Detecting Phone Theft Using Machine Learning

ABSTRACT

Millions of smartphones are stolen in the United States every year, exposing victims' personal information at risk since many users often do not lock their phones. To protect individuals' smartphones and the private data stored on them, we develop a system that automatically detects pickpocket, and grab-and-run theft, where a thief grabs the phone from a victim's hand then runs away. We collect a dataset which consists of samples from simulated theft experiments and a user study about smartphone usage in subjects' everyday lives. We then build three binary classifiers to classify a data point as theft or normal usage based on features extracted from smartphones' accelerometer data. Among the three models, logistic regression detects 100% of theft instances at a cost of 1 false alarm per week. The phone theft detecting technology presented in this paper also has broad applications. For example, we can use the detector as a protection for an authentication scheme that uses a smartphone, and it triggers an alarm and an automatic screen lock after the phone is stolen.

1. INTRODUCTION

According to the Consumer Reports, 2.1 million smartphones were stolen in the United States in 2014 [3]. The Pew Research Center's Internet & American Life Project reported in 2012 that nearly one third of mobile phone users have experienced a lost or stolen of their devices [1]. [more stats here if found] Checking email and using payment services on their smartphones are in many users' daily routine. Those services require user to store private data, such as credit card information on their devices. The private information stored on the smart devices often worth more than the hardware, which is jeopardized by smartphone theft. However, many smartphones only offer owners PIN authentication, and consumers often turn this feature off to avoid the complication of having to type in PIN every time they want to unlock the screen. Egelman et al. indicates that 42% of users do not lock their smartphones, which allows thieves to easily gain access to the victims' personal information [6].

Conference year, month dates, city, state.

To protect individuals' smartphones and the private data stored on them, we use a machine learning technique called supervised learning to develop a system that automatically detects pickpocket and grab-and-run smartphone theft. To be more specific, we train a binary classifier to distiguish between theft and normal usage of smartphones. When deployed, the detector will run in the background. When theft is detected, it can signal the device to lock the screen and send alarm email to the victim. Therefore our theft detector offers another layer of unobtrusive protection against smartphone theft.

In order to generate a labeled dataset for training and validation, we carefully design experiments that simulate three types of phone theft, grab-and-run when the user stands still, grab-and-run while the user is walking at a constant speed and pick-pocket. We conduct 20 trials for each type of theft in two sessions. We also conduct a user study, where we track 53 participants for 3 weeks and collect sensor data, such as accelerometer data, from their smartphones. We then extract 12 features from the acceleromter sensor data to constitute the training and validation sets. Our detector is triggered when the magnitude of the acceleration exceeds $40m/s^2$. Using this dataset, we train and evaluate three standard machine learning models: linear SVM, logistic regression and random forest. We show that for our purpose of smartphone theft detection, logistic regression produces 1 false alarm per week while detecting 100% simulated theft.

Our contributions are:

1. conducting a user study to collect a large dataset of smartphone sensor data while devices are being used in real world.
2. designing experiments to simulate theft and collecting data representing common smartphone theft scenarios.
3. developing a smartphone theft detecting system that detects 100% simulated theft at a cost of 1 false alarm per week.

2. RELATED WORK

The use of intrinsic sensor data to improve the security of smart devices and offer a way of continuous user authentication has drawn more and more attention recently. The embedded sensors on a smartphone have been used to develop biometric identification and authentication techniques. The major benefit of using smartphone-based biometric is to allow authentication to happen unobtrusively, i.e. not

requiring any users action. We are particularly interested in research done to study gait recognition by mining the embedded accelerometer data on smartphones. For the purpose of our research, we emphasize on the feature extraction methods used in the previous work.

The unobtrusive biometric gait recognition technique introduced by Derawi et al. uses accelerometer data collected on smartphones from 51 subjects in the context of a laboratory setup for continuous authentication and results in a high equal error rate of 20% [4]. They applied time interpolation and weighted moving average filter to clean the raw accelerometer data before extracting accelerometer measurements in average cycles as features.

Primo et al. further investigate and show that accelerometer-based gait authentication is somewhat dependent on the position in which the phone is held, which is a challenge for deploying gait authentication outside of a laboratory environment [10]. They preprocessed the raw measurements of X, Y, Z and magnitude (M) of the accelerometer data by a moving average of a 3 point window to mitigate sensor noise. They broke the resulting time series into windows, each containing 100 points and having an overlap of 50 points with the next window. They extracted and selected a set of top ranking features from X, Y, Z and M in each window to first determine phone position, then another set of highest ranked features to identify users. Their system was trained and achieved 80% accuracy in user identification on data from 30 subjects.

Juefei-Xu et al. researched ways to identify human walking patterns at normal and fast pace using accelerometer on a smartphone [8]. The acceleration is collected from 36 subjects walking at two different paces in a laboratory environment. They extracted feature vectors by concatenating x, y, z accelerations in two different intervals. One is a 3-second window centered around every spike in z acceleration. They also concatenated x, y, z accelerations from intervals between adjacent spikes after normalizing data in each interval to 500 samples. In addition, they applied signal processing methods to extracted and selected discriminative features. Their system achieved over 95% accuracy when it was both trained and tested on two same-pace datasets. The system identified subjects walking at fast pace with 61% accuracy when it was trained on data from the same subjects walking at normal pace.

Kwapisz et al collected a dataset of 36 subjects instructed by the researchers to perform 4 specific daily activities, walking, jogging, ascending and descending stairs [9]. They extracted features, mean, standard deviation, mean absolute difference, mean of magnitude, time between peaks, and binned distribution from 10-second intervals and use decision tree and neural network identify individual users. They achieve 90% accuracy using 10 second worth of data from one of the activities, walking. They also built a binary classifier for each subject to match a query to a particular user for the purpose of user authentication and achieved over 80% accuracy on positive authentication and over 90% accuracy on negative authentication.

In addition to gait, Feng et al investigated using pickup motion as a biometic modality for user authentication [7]. They explored two methods to extract features from accelerometer, gyroscope and magnetometer data. In the statistical method, they used a move average filter to smooth raw data, then extracted temporal and numerical features, i.e. duration time, mean, variance, and standard derivation after normalization and segmentation. They used SVM to classify if given data is from a specific user. They also used data from multiple sessions to reconstruct pickup motion trajectories and a distance metric to distinguishes trajectories of different users. They tested their proposed method on a dataset of 31 subejcts performing pickup motion while standing and walking achieved equal error rate of 6.13% and 7.09% respectively.

The most relevant work is done by Chang et al in using pickup motion from pockets or single-shoulder bags as a biometric modality, uniquely associated to the owner, to detect smartphone theft in real-time [2]. Their authentication system prevents pickpocket smartphone thefts by analyzing accelerometer and gyroscope data to detect illegal pickup of a smartphone. Our acceleration-based approach detects pickpocket as well as grab-and-run thefts when the phone is in the possesion of the owner. They preprocessed raw accelerometer and gyroscope data by removing high freqency noise. They integrated six weak classifier trained by data from each of six dimenstions of accelerometer and gyroscope into one strong classifier then use the strong classifier to determine the legitimacy of picking up motions of a user. Their system achieved 10.2% false positive rate and 5.5% false negative rate on average. [need to covert these rates to comparable results to our work, but the author did not provide the number of negative samples used in test.]

The aforementioned acceleration-based approaches for gait recognistion have a common procedure. They first preprocess the noisy raw sensor data usig a filter, then segment and extract feature vectors from windows to be used for a classification algorithm. While an accelerometer-based biometric modality enables theft prevention through user authentication, we are more interested in theft detection. To the best of our knowledge, not much work has been done in detecting pickpocket and grab-and-run smartphone theft, where a rapid change of acceleration occurs. We also do not have a set of subjects whom we intend to identify individually. In addition, instead of a laboratory environment, all the negative samples used in this work is collected in the real world, where sensor data is recorded on the smartphones carried by the subjects who are are performing everyday activities. We use the data recorded by the embedded accelerometer and apply machine learning techniques to detect smartphone theft.

3. METHODOLOGY

3.1 Data Collection

3.1.1 *Software and Hardware*

We use an Android application to record smartphone sensor data, including 3 axial acceleration, step count, ambient light, bluetooth connection etc, on a smartphone with Android version at least 5.0. It collects and transmits sensor data from a smartphone to a private account we set up on a cloud server. All log files that contain sensor data are encrypted using AES/CCM with an 11 byte Nonce and a 16 byte MAC upon an hourly upload to a cloud server. The application uses SENSOR_DELAY_FASTEST to acquire sen-

sor data as fast as possible [5]. On most devices including the one we use for the simulated theft experiment, the sampling rate is 100 Hz.

We use a commerically available smartphone, Nexus 5X, with Android version 7.1.1 in our simulated theft experiment. The embeded accelerometer is InvenSense MPU6515. For the user study, we require subjects to use an Android smartphone with Android version at least 5.0. Researchers also provide each participant with a Basis Peak smartwatch and explain to them that they may experience overheat while wearing it, and if it does, they should take it off and contact us immediately. The watch is paired with participant's smartphone via bluetooth. The bluetooth connectivity data collected by the appliction can be used to determine if the smartphone is in the range and connected to the watch. We use this heuristic as the ground truth of the smartphone being in user's possession.

3.1.2 Simulated Theft Experiment

We carefully design the experiment to simulate three types of smartphone theft scenarios with one researcher acting as a smartphone user, and another one playing the role of a thief. In the first scenario, the user stands still and holds the phone with one hand as she is using the device, for instance reading a text; the thief approaches from behind, grabs the phone with both hands and runs away in the forward direction. In the second scenario, the user again holds the phone in front of her with one hand while walking at a constant speed; the thief approaches from behind, grabs the phone with both hands and runs away in the forward direction. The third scenario simulates pick-pocket thefts. The user places the phone in a back pant pocket and stands still; the thief approaches from behind, steals the phone from user's pocket and runs away in the forward direction.

We collect 20 instances of each scenario and a total of 60 trials. Between two consective trials, researchers put the phone down on the ground for 30-50 seconds, which creats a gap in the time series to help seperate trials in the feature extraction step. We conduct all three scenarios at two different times, each of which consists of 10 trials per scenario and lasts approximately one hour. Different researchers act as the victim and thief in these two sessions. We choose to run the experiment on a flat ground at an open space, so the experiment is not be interrupted. We also make sure that the thief can run at least 40 feet after gaining possession of the victim's phone.

3.1.3 User Study

We obtained approval from the University of California, Berkeley IRB (Institutional Review Board). We conducted a user study in the Bay Area in the United States from September to December 2016 to collect smartphone sensor data while participants are performing their daily activities. The accelerometer data collected from this user study was then used to generate negative samples for the machine learning algorithms.

We first posted a recruitment advertisement on the Craigslist under the SF Bay Area 'jobs et cetera' category in September 2016. All subjects were required to take an online screening survey, in which they provided information about their age, gender, smartphone maker and model, the way to carry a phone, e.g. in a pocket or a purse, and whether they were comfortable with wearing a smartwatch. We only recruited participants who used an Android phone with version 5.0 and above, and were willing to wear a smartwatch during the study.

All qualified participants were scheduled a 30-minute meeting with a researcher. During the meeting, they were instructed to install the monitoring application described above on their smartphones. We asked participants to wear a smartwatch we provided for as long as possible during the study except while sleeping. At the end of the meeting, they signed a consent form, which explained the purpose, requirements, risks, confidentiality and compensation of the study. Participants then received $25 Visa gift card. During the user study, researchers contacted participants weekly to make sure that their phones and watches were functioning correctly. After completing the study, participants returned the watch, filled out a short exit survey and received compensation of $125 Visa gift card.

We had a total of 3 rounds; each round lasted 3 consecutive weeks. A total of 55 participants were recruited. 53 out of the 55 subjects completed the study. In the first round, 16 out of 18 participants finished the study. All 18 subjects who participated in the second round completed the study. So are all 19 participants in the third round. The detailed demographic information about the particpants of this user study, incluing gender and age distributions are listed in the table below.

	Male	Female	Age 20-29	30-39	40+
R1	5	11	8	6	2
R2	10	8	7	7	4
R3	11	8	9	4	6
Total	26	27	24	17	12

Table 1: Demographic Info of the User Study

3.2 Feature Extraction

We decide to use the magnitude of acceleration exceeding $40\ m/s^2$ as an activation condition for our detector. We extract features from one-second window before and n-second window after the time when the magnitude exceeds the threshold. By selecting a window both before and after the thresholdhe, the classifiers can capture the rapid change in acceleration before and after the moment when the smartphone was stolen. We choose $40\ m/s^2$ since it is approximately the vaule of magnitude when the phone is grabbed in our simulated theft experiment. We vary n from 1 to 7 to find the dataset where each classifier has the best performance.

We first select and evaluate 40 candidate features, i.e. minimum, maximum, mean, standard deviation, root mean square, arc length, product of arc length and standard deviation, and mean absolute of the x, y, z and magnitude of acceleration, in each window. We choose to only use the magnitude because compared to the x, y, z components of acceleration, the magnitude is non-directional thus more robust to various orientation of the phone. We remove minimum and mean absolute from the feature list because it does not affect the performance of the classifiers. As a result, we extract a 12-dimensional feature vector, 6 from before-window and 6 from after-window, every time the detector is triggered.

The detailed description of the selected features are listed below. *Maximum*: the maximum value of magnitude in a window.
Mean: the average value of magnitude in a window.
Standard deviation: the standard deviation of magnitude values in a window.
Root mean square: the rms of magnitude values in a window.
Arc length: the average of the differences between all adjecent magnitude values in a window.
Product of arc length and standard deviation: the product of the two feature values.

As a result, we generated 60 positive data points from the data collected in the simulated theft experiments, and approximately 248000 negative data points from the data collected in the user study. Each data point is a 12 dimentsional feature vector.

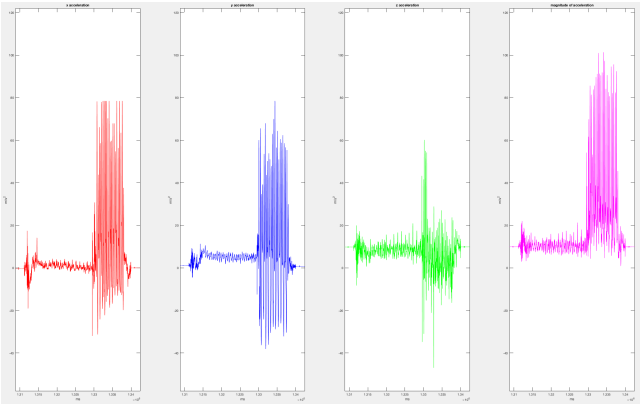

Figure 1: The above plots are x, y, z and magnitude of acceleration, respectively, of one theft instance.

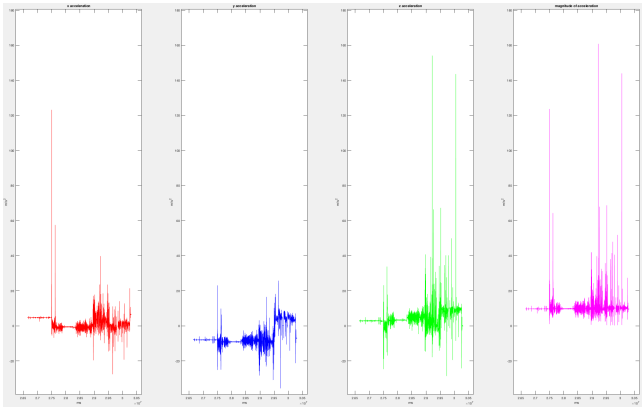

Figure 2: The above plots are x, y, z and magnitude of acceleration, respectively, during normal usage at an arbitrary time period.

3.3 Machine Learning Algorithms

We evaluate three standard machine learning algorithms: linear SVM, logistic regression and random forest, provide by the Python scikit-learn library, on their efficacy of distinguishing between theft and normal usage. In order to mitigate the fact that we have many more negative samples than positive ones, we tweak the class weight, a class attribute built in the scikit-learn library, to produce different ratios of negative and positive class weights used in training, for example $1:1$, $1:8000$, and 'balanced,' which adjusts the class weights to be inversely proportional to class frequencies in the input data.

4. RESULTS

We ran a 10-fold cross validation on the entire dataset described in the Feature Selection section, which consists of 60 positive samples and approximately 248,000 negative samples. Among the three classifiers, logistic regression performs the best in terms of producing the lowest false negative rate and the highest true positive rate. Besides class weights, another parameter we finetune in training is the window size n. Confusion matrices of logistic regression with class weight = 0: 1.0, 1: 200.0, random forest with class weight = 0: 1.0, 1: 5000.0 and linear SVM with class weight = 0: 1.0, 1: 1000.0 when the window size $n = 2$ second are shown below.

	False Negative	True Positive
True Negative	248223	170
True Positive	0	60

Table 2: Confusion Matrix of Logistic Regression

	False Negative	True Positive
True Negative	248360	33
True Positive	32	28

Table 3: Confusion Matrix of Random Forest

	False Negative	True Positive
True Negative	246522	1871
True Positive	42	18

Table 4: Confusion Matrix of Linear SVM

The logistic regression classifier has a false negative rate of 0.6844%, which means that on average users receive 1 false alarm every week, and a true positive rate of 100%. For our purpose of theft detection, linear SVM's false negative rate is too hight.

We also compute feature rankings to find the most predictive features for each classifier. For logistic regression, to calculate feature scores that are invariant to scaling feature values, we adjust the coefficient of each feature in the decision function by multipling it by the standard deviation of the values of this feature across all instances in the training set. Then we get the adjusted coefficients as the feature importances of logistic regression as -2.73319487, 0.0129842047, 0.477534932, -0.0143829911, -1.18427357, -0.0470492037, 0.0580951517, 0.880738694, 0.666823228, -0.937781183, 0.273227115, -2.69540663, from which we can tell the most representative features are maximum, and arc length in the before-window and product of arc length and

standard deviation in the after-window. For random forest, we use the built-in attribute feature importances in the scikit-learn library, which estimate the relative importance of the features by computing the expected fraction of the samples they contribute to. Thus the higher in the tree, the more important the feature is [11]. We get the feature importances of random forest as 0.04751106, 0.0080411, 0.02881483, 0.01533719, 0.02245781, 0.03309431, 0.12388036, 0.20288397, 0.16426152, 0.21908845, 0.05162059, 0.0830088.

Another way to find the feature importance is to plot a histogram of dataset and to see which features can better seperate the positive and negative data points, as shown in Figures 1 and 2.

To analyze the trade-off between random forest and logistic regression, we finetune the class weight of logistic regrssion to lower its true positive rate until it is approximately the same as random forest Then we compare their numbers of false postive instances. As a result, at a 41.7% true positive rate (25 true positive instances), logistic regression with class weigth = 0: 1.0, 1: 2.0 has 34 false positive instances. At a 46.7% true positive rate (28 true positive instances), random forest with class weight = 0: 1.0, 1: 5000.0 has 33 false positive instances. Thus the performance of logistic regression is at least as good as random forest.

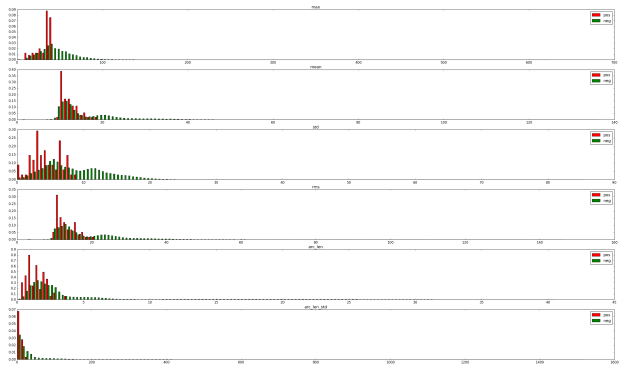

Figure 3: histogram of features in 1-second windows before *40* m/s^2 thresholds.

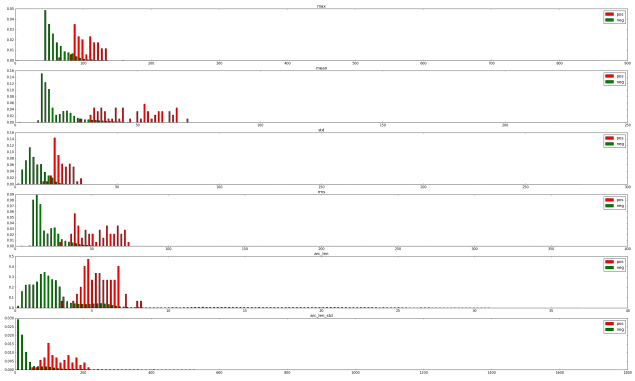

Figure 4: histogram of features in 2-second windows after *40* m/s^2 thresholds.

add ROC of random forest and logistic regression

5. DISCUSSION

In this work, we demonstrate that with proper feature extraction we can use accelerometer data alone to detect common smartphone theft, such as pickpocket and grab-and-run, without sacrificing user experience. In the future, in addition to accelerometer, we propose to utilize other intrinsic-sensor available on a smartphone, such as step count and try using techiques, such as time interpolation and filtering to preprocess raw sensor data for better performance. We would also like to collect more positive samples to have a more diverse theft dataset that covers more theft scenarios and test our models on a hold-out test set. We are interested in deploying theft detector on a smartphone and test it real time performance.

6. ACKNOWLEDGMENTS

Comment out for double blind review.

The authors would like to thank Prakash P. Bhasker and Micah J. Sheller for proivding with the Android sensor monitoring software, Jennider Chen from the Good Research for her assistance on conducting the user study, and Irwin Reyes Who else gives feedback to draft for giving feedback. This research was conducted at The Intel Science and Technology Center for Secure Computing (http://scrub.cs.berkeley.edu/) at UC Berkeley. The work is supported by What Grants and Funds.

7. REFERENCES

[1] J. L. Boyles, A. Smith, and M. Madden. Privacy and data management on mobile devices. `http://www.pewinternet.org/2012/09/05/privacy-and-data-management-on-mobile-devices/`, September 5 2012.

[2] S. Chang, T. Lu, and H. Song. Smartdog: Real-time detection of smartphone theft. In *Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2016 IEEE International Conference on*. IEEE, 2016.

[3] C. Deitrick. Smartphone thefts drop as kill switch usage grows. `http://www.consumerreports.org/cro/news/2015/06/smartphone-thefts-on-the-decline/index.htm`, June 11 2015.

[4] M. O. Derawi, C. Nickel, P. Bours, and C. Busch. Unobtrusive user-authentication on mobile phones using biometric gait recognition. In *Intelligent Information Hiding and Multimedia Signal (IIH-MSP) Sixth International Conference Processing*, pages 306–311. IEEE, October 2010.

[5] A. Developers. Sensor manager. `http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`, 2017.

[6] S. Egelman, S. Jain, R. S. Portnoff, K. Liao, S. Consolvo, and D. Wagner. Are you ready to lock? understanding user motivations for smartphone locking behaviors. In *2014 ACM SIGSAC Conference on Computer and Communications Security*

Proceedings, pages 750–761. ACM SIGSAC, November 2014.

[7] T. Feng, X. Zhao, and W. Shi. Investigating mobile device picking-up motion as a novel biometric modality. In *Biometrics: Theory, Applications and Systems (BTAS), 2013 IEEE Sixth International Conference on*. IEEE, 2013.

[8] F. Juefei-Xu, C. Bhagavatula, A. Jaech, U. Prasad, and M. Savvides. Gait-id on the move: Pace independent human identification using cell phone accelerometer dynamics. In *Biometrics: Theory, Applications and Systems (BTAS), 2012 IEEE Fifth International Conference on*. IEEE, 2012.

[9] J. R. Kwapisz, G. M. Weiss, and S. A. Moore. Cell phone-based biometric identification. In *Biometrics: Theory Applications and Systems (BTAS), 2010 Fourth IEEE International Conference on*. IEEE, 2010.

[10] A. Primo, V. V. Phoha, R. Kumar, and A. Serwadda. Context-aware active authentication using smartphone accelerometer measurements. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 98–105. IEEE, June 2014.

[11] scikit learn. 3.2.4.3.1. sklearn.ensemble.randomforestclassifier. `https://developer.android.com/reference/android/hardware/SensorManager.html`, 2016.