# EavesDroid: Keystroke recovery using mobile phone accelerometers

Jennifer Guo
Princeton University

Yi-Hsien Lin
Princeton University

Akshay Mittal
Princeton University

Wathsala Vithanage
Princeton University

***Abstract*—[to - fill]**

## I. Introduction

Mobile phones are becoming increasingly powerful devices. In addition to being able to run simple applications such as email clients and web browsers; by equipping a wide variety of sophisticated sensors, these smartphones can actively interface with the world aound them.

Unfortunately, the array of sensors in these devices can also be used in malicious ways. For example, malware could potentially gain access to a mobile phone's camera to take photos or video without the notice of the owner [**?**]. Other possible attempts are activating the device's microphone to record converstaions or turn on the GPS to track the target's position [**?**], [**?**], [**?**], [**?**]. In light of the possible privacy threats mentioned above, operating systems that are implemented in modern mobile phones usually provide mechanisms so that applications must acquire the user's explicit permission before it can access the device's sensors.

For instance, an Android application would have a manifest file that includes a list of all permissions an application can request during installation time. Under this design, the application is either granted a particular permission when installed, and can use that feature as desired, or the permission is not granted and any attempt to use the feature fails without prompting the user. However, not all sensors' accesses are tightly regulated by this mechanism, which leads to several potential sensors abuse.

One such example is the accelerometer. Access to acceleromters are usually un-restricted on most of the current mobile phone operating systems. In this paper, we developed **EavesDroid**, which is a malicious application with access to the accelerometer's data. We demonstrate that **EavesDroid** is able to record and reconstruct the keypresses made on a nearby keyboard with decent accuracy, based solely on the observed vibrations. We develop profiles for keypress by extracing features from the keypress signals and classifies them using boosted Random Forests. This allows us to establish an abstract representation of the relationship between a keystroke signal and its corresponding key. We then recover the typed content by translating from our intermediary form to English words using a number of different dictionaries. As mentioned in [**?**], reconstructing keystrokes from vibration signals is very different and much more difficult from previous efforts in this space since they are mostly based on acoustic or electromagnetic eavesdropping and enjoys a higher sampling rate.

Bt developing **EavesDroid** we have made the following contributions to the smarphone security domain:

1) **Develop an infrastructure for characterizing keypress vibrations**: We capture, analyze and develop profiles of keypress events on a nearby keyboard based on the vibrations created when they are pressed. Features such as mean, skewness and FFT are first extracted from our input signals. We then use a set boosted Random Forests to create an intermediary representation, which is combined with candidate dictionaries to successfully recover words at rates as high as

—

2) **Dataset made publicly available**: To the best of our knowledge, there does not exists a dataset with the keystrokes for the vibrations recorded using an accelerometer. In this paper, we demonstrate we construct the dataset of keystroke vibrations, recorded using the accelerometer, present in a smartphone and make the dataset publicly available to enhance further research on this topic. Given the widespread about security breaches and email leaks, we perceive this to be of great importance in motivating the mobile operating

system manufacturers to restrict the usage of accelerometers to permitted applications only.

Note that our approach is greatly influenced by the work of Marquardt *et. al* [**?**] and we present key differences with our approach.

## II. Related Work

Researchers have studied malicious code on mobile devices that learn information about the device owner using other embedded sensors. In the most closely related work to our paper, Marquardt *et. al* [**?**] propose *(sp)iPhone* which uses motion sensors in the iPhone 4 to infer keystrokes of nearby laptop users. *(sp)iPhone* uses 3 labels for each pair of keystrokes in order to determine the correct identification for the letters of the word. The location of the key is modelled using a neural network trained on 150 keystrokes for each letter of the English alphabet. A profile is then constructed for each pair using the neural network and determines and models the distance between the two events. The predictions of the neural network are matched against a dictionary to determine the top matches. The accuracy achieved is roughly 80% but this decreases to 40% as the size of the dictionary increases.

In a very similar, but orthogonal, use case, Owusu *et. al* [**?**] present the threat of applications which extract the keystrokes of the user's typing on the smartphone screen itself. Again, due to no restrictions on the usage of the accelerometer, they are able to extract 6-character passwords in as few as 4.5 trials(median). TouchLogger [**?**] uses orientation of the smartphone device to infer the keystrokes.

## III. Threat Model

Our attack is based upon two observations:

Many users like to place their smartphone close to their laptop while working, so that it is easily accessible and notifications are not missed.

Furthermore we note that while smartphones offer a broad range of sensors such as microphone, GPS, gyroscope and accelerometer, the accelerometer is the only sensor to which access is not protected in any current mobile phone operating system. That is mainly due to the frequent usage needs of the accelerometer to determine window orientation and update screen orientation, a feature any app can make use of.

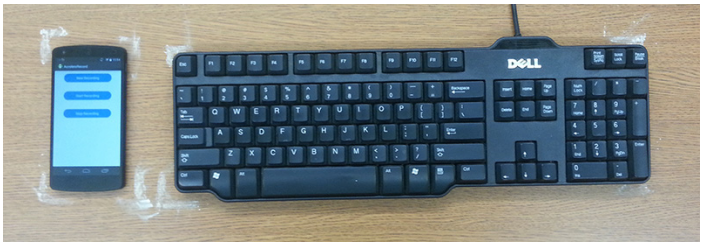


Fig. 1. Threat model of a smartphone placed next to a keyboard. Used in our experimental setup to capture the dataset.

## IV. Framework Description

### A. Dataset Capturing

Our experimental setup is shown in fig.1. We placed a Dell USB keyboard and a LG Nexus smartphone on a wooden table. No other items were placed on or were touching the table. All keys were pressed with the index finger of the right hand. The hands do not touch the keyboard or table, and the fingers do not pause on the keyboard but only touch it during the brief duration of the key press.

We recorded 1000 datapoints in 40 sessions with 25 letters in each sessions. For each session we generate a sequence of 25 random letters and display the letters in 3 second intervals to ensure an even recording of the letters and to ensure that there is no bias towards a specific letter. For the data capturing we developed an Android app that reads from the hardware accelerometer and syncs with our laptops, where we further process the data. The raw accelerometer data as returned by Android contains the $x$, $y$ and $z$-direction components of the acceleration of the vibration received. We calculate the G-force magnitude as $\sqrt{x^2 + y^2 + z^2} - g$ where $g = 9.81$ is the acceleration due to gravity.

The original recording will contain a large spike at the beginning and end due to the pressing of "Start" and "Stop" on the recording app. We cut off these noise signals and further segment the data into 25 individual data points. An example of a plot of one such signal is shown in fig.2.

### B. Feature Extraction

We use a combination of time-domain and frequency-domain features to construct our feature vector.

## V. Implementation

[table with all combinations of accuracies]

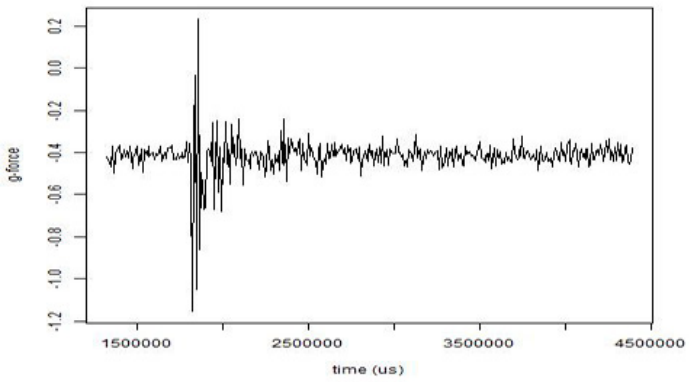


Fig. 2. Sample Recording for letter 'a'

LR predictor and Up down predictor .

Adaboost with Random Forest

Mention triads did not give too good accuracies

## VI. Experimental Results

[Box sentence: exact matches and possible choices for non-exact matches]

[Negative analysis, word watch, many potential matches when dictionary size increases]

## VII. Conclusions

### A. Future Work

Mobile browsers having allowing access to accelerometer data,

http://blogs.adobe.com/cantrell/archives/2012/03/accessing-the-accelerometer-and-gyroscope-in-javascript.html

http://isthisanearthquake.com/

http://isthisanearthquake.com/seismograph.html

## VIII. Acknowledgments

## Appendix

### A. Appendix Subsection