

Size of training dataset The smaller the required training dataset, the easier it is for the attacker. We examined the convergence of the mean and standard deviation of the features used in classification as the training set size increases. Figure 4 shows that *AUB* and *ALB* for one key converge decently after 5 keystrokes.

5 Discussions

Factors affecting inference accuracy The motion of the smartphone during a keystroke is affected by many factors, such as the typing force, the resistance force of the holding hand, the original orientation of the device, and the location where the supporting hand holds

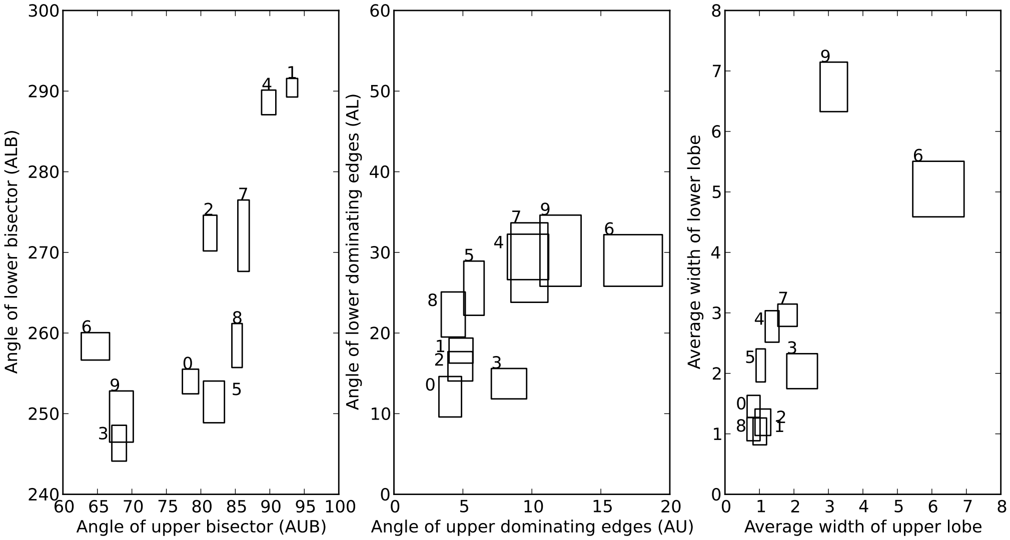

Figure 2: The distribution of three pairs of features extracted from the device orientation data over different digit keys. Each pair has one value for the upper lobe of the pattern and one for the lower lobe. The horizontal and vertical centers of each box represent the means of one feature pair while its width and length represent the standard deviations. The digits around the boxes label their corresponding keys.

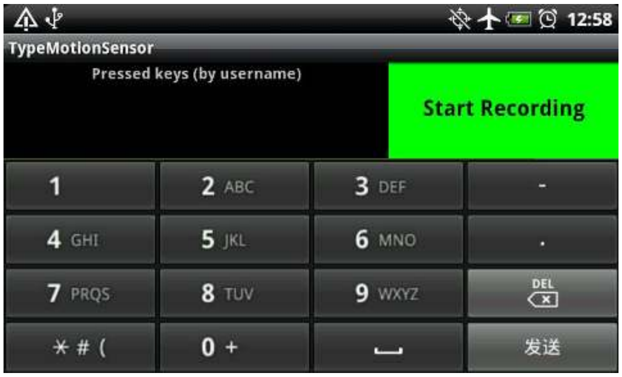

Figure 3: User interface of the data collection application.

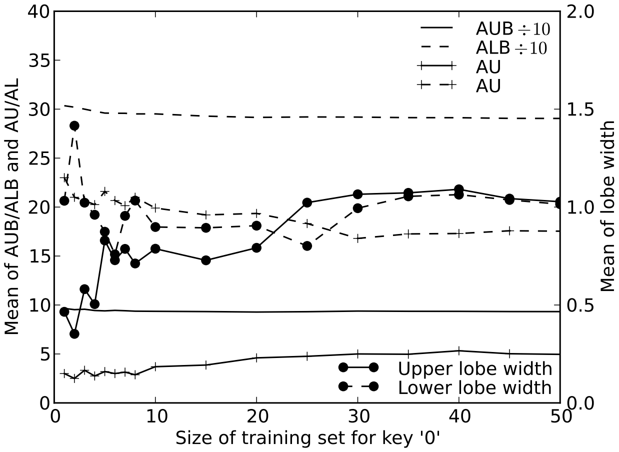

Figure 4: The relationship between the means of signatures and the size of the training set.

the device. Among these factors, only the last one may have significant impact on TouchLogger, because it may change the pivot points of the device. However, our evaluation suggests that a user usually holds his smartphone at the same location. The datasets presented in Section 4 were collected from the same user on different days, when the user held his device in his naturally way each time rather than striving to be consistent.

Application to other devices We believe that TouchLogger can be applied to other devices. Particularly, we expect TouchLogger to perform even better on devices with larger screens, such as tablet computers, as soft keys are separated farther apart there.

Other motion sensors TouchLogger uses data in device orientation events, which are mainly derived from the accelerometer. We could try other sensors that capture motion. Gyroscope, for instance, measures the rate of rotation around the X,Y and Z axes, and its output

Actual Key	Inferred Key									
	0	1	2	3	4	5	6	7	8	9
0	64%	-	6%	10%	-	12%	-	-	8%	-
1	-	86.3%	-	-	13.7%	-	-	-	-	-
2	8.3%	4.2%	68.8%	4.2%	-	2.1%	3.1%	4.2%	6.2%	-
3	18%	-	-	70%	-	-	6%	-	-	6%
4	-	10%	8%	-	72%	2%	-	8%	-	-
5	8%	4%	4%	8%	-	60%	-	4%	12%	-
6	-	-	1.9%	7.5%	-	1.9%	77.4%	-	-	11.3%
7	2%	-	4%	-	16%	14%	-	56%	8%	-
8	-	-	10%	-	-	15%	-	-	75%	-
9	-	-	-	3.8%	-	3.8%	11.5%	-	-	80.8%

Table 1: Distribution of inferred keys. 321 out of 449 keystrokes (71.5%) were correctly inferred.

can be easily converted to device orientation through integral. Camera could also be used to detect motion.

6 Related works

Key logging based on side channels Researchers have studied keystroke inference based on side channels, such as sound [2, 11], electromaganetic wave [8], and timing [7, 4]. Since these attacks exploit characteristics of physical keyboards, they become ineffective on smartphones with soft keyboards.

Attacks using sensors on smartphone [3] raises the awareness of privacy attacks on smartphone sensors. Besides the obvious privacy concern over the GPS sensor, researchers have shown attacks using the camera [9] and microphone [10]. To the best of our knowledge, this paper is the first to show the privacy risks of motion sensors.

7 Conclusion

We investigated the use of motion as a side channel to infer keystrokes on soft keyboard on smartphones. We observed that, due to the locations of keys on a soft keyboard, typing different keys causes different motions of the smartphone. We developed TouchLogger, a smartphone application that extracts features from the device orientation data to infer the keys being typed. Our evaluation shows that TouchLogger can correctly infer more than 70% of the keystrokes on a number-only keyboard in the landscape mode. We have demonstrated that motion is a significant side channel, which may leak confidential information on smartphones.

8 Acknowledgments

This work was partially supported by the National Science Foundation through grants CNS 0644450 and 1018964. We also thank Xinwen Zhang for his support.

References

[1] Android developer's documentation: Sensorevent. `http:/developer.android.com/reference/android/hardware/SensorEvent.html`.

[2] Asonov, D., and Agrawal, R. Keyboard acoustic emanations. In *IEEE Symposium on Security and Privacy* (2004).

[3] Cai, L., Machiraju, S., and Chen, H. Defending against sensor-sniffing attacks on mobile phones. In *Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds* (New York, NY, USA, 2009), MobiHeld '09, ACM, pp. 31–36.

[4] Foo Kune, D., and Kim, Y. Timing attacks on pin input devices. In *Proceedings of the 17th ACM conference on Computer and communications security* (New York, NY, USA, 2010), CCS '10, ACM, pp. 678–680.

[5] Pachal, P. Google removes 21 malware apps from android market. Accessed March 18, 2011. http://www.pcmag.com/article2/0,2817,2381252,00.asp.

[6] Popescu, A., and Block, S. DeviceOrientation event specification, editor's draft 9. `http://dev.w3.org/geo/api/spec-source-orientation.html`, February 2011.

[7] Song, D. X., Wagner, D., and Tian, X. Timing analysis of keystrokes and timing attacks on ssh. In *Proceedings of the 10th conference on USENIX Security Symposium - Volume 10* (Berkeley, CA, USA, 2001), USENIX Association, pp. 25–25.

[8] Vuagnoux, M., and Pasini, S. Compromising electromagnetic emanations of wired and wireless keyboards. In *Proceedings of the 18th conference on USENIX security symposium* (Berkeley, CA, USA, 2009), SSYM'09, USENIX Association, pp. 1–16.

[9] Xu, N., Zhang, F., Luo, Y., Jia, W., Xuan, D., and Teng, J. Stealthy video capturer: a new video-based spyware in 3G smartphones. In *Proceedings of the second ACM conference on Wireless network security* (New York, NY, USA, 2009), WiSec '09, ACM, pp. 69–78.

[10] Zhang, K., Zhou, X., Intwala, M., Kapadia, A., and Wang, X. Soundminer: A stealthy and context-aware sound trojan for smartphones. In *Proceedings of the 18th Annual Networkand Distributed System Security Symposium* (2011), NDSS '11.

[11] Zhuang, L., Zhou, F., and Tygar, J. D. Keyboard acoustic emanations revisited. *ACM Trans. Inf. Syst. Secur. 13* (November 2009), 3:1–3:26.