WristSnoop: Smartphone PINs Prediction Using Smartwatch Motion Sensors

Allen Sarkisyan
Applied Mathematics
Cal State Northridge (CSUN)
Northridge, USA
Allen.sarkisyan.860@my.csun.edu

Ryan Debbiny
Computer Science
Cal State Northridge (CSUN)
Northridge, USA
ryan.debbiny.575@my.csun.edu

Ani Nahapetian
Computer Science
Cal State Northridge (CSUN)
Northridge, USA
ani@csun.edu

Abstract— **Smartwatches, with motion sensors, are becoming a common utility for users. With the increasing popularity of practical wearable computers, and in particular smartwatches, the security risks linked with sensors on board these devices have yet to be fully explored. Recent research literature has demonstrated the capability of using a smartphone's own accelerometer and gyroscope to infer tap locations; this paper expands on this work to demonstrate a method for inferring smartphone PINs through the analysis of smartwatch motion sensors. This study determines the feasibility and accuracy of inferring user keystrokes on a smartphone through a smartwatch worn by the user. Specifically, we show that with malware accessing only the smartwatch's motion sensors, it is possible to recognize user activity and specific numeric keypad entries. In a controlled scenario, we achieve results no less than 41% and up to 92% accurate for PIN prediction within 5 guesses.**

Keywords—Wearable computing; mobile security; PIN prediction; keystroke inference; mobile malware; smartwatches.

I. INTRODUCTION

Smartwatches are becoming popular items in the wearable technologies market, and thus opening new channels for malware attacks. In this paper we examine the feasibility of extracting personal identification numbers (PINs) entered on a smart phone, using only motion sensor information captured, via a malicious app, from a wrist-worn smart watch. PINs are commonly used for authentication purposes on smart phones, including for unlocking smart phones, for accessing voicemail, and for access banking accounts using ATM PINs, thus making them attractive targets for malicious agents. They involve the user tapping one of 10 digits (0-9) in the correct multiple digit sequence. By random entry, a malicious agent would have a 1 in 10,000 chance to guess a four-digit PIN.

Smartphones have the capability to log motion, enabling the possibility reliable predictions of user inputs and actions. The motion sensors often provide acceleration and rotation information which is plentiful for inferring the smartphone four-digit PIN and pattern passwords based on a few seconds of data collection [1]. With the recent addition of the smartwatch to the user's list of personal devices, more information can be gained and utilized than ever before for motion sensor side-channel attacks. This paper attempts to demonstrate a method for inferring smartphone PINs via the motion sensor logs of a wrist-worn vulnerable smartwatch.

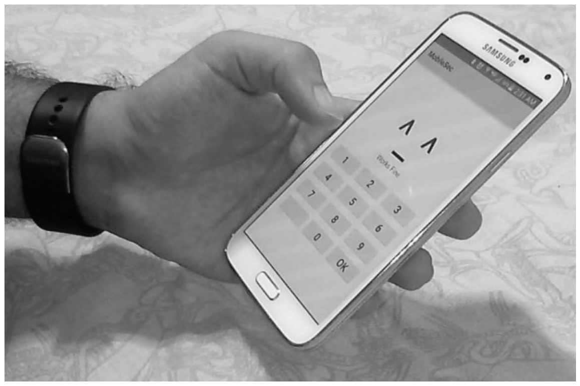

Fig. 1: A demonstration of PIN entry with a smartphone while wearing the smartwatch.

In our experiments, a user wore a smartwatch on the left hand, held a smartphone in the left hand, and entered PINs with his thumb using the same hand. A demonstration of the method can be seen in Figure 1. A sequence of digits was tapped on the smartphone, simulating a typical PIN entry by the user. Motion sensor readings of the smartwatch were collected alongside the digits being pressed from the smartphone, and the timestamps for each data point to match the two datasets into sensor-label pairs. A training set of data was aggregated through this process to model the motion the user employs to tap specific numeric PIN labels on the smartwatch for the purpose of inferring them in a test set.

Random forest classification was applied to predict PINs from unlabeled data points in 21 data sets from a single user across multiple sessions, simulating a side-channel PIN attack. For each set of four-digit taps, the model was given 5 tries at predicting the PIN correctly before moving onto another PIN. After several hundred trials, the results were gathered to examine the performance of the model. After applying the Savitzky-Golay filter for data smoothing, the accuracy for PIN prediction varied across the different datasets allocated, with prediction accuracy reaching 92% in the best dataset and 41% in the worst dataset.

II. Attack Model

This paper examines the attack scenario where malware accessing a smartwatch is used to determine a PIN entered on the user's smartphone. The attack assumes a user is wearing a smartwatch paired with a smartphone, where PINs are entered via touch screen taps.

Malicious software installed on the devices, as a smartwatch app, will monitor the smartwatch's motion sensors and transmit (with some possible filtering) the data to the smartphone, which will then relay the information via a mobile data or WiFi connection to a remote site for data processing and PIN cracking.

Some assumptions regarding the attack include the following:

- The user enters PIN information using the same hand that carries the smartwatch. This attack method could be complimented with an existing method of smartphone PIN prediction such as [1] [2] [3] and [4] for boosting classification accuracy and reliability.
- Either the victim's smartphone or smartwatch is able to send data (< 100KB in total size) via the internet to the attacker servers for analysis.
- The malicious software will be installed onto the victim's smartwatch, perhaps disguised as a harmless smartwatch application.

  For Android devices, the use of the accelerometer and gyroscope does not require permission from the user [5].
- The attacker can determine the approximate time interval at which the user is entering a PIN onto the smartwatch. This event can be detected utilizing a classification algorithm to approximate the time interval for the handling of the phone during PIN entry.

  In conducting this analysis, a random forest classifier is used to infer the tapped PINs based off of the unlabeled sensor data collected from the smartwatch during a certain interval of time where the user has tapped the screen for PIN entry.

III. Related Work

Previous work has examined the leakage of sensitive data via side-channel attacks on mobile devices that access the GPS and other location sensors [6], the camera [7], bioimpedance differences between people [8], and the movement of the phone from taps [2]. More recently, researchers have looked at using the pervasiveness of the mobile device to determine environmental information, notably using smartphone's microphone [9] and accelerometer [10].

Specifically, in terms of using motion sensors on the smartphone, TouchLogger achieved 70% accuracy in inference of the number pad on a smartphone using motion and orientation side channel attacks [3]. ACCessory was able to crack six character passwords using only the accelerometer of the phone [4]. Trojan software designed by the authors of TapLogger was able to classify a user's activity, often with more than 90% accuracy, on the number pad of a smartphone using the onboard accelerometer and gyroscope [2]. A more recent study [11] was conducted to demonstrate the feasibility of capturing keystrokes using smartwatch motion.

To the best of our knowledge this is the first work looking at smartphone PIN detection using user-worn smartwatches. Wrist-mounted sensors and/or smartwatch gesture classification has been used previously in other applications, such as for smoking gesture detection [12] and gestured alphabetic character classification [13].

IV. Data Collection

Approximately 100,000 sensor records were collected with a single user for a 2-3 minute session across 21 sessions. The user wore the smartwatch on the left hand while holding the smartphone in the left hand while sitting down. The left thumb was used for the typing on the smartphone numeric pin pad. The smartphone application used in the testing shown in Figure 2 simulates a typical Android smartphone PIN layout. A Samsung Galaxy S5 smartphone paired alongside a Samsung Galaxy Gear Live smartwatch were used in all the data collection.

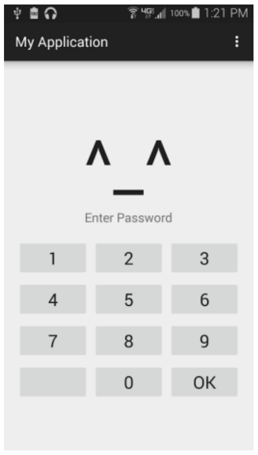

Fig. 2: A screen shot of the Android application PIN entry screen used for the data collection.

To avoid iterating through all 10,000 possible combinations with a 4-digit PIN, the following number sequences were tapped to cover all possible PIN pairs: (1,1), (1,2), (1,3), … , (1,8), (1,9), (1,0), (2,1), (2,2), … , (2,9), (2,0), (3,1), … , (3,0), … , (0,9), (0,0). Each tuple represents two taps in order, and each tuple is followed by the previous tuple sequence. For example, the first 6 labels tapped are 1, 1, 1, 2, 1, 3.

This method ensures that all label taps are gathered with all possible following and prior labels. The sensors are thus gathering the change in motion for all possible PIN combinations with 100 unique tuples. The result of one collection of the full set of tuples is about 200 taps (with minimal human error missing a few taps and including some extra taps).

A total of 21 datasets were gathered with a few including up to 3 iterations of the cycle of tuples and with the majority with 1 cycle.

The data gathered directly from the smartwatch included the x,y, and z-axes of the accelerometer (without gravity), x,y, and z-axes of the gyroscope, and x,y, and z-axes of the rotation vector. Both the smartwatch and smartphone were programmed to collect timestamp data: the smartphone to collect the times of pressing and releasing labels, and the smartwatch for each recorded sensor reading.

The smartwatch sensor data was transmitted to the smartphone and written into CSV files at the touch of the "OK" button. The attack model thus assumes that the malicious software can detect the approximate beginning and end of the PIN entry to cut off extra noise. The sensors were sampled at rate of 0.5KHz, with samples collected at approximately 20-30 records on average per tap, or in other words 80-120 records in a 4-tap interval.

V. Feature Selection

The method for collecting data to train the classifier involved collecting sensor data from the smartwatch and the label data from the smartphone. The sets of data were merged in such a way that the labels of the smartphone were applied to the records of the smartwatch by matching the timestamp interval of the user's taps. The labeling also included a null state denoted as -1 to indicate sensor readings during periods without taps i.e. the time interval between one tap and another.

TABLE I: Features explored for PIN value classification

Ranking	Features	Description
-	Watch Time	Unix Timestamp
1	Accelerometer xyz	Linear Acceleration Excluding Gravity
2	Gyroscope xyz	Rotation around the xyz axes
3	Rotation Vector xyz	Rotation vector component along the xyz axis e.g. ($x\times$ sin(θ/2))
4	Magnitude of Accelerometer Vector	Magnitude of xyz components
5	Magnitude of Gyroscope Vector	Magnitude of xyz components
6	Magnitude of Rotation Vector	Magnitude of xyz components
7	Change (denoted with delta or a prefix 'd') in each of the above motion sensors (12 features)	Change in value from prior point in time, divided by the length of time.
8	Product of Rotation and Accelerometer Magnitudes	Rotation Mag. $\times$ Accelerometer Mag.
9	Product of Rotation and Gyroscope Magnitudes	Rotation Mag. $\times$ Gyroscope Mag.
10	Product of Accelerometer and Gyroscope Magnitudes	Accelerometer Mag. $\times$ Gyroscope Mag.
-	Label	The Class Label -1 for null 0-9 for PIN labels

As shown in Figure 3, using random forest tree feature selection, the most important features in the dataset for predicting the class labels were the original sensor values. The feature sets were tested separately to compare the results of the full feature set versus the subset of features chosen by the random forest selection. Table II shows the performance of the models for comparison.

Further features were created from the original dataset to include the changes in values across each axis of each sensor reading, the magnitude of each sensor reading, and the product of each possible pair of magnitudes. The final column space included the features listed in Table I.

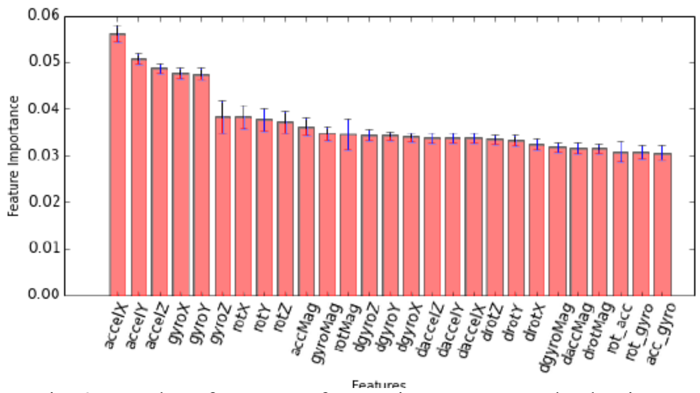

Fig. 3: Random forest tree feature importance and selection

VI. Approach

A. Model Selection

Although several models were examined, the random forest classifier proved to have the best performance across the datasets for accuracy, precision, and recall. The classifier was trained with 150 trees using entropy as the criterion for information gain, and was tested with 5-fold cross validation to ensure against overfitting.

It was shown, as in the tree selection Table II, that the best feature set excluded the delta and product creations. The final model thus excluded those features in PIN predictions.

The model was further improved by applying the Savitzky-Golay filter to the dataset for noise reduction. The classifier was trained on both the original sensor data and the filtered data to examine the differences in performance. The model's precision score as shown in Table III was improved after filtering, and so the method of smoothing was adopted for PIN prediction.

Table II. Random forest classification model performance using 150 trees for both filtered and non-filtered datasets, and for full-features and selected-features. Tested with 5-fold cross-validation.

			Accuracy	Precision	Recall	F1 Score
No Filter	Full Feature	Avg	0.66	0.57	0.66	0.54
		Std Dev	0.00	0.01	0.00	0.00
	Without Delta, Products	Avg	0.67	0.60	0.67	0.56
		Std Dev	0.00	0.01	0.00	0.00
Savitzky-Golay Filter	Full Feature	Avg	0.68	0.70	0.68	0.57
		Std Dev	0.00	0.01	0.00	0.00
	Without Delta, Products	Avg	0.70	0.72	0.70	0.62
		Std Dev	0.00	0.00	0.00	0.00

Table III. PIN prediction results from running 100 trials at 5 times for each dataset, without delta or product features. Tested with 5-fold cross-validation.

	Dataset	1	2	3	4	5	6	7	8	9	10	11
No Filter	Avg	0.76	0.84	0.83	0.92	0.73	0.91	0.69	0.71	0.95	0.90	0.33
	Std Dev	0.03	0.02	0.04	0.02	0.03	0.02	0.04	0.06	0.02	0.03	0.07
	Dataset	12	13	14	15	16	17	18	19	20	21	
	Avg	0.42	0.36	0.50	0.66	0.67	0.72	0.53	0.59	0.71	0.91	
	Std Dev	0.04	0.03	0.02	0.04	0.02	0.05	0.04	0.03	0.06	0.03	
Savitzky-Golay	Dataset	1	2	3	4	5	6	7	8	9	10	11
	Avg	0.75	0.84	0.84	0.92	0.81	0.89	0.73	0.70	0.89	0.86	0.42
	Std Dev	0.05	0.02	0.04	0.05	0.03	0.02	0.05	0.05	0.01	0.02	0.03
	Dataset	12	13	14	15	16	17	18	19	20	21	
	Avg	0.41	0.44	0.48	0.60	0.62	0.78	0.55	0.56	0.75	0.87	
	Std Dev	0.06	0.03	0.06	0.03	0.03	0.06	0.04	0.05	0.05	0.03	

B. Pin Prediction

Due to the large class imbalance between the active states (labels 0-9) and the null state (non-tap event), classification of the states often results in many more predictions of the null state than any other active state. This outcome was utilized as an advantage for the PIN prediction process. By predicting labels for each sensor reading (as opposed to a sequence), we discovered a redundant logic in the classification that boosted prediction accuracy. By randomly selecting an interval of 4 taps (a typical PIN) from the dataset, we were able to make predictions on each sensor reading. The result appeared as a list of about 120 class label predictions (about 30 data points during the interval of any single tap), and as expected there were many '-1' (null state) predictions. After removing the null state predictions, what was left was often a list of several repeating class label predictions.

An example output after the removal of null state predictions is as follows:

(1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 3, 3, 3, 3, 3, 1, 3, 3, 3, 3, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 2, 6, 6, 6, 6)

By examining the mode of the quartiles, we can accurately predict the PIN for a 4-tap interval, even in the presence of errors. For example, the above list would have an output of (1, 3, 5, 6) as the predicted PIN. This method works well because misclassifications often result in null states, as opposed to active states.

C. Null-Active State Prediction

An exploration was made into the question of whether the active (labels 0-9) and null (-1) states could be clustered, and whether such clustering would result in or provide an accurate method for predicting future labels on this binary measure.

Using k-means clustering on a train-test split of the dataset (test size of 20% holdout), it was found to be the worst measure among several classifiers, alongside logistic regression at about a 50-50 chance of correctly classifying null or active state labels.

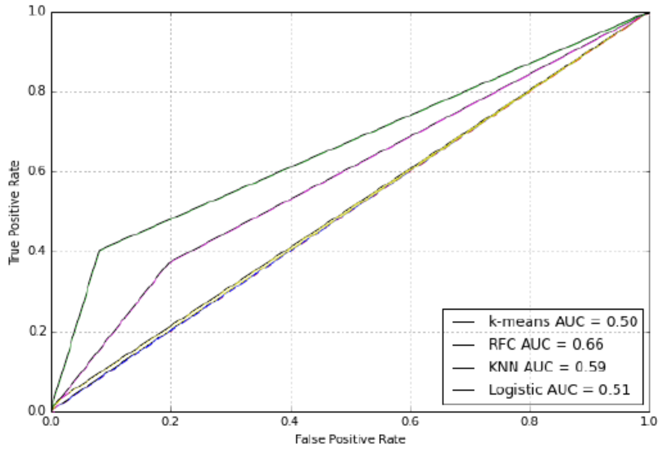

Fig. 4: Original sensor data binary classification ROC plot for null-active state prediction.

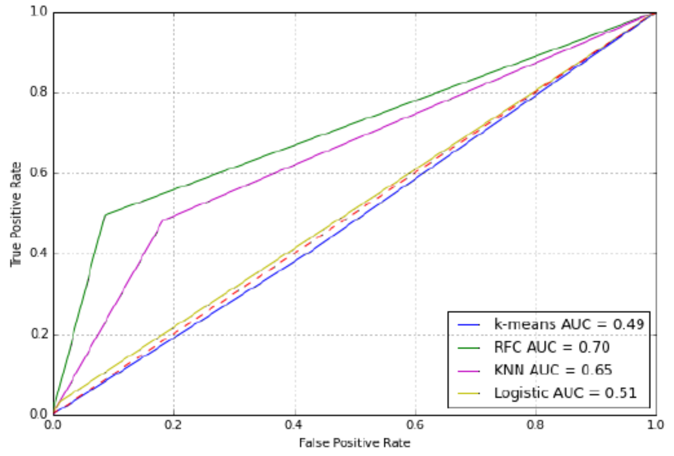

Fig. 5: Savitzky-Golay filtered binary classification ROC plot for null-active state prediction.

The models were run on both the original dataset and for the filtered one to inspect the differences. As shown in Figures 4 and 5, the receiver operating curve (ROC) plots show the accuracy of each model as the area under the curve (AUC) score, and their corresponding true positive and false positive rates. The random forest classifier (150 trees using entropy as the criterion for information gain) performed the best, followed by the k-nearest neighbors classifier (5 neighbors). Although the k-nearest neighbors classifier did not perform well for PIN prediction, it is useful for such binary classifications.

The final model did not rely on the use of the binary classification as the probability for correct classification through two models of first classifying null-active and then classifying within the active states was lower than simply using a single classifier on the full class-selection range. The final model was chosen as the random forest classifier, trained to classify across all states.

VII. Results

Using the random forest model as a form of feature selection (see Figure 3), we found that the acceleration in the x-axis of the smartwatch proved to be the most relevant feature for predicting class labels. It further followed that the other accelerometer axes were all more valuable than the gyroscope sensor readings, which then followed the rotation vector. The creation of further features was not very helpful with boosting the performance of the model as seen in Table II.

The model was also run with the first two principal components using principal component analysis (PCA), it however performed worse than the original feature set. It is worth noting that the random forest model demonstrated that the PCA features of the least importance among the full feature set, and thus was left out of the analysis. This paper relied on using PCA plots for visually representing the distinction between active and null states of the data, to find a way for better separation of the data points. The plots confirm the difficulty in separating the classes, but it also points to the difficulty in creating a more uniform dataset. Due to the bias gained from training from only one user's wrist, the PCA plots show sets of clusters forming. In plots containing fewer concatenations of datasets, the clusters become more visible.

The variance explained ratio given by the principal components (see Fig. 6) is expected, and typical for PCA. The majority of the variance (about 65%) can be explained with the first two principal components, so what we see in the PCA plot is the best visualization for separation. The resulting PIN predictions (Table III) shown for a single user's wrist are optimistic as they overfit one user, and it is expected that the PCA plot for multiple users will contain fewer visible cluster separations and a more blurred separation between active and null states.

The model was trained on the full dataset using a train-test split with a testing size of 20% using the random forest classifier (150 trees, entropy criterion). It was tested on each dataset individually by predicting 5 sets of 100 PINs.

For each prediction, the model makes at most 5 unique guesses for the PIN to simulate a realistic password attack. The reasoning for this is that most devices are configured to lock the PIN entry screen at 5 incorrect guesses.

The different guesses are created by examining the threshold for probabilities of class labels by the random forest classifier model. For each prediction, the model examines the probabilities for class label assignment across each label and appropriates the one with the highest value.

For further guesses, the next-best probabilities are used to substitute the best predicted values. This way, if for one prediction, two class labels are differing by a small probability in the model, that prediction will be the one to swap to the next-best guess of a class label. This method preserves the confident guesses and swaps out the questionable ones, often resulting in success within 4 guesses. This process is taken in steps of incremental 10% threshold limit values until a unique new guess is created and tried. If the model fails to predict the PIN value correctly, it is returned as a 0 (otherwise a 1 for success).

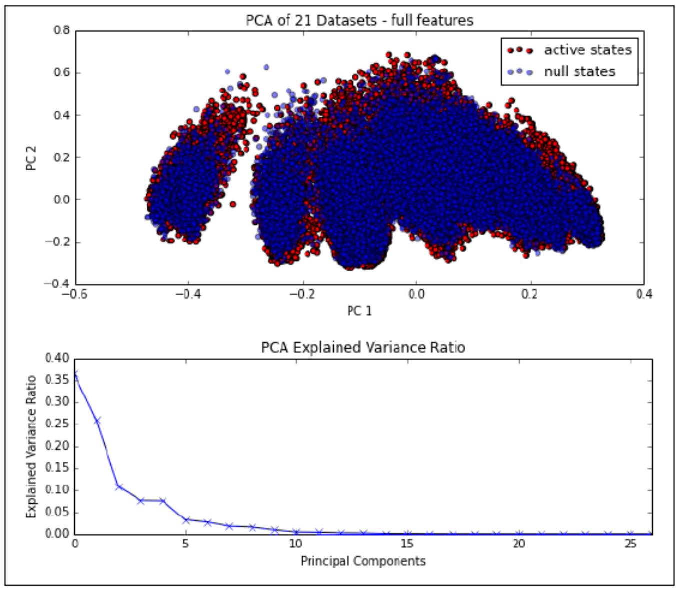

Fig. 6. PCA plot and PCA explained variance ratio for full feature set

The results in Table III show the averages and standard deviation for these 5 sets, for both the original dataset and the filtered dataset. The scores represent percentage accuracy measures, and are a good reflection of the performance since many scores have low variance across the 5 tests. If for example the score is 0.90, then the model was able to on average predict 90 out of 100 PINs.

For the majority of the datasets, the two models (unfiltered versus filtered) performed with similar accuracy given the range of uncertainty for each set. The interesting distinction of the Savitzky-Golay filtered model is that it tended to have greater accuracy for two datasets, 5 and 13, beyond the range of the standard deviation. It is a minor note, but reflective of the boosted precision of the cross-validated scores in Table II.

The model was also tested on unseen data, with results showing poor classification. Due to the wide range of initial starting positions for the user, and the inherent differences in wrist motion from one session to another, the random forest classifier was not able to classify the data points correctly. This is likely due to the classifier not being trained on a particular motion before, which is very likely given that the user was only tested across 21 sessions. It is expected that in an extended study with a diverse user group and several million data points, the classifier would fare far better with unseen data classification.

VIII. Conclusion

In this paper, the feasibility of smartwatch motion sensor attacks to extract PINs entered on smartphones was demonstrated. We looked at the scenario where the user is wearing a smartwatch and holding a smartphone to enter sensitive numeric data. In a controlled scenario, at least 41% accuracy is achieved with the smartwatches' motion sensors for PIN prediction within 5 guesses.

REFERENCES

[1] A. J. Aviv, B. Sapp, M. Blaze, and J. M. Smith, “Practicality of accelerometer side channels on smartphones,” in Proceedings of the 28th Annual Computer Security Applications Conference. ACM, 2012, pp. 41–50.

[2] Z. Xu, K. Bai, and S. Zhu, “Taplogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors,” in Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks. ACM, 2012, pp. 113–124.

[3] L. Cai and H. Chen, “Touchlogger: Inferring keystrokes on touch screen from smartphone motion.” in HotSec, 2011.

[4] E. Owusu, J. Han, S. Das, A. Perrig, and J. Zhang, “Accessory: password inference using accelerometers on smartphones,” in Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications. ACM, 2012, p. 9.

[5] “Android Developer API Guides security permissions,” http://developer.android.com/guide/topics/security/permissions.html, accessed: 2015-06-25.

[6] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, “Taintdroid: an information flow tracking system for realtime privacy monitoring on smartphones,”ACM Transactions on Computer Systems (TOCS), vol. 32, no. 2, p. 5, 2014.

[7] M. Backes, M. Durmuth, and D. Unruh, “Compromising reflections -or - how to read lcd monitors around the corner,” in Security and Privacy, 2008. SP 2008. IEEE Symposium on. IEEE, 2008, pp. 158–169.

[8] C. Cornelius, J. Sorber, R. Peterson, J. Skinner, R. Halter, and D. Kotz, “Who wears me? bioimpedance as a passive biometric,” in Proc. 3rd USENIX Workshop on Health Security and Privacy, 2012.

[9] M. Backes, M. D¨urmuth, S. Gerling, M. Pinkal, and C. Sporleder, “Acoustic side-channel attacks on printers.” in USENIX Security Symposium, 2010, pp. 307–322.

[10] P. Marquardt, A. Verma, H. Carter, and P. Traynor, “(sp) iphone: decoding vibrations from nearby keyboards using mobile phone accelerometers,”in Proceedings of the 18th ACM conference on Computer and communications security. ACM, 2011, pp. 551–562.

[11] H. Wang, T. Tsung-Te Lai, and R. R. Choudhury, “Mole: Motion leaks through smartwatch sensors,” in Proceedings of the 21st Annual International Conference on Mobile Computing and Networking. ACM, 2015, pp. 155–166.

[12] A. Parate, M.-C. Chiu, C. Chadowitz, D. Ganesan, and E. Kalogerakis, “Risq: Recognizing smoking gestures with inertial sensors on a wristband,” in Proceedings of the 12th annual international conference on Mobile systems, applications, and services. ACM, 2014, pp. 149–161.

[13] D. Moazen, S. Sajjadi, A. Nahapetian. “AirDraw: Leveraging Smart Watch Motion Sensors for Mobile Human Computer Interactions,” in Proceedings of the 12th annual internationl Consumer Communications and Networking Conference (CCNC), IEEE, 2016.