Department of Computer Science and Engineering

Indian Institute of Information Technology, KOTA

2015-2019

A Project Report

on

Exploiting user privacy using sensor data extracted from smart devices

Submitted by :
Puneet Saluja (2015KUCP1019)
Tanmay Sonkusle (2015KUCP1023)
Krishna Sharma (2015KUCP1040)

Supervisor:
Dr. Smita Naval

May 3, 2018

Declaration

We hereby declare that the project work entitled "**Exploiting user privacy using sensor data extracted from smart devices**" submitted at Indian Institute of Information Technology, Kota, is an authentic record of our work carried out under the supervision of **Dr.Smita Naval**. This project work is submitted in partial fulfillment of the requirements for the award of the degree of Bachelors of Technology in Computer Science and Engineering. The results embodied in this thesis have not been submitted to any other university or institute for the award of any other degree.

Puneet Saluja
(2015KUCP1019)

Tanmay Sonkusle
(2015KUCP1023)

Krishna Sharma
(2015KUCP1040)

Department of Computer Science and Engineering
Indian Institute of Information Technology, Kota
Jaipur, 302017
India

Acknowledgements

We are profoundly grateful to **Dr. Smita Naval** for her expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion. Her timely and efficient contribution helped us shape our work into its final form and we express our sincerest gratitude for her assistance in any way that we may have asked. We appreciate her guidance in our project that has improved our project many folds, thanks for the comments and advises.

Puneet Saluja
Tanmay Sonkusle
Krishna Sharma

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

A few years back people used to do daily activities differently, like making a simple phone call, transferring money, etc. People back then used land-line phones to make calls, they needed to visit a bank to transfer money, But now we can make calls from anywhere, checking the bank balance, transferring money and many other daily activities can be done easily. One of the reasons behind all these comforts is smart-devices. It is estimated that there will be 35 billion smart-devices by 2020. Smart devices include smart-phone, smart-watches, automated cars and many more. These devices are successful because of they provide real-time decision faster as they have processing capabilities.

1.2 Smart-devices

Internet of things (IoT), which adds sensors and internet capability to everyday physical objects has transformed the lives of individuals dramatically. The real-time insights and analysis of IoT devices deliver ecient and smart decisions faster. Nowadays, users rely on these devices to carry their personal data and day-to-day activity information. Now, this data can be sensitive if it contains credentials for an email account, bank details, medical information to name a few. To capture this information, IoT devices utilize various sensors (Analog and Digital). An

attacker can exploit this sensor data to extract the private details of the user as it has been seen in the past that security restrictions on sensors are negligible. Therefore, the sensor data can be exploited to threaten user's privacy.

1.3 Sensors

Today's smart devices are packed with nearly 14 different type of sensors that produce raw data on motion, location and the environment around us. An attacker can use these sensor's data to infer user activity like accelerometer and gyroscope data can be used to infer user's input keystroke, ambient light sensor data can be used to infer user's pin input and many other attack scenarios are possible. Currently, Android sensor manager asks for permission to access few sensors only such as Camera, GPS, and Microphone but it does not impose permission on zero-permission-sensors like Accelerometer, Gyroscope, Magnetometer and Proximity sensor. Android allows third-party applications to read and access sensor data without any limitations.

1.4 Experiments

In our project, we have performed two experiments which prove that the sensor data can be used as a side channel to exploit the privacy of a user. The main observation that we have made during our work is that touch input actions at the different position will bring the different level of motion and orientation change in smart-phones. We have created android applications to collect data from sensors in Android devices and later used this data to train model and test user input using WEKA which is open source machine learning software. In the first experiment, we have used accelerometer sensor data to detect the motion activity of a user like walking, jogging, going upstairs, going downstairs or stationary. In this experiment, we used Random Forest Classifier with 10 fold cross-validation to classify the user activity successfully with an accuracy

of about 94%. In our second experiment, we have demonstrated that accelerometer and gyroscope sensor data can be used to perform a side channel attack against the secure input. We have used these two sensors to learn user taps to identify the input digit on a number pad. We used the Random Forest Classifier to classify the input digit successfully with an accuracy of around 57.3%

Chapter 2

Literature Review

Input extraction via motion-sensor behavior analysis on smartphones. ChaoShen *et al.* [2]. They inferred user input using accelerometer and magnetometer sensors. They used the accelerometer data in all three axes and the orientation data in two axes as a source to represent the posture change of a smartphone.

When Good Becomes Evil: Keystroke Inference with Smartwatch. Xiangyu Liu *et al.* [4]. They investigated security issues that can be generated from smartwatch sensors especially using accelerometer and microphone.

MoLe: Motion Leaks through Smartwatch Sensors. He Wang *et al.* [5] They mined accelerometer and gyroscope data from smart watches to infer what user is typing.

PIN Skimming: Exploiting the Ambient-Light Sensor in Mobile Devices. Raphael Spreitzer *et al.* [6]. They demonstrated that ambient light sensor could be used to infer a user's PIN input.

iPhone: Decoding Vibrations From Nearby Keyboards Using Mobile Phone Accelerometer. Arunabh Verma *et al.* [7]. They detected and decoded the keystrokes by measuring relative position and distance between each vibration.

6thSense: A Context-aware Sensor-based Attack Detector for Smart Devices. Amit Kumar Sikder *et al.* [8]. They made a framework 6thSense which observes sensors data in real time and determines the current use context of the device according to which it concludes whether the current sensor use is malicious or not.

Gyrophone: Recognizing Speech from Gyroscope Signals. Dan Boneh *et al* [9] They used gyroscope sensor to infer acoustic signals. Since gyroscope is very sensitive and can detect changes due to voice signals

Practicality of Accelerometer Side Channels on Smartphones. Matt Blaze *et al* [1]. Authors demonstrated how to use the accelerometer sensor to learn user tap and gesture-based input as required to unlock smartphones using a PIN/password or Android's graphical password pattern.

Chapter 3

User Activity detection

3.1 Data Acquisition

For this experiment, we have developed an android application which captures the changes in accelerometer reading. The application uses the android sensor manager API to access the accelerometer data without any user consent. We collected their movement data from the smart-device that we have provided to them. We have used Samsung Galaxy J2 2016 device to collect accelerometer data for detecting user motion activity. The application logs the averaged value of the acceleration force applied on all three physical axes in ms^{-2} over a time interval of 1 second. We collected data from 10 different users in the handheld scenario for all the activities (walking, jogging, going upstairs, going downstairs and stationary).

3.2 Feature Vector Creation

After collecting the accelerometer data for a user activity, we pre-processed the data by calculating Z-Score and then applying mean normalization. Then we constructed the feature vector. The feature vector table consists of readings of the accelerometer in x, y and z-axis along with the timestamp. We collected this data for 2 minutes for each user and averaged out the reading data for a duration of 1 second. For each user, we collected data for two minutes thus 120 readings for x, y and

z-axis in our feature vector table and labeled class accordingly for all the activities.

3.3 Feature Vector Table

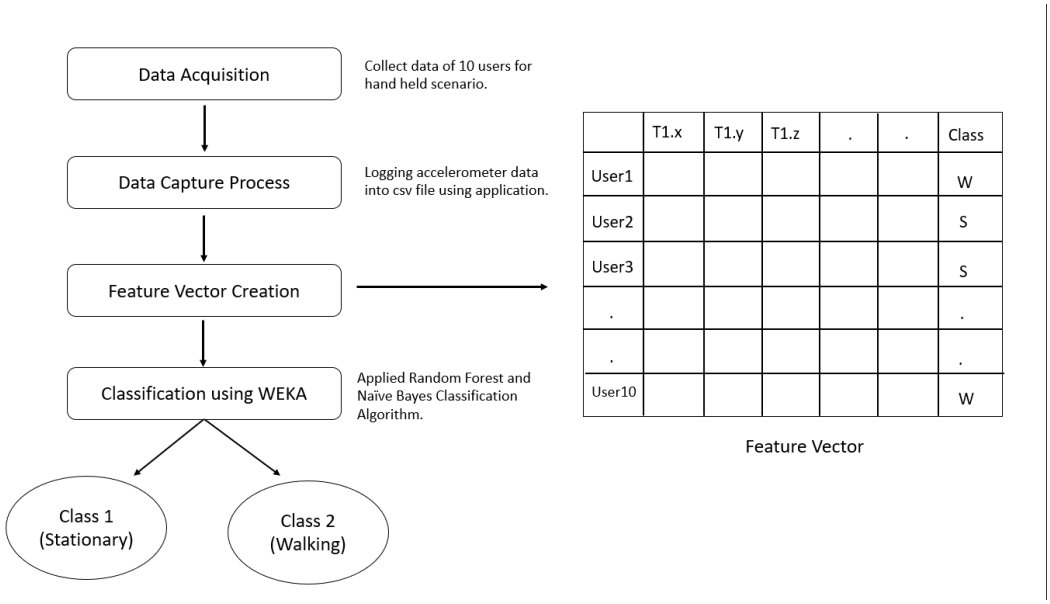

Figure 3.1: Feature Vector Table

3.4 Training and Testing

We used WEKA which is a machine learning software to train and test our model using 10 fold cross-validation. Using the Random Forest classifier, we were able to successfully classify the user motion activity with an accuracy of about 94%.

Chapter 4

Input Action Detection

4.1 Data Acquisition

For our experiments, we developed an android application which can record both accelerometer and gyroscope sensor data simultaneously. The application can run in the background and can be used to log the data from both sensors into a .csv file. Since Android does not impose any security restriction on these two sensors, no permission is asked for accessing these two sensors at the time of installation.

We have used One Plus 5 device to collect both accelerometer and gyroscope data for inferring user PIN input and to recognize each and individual key-press. The accelerometer gives the acceleration force along the three axis x, y, and z. The gyroscope gives the rate of the rotation of the device about all the three axes. We collected this data from 10 different users where each user entered 50 random pins of four-digit each and we allowed users to choose these pins at random of their choice.

4.2 Input Action Detection

To infer the user input the biggest challenge is to get the time of input, i.e. when the key is pressed. For this, we found out the squared sum of the accelerometer readings and plotted a graph along with the timestamp

$$AccSUM = x^2 + y^2 + z^2 \tag{4.1}$$

where x, y, and z are the readings of the accelerometer in the x, y, and z-axis respectively. This squared sum will give the magnitude of external force F on the touchscreen. The observed graph of the squared sum of accelerometer readings vs timestamp is as follows:

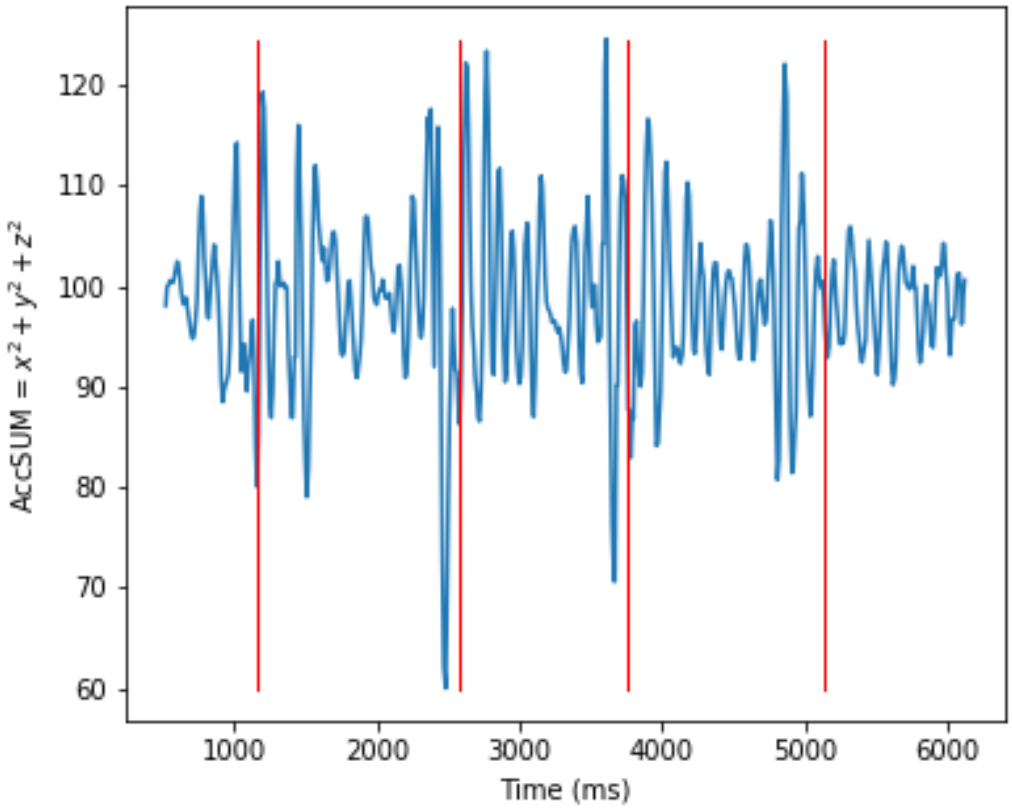

Figure 4.1: Observed Peaks on user taps

The red line here shows the time when a key is pressed. Though the graph had some peaks when the key is pressed but these peaks were not quite distinguishable. The reason behind this is that the accelerometer sensor captures the raw data which also includes gravity component

which makes it quite difficult to accurately infer the motion changes of the smartphones. Gravity component can be considered as a constant component. Thus, some filtering technique need to be employed in order to remove the gravity component from the accelerometer data.

```
// Constants for the low-pass filters
private float timeConstant = 0.18f;
private float alpha = 0.9f;
private float dt = 0;

// Timestamps for the low-pass filters
private float timestamp = System.nanoTime();
private float timestampOld = System.nanoTime();

// Gravity and linear accelerations components for the low-pass filter
private float[] gravity = new float[]{ 0, 0, 0 };
private float[] linearAcceleration = new float[]{ 0, 0, 0 };
// Raw accelerometer data
private float[] input = new float[]{ 0, 0, 0 };
private int count = 0;

public float[] addSamples(float[] acceleration)
{
    // Get a local copy of the sensor values
    System.arraycopy(acceleration, 0, this.input, 0, acceleration.length);

    timestamp = System.nanoTime();
    // Find the sample period (between updates).
    // Convert from nanoseconds to seconds

    dt = 1 / (count / ((timestamp - timestampOld) / 1000000000.0f));
    count++;
    alpha = timeConstant / (timeConstant + dt);
    gravity[0] = alpha * gravity[0] + (1 - alpha) * input[0];
    gravity[1] = alpha * gravity[1] + (1 - alpha) * input[1];
    gravity[2] = alpha * gravity[2] + (1 - alpha) * input[2];
    linearAcceleration[0] = input[0] - gravity[0];
    linearAcceleration[1] = input[1] - gravity[1];
    linearAcceleration[2] = input[2] - gravity[2];
    return linearAcceleration;
}
```

We looked into the Android Developer documentation for the implementation of this filter. We then performed all the calculations within the application and now we captured the accelerometer readings without any gravity component. Once again we plotted the graph of the squared sum of accelerometer readings vs timestamp and this time we

observed distinguished peaks at each keypress event. This curve can be used to accurately measure the occurrence of input actions since it exhibits periodic and obvious peaks.

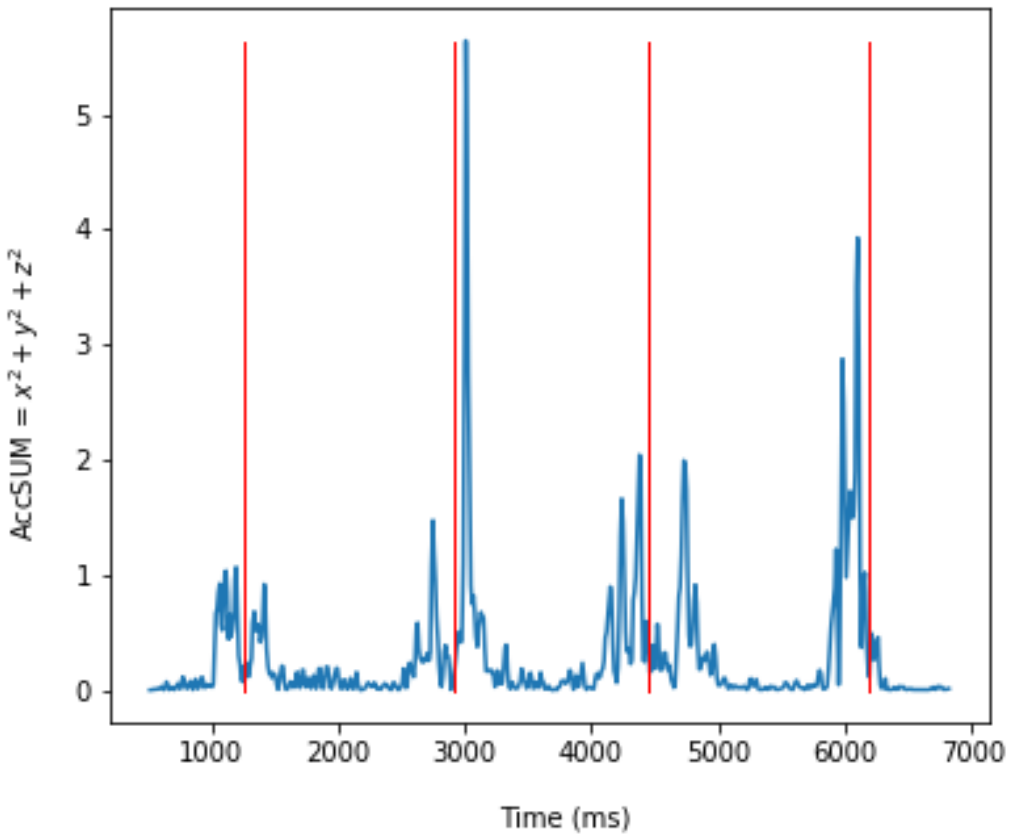

Figure 4.2: Observed peaks on user taps on applying filters

Chapter 5

Input Position Inference

5.1 Screen Division

A User might enter some valuable information on number pad such as passwords, PIN, mobile numbers etc. These interfaces are usually composed of similar layouts which include one display interface and one input interface. Thus, this input interface can be easily divided into different areas, each of which corresponds to a single digit on a number pad. The layout of a common input interface is shown in the following figure:

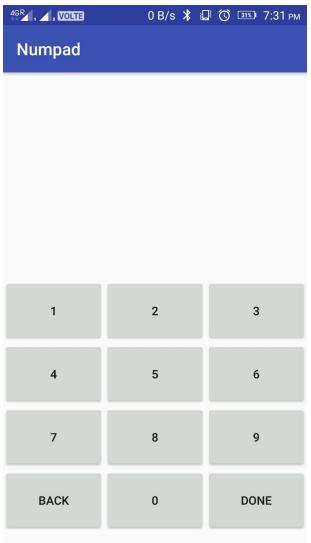

Figure 5.1: User Interface for numbers

5.1.1 Observations

Considering a user who is operating his smartphone, entering digits with the thumb of one hand, we make the following observations. A person who is right handed will slightly tilt the smartphone towards right side while he enters the PIN digit in the middle and the left column, i.e, 1, 2, 4, 5, 7, 8 because while making any input a user try to push smartphone towards his thumb. These tilts and turns will bring about the variation in the accelerometer and gyroscope readings which can be used to infer the input. We have made following assumptions based on these observations.

5.1.2 Assumptions

Assumption 1: We assume that a user holds the smartphone in his right hand while entering any PIN and we ignore the case when mobile is laying on a flat surface such as a table because in this case there will be only minor changes in the readings of the sensor data which makes input position inference quite difficult to analyze.

Assumption 2: We have not considered the QWERTY keyboard in which the numbers are present in a single row. We have assumed keypad as similar to the fig 5.1. Since it has been observed that for any authentication PIN the keypad layout is same as that of shown in fig 5.1.

5.1.3 Input Mapping

Input Interface of a number pad consists of several buttons where users can enter the information. One can map inferred position to input interface based on known knowledge of the inferred position. Input number pad can be divided into different areas, each of which will represent a digit which will have a fixed position range on the touchscreen. For the training data, we recorded pressed number and labeled the tap action data as the same class as that of the pressed number.

5.2 Posture Change Analysis

Whenever a user tries to perform an input action, it will bring about the change in the orientation data. Gyroscope gives the rate of rotation in rad/sec along the three axis x, y, and z. Whenever a digit is pressed it will cause the different level of changes in the rotation about the three axes thus giving the numerical values for the posture changes of the smartphones.

5.3 Feature Extraction

In this section, we describe the methods we used to pre-process the data and the feature set which is used as an input to the machine learning classifier. The raw data collected from the accelerometer and the gyroscope have some noise in it due to which using raw data directly in machine learning is not feasible. To pre-process the raw data we did two things, we first calculated Z-Score of the data and the applied mean normalization on that data.
Z score of the value tells the number of standard deviation the value is away from the mean value.

$$Z_i = \frac{x_i - \overline{x}}{\sigma} \tag{5.1}$$

$$x_i = \frac{x_i - \overline{x}}{x_{max} - x_{min}} \tag{5.2}$$

We used this pre-process data to generate the feature vector table. Feature vector includes the readings of the accelerometer and gyroscope with timestamp as a feature and additional descriptive-statistical attributes of A_x, A_y, A_z, G_x and G_y like min, max, median, kurtosis, mean, standard deviation, and variance.

5.4 Training and Testing Data

We used the Random Forest Classifier to train and test the data using 10 fold cross-validation on WEKA.

```
=== Confusion Matrix ===

   a   b   c   d   e   f   g   h   i   j   <-- classified as
 143  27   0   0   0   0   0   0   0   0 |   a = c1
  33 169   1   1   0   4   7   0   2   0 |   b = c4
   0   5  53  43  27  28  10   1   1   8 |   c = c8
   0   1   5 214  21  14  12  24   0   1 |   d = c6
   0   0  11  99  61   9   0   4   0   3 |   e = c9
   0   2  19  50  12  83  62  22   0   1 |   f = c5
   0  19   3  14   3  46 118  23   1   0 |   g = c2
   0   2   1  46   3  22  35 103   0   0 |   h = c3
   2  16   0   0   0   0   0   0 135   3 |   i = c7
   1   4  10   3   6   1   1   1  19  66 |   j = c0
```

Figure 5.2: Confusion Matrix

Following are some test cases:

input: 4 3 8 0 0 5
1st attempt predictions: 4 2 7 0 0 5
2nd attempt predictions: 4 3 8 0 0 5

input: 2 4 8 3 5 9 6
1st attempt predictions: 2 4 8 3 5 6 5
2nd attempt predictions: 2 4 8 3 5 3 6

input: 6 0 5 2 4 7 6 0
1st attempt predictions: 6 0 5 2 1 7 6 0
2nd attempt predictions: 6 0 5 2 4 7 6 0

Here we observed that our model correctly predicted the entered digits with an accuracy of around 57.3% in the first guess. And if consider till the second probability in the probability distribution of the digits, it almost correctly predicts the input with much higher accuracy. Here in the confusion matrix, we observed that cases like 8, 9, and 5 are not predicted with good accuracy. The reason behind that is that we assumed that user is typing with the right hand and if we check, while typing with the right hand there are very minute tilts and turns while typing 8, 9 or 5, and all the tilts and turns are almost similar when user type these numbers. Therefore it is hard to predict those numbers accurately for a right hand user.

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

In this project, we have presented a study of analyzing accelerometer and gyroscope data extracted from smart-devices to infer user input on an Android smart-phone. We were able to detect user motion activity (walking, stationary, jogging, walking upstairs, walking downstairs) using accelerometer data with an accuracy of 94%. In our second experiment, we were correctly able to infer each individual key press on a number pad with an accuracy of 57.3%. We also examined the size of training data on the performance of our model and found that accuracy increases with the increase in the training data size. We also observed position inference accuracy against different sampling rates of sensor data and observed input action detection were quite accurate in case of the high sampling rate.

With this work, we have successfully demonstrated that the leaked information from these sensors can act as a side channel to compromise user's privacy. Since no permission is required to access data from these sensors, an attacker is able to collect sensitive information without raising any suspicion.Thus, Android should impose some security restrictions on these sensors. We hope that our project could raise awareness among people about the security of smart-devices as they may threaten their privacy.

6.2 Future Work

We have shown that user input numbers can be detected using the sensors in smart-devices, this work could be extended in future to infer other kinds of attack like inferring user input text and we can use some more sensors such as magnetometer and ambient light sensors or use a combination of these sensors to carry out these attacks. Not only that we can extend our work to infer acoustic signals from the data extracted from various sensors in smart-devices

Bibliography

1 Adam J. Aviv, Benjamin Sapp, Matt Blaze and Jonathan M. Smith, "Practicality of Accelerometer Side Channels on Smartphones", in *Proceedings of the 28th Annual Computer Security Applications Conference*, Pages 41-50.

2 ChaoShen, Shichao Pei, Zhenyu Yanga, Xiaohong Guan, "Input extraction via motion-sensor behavior analysis on smartphones", Volume 53 Issue C, September 2015, Pages 143-155.

3 Zhi Xu, Kun Bai, Sencun Zhu, "TapLogger: Inferring User Inputs On Smartphone Touchscreens Using On-board Motion Sensors", in *the Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks*, Pages 113-124.

4 Xiangyu Liu, Zhe Zhou, Wenrui Diao, Zhou Li, Kehuan Zhang, "When Good Becomes Evil: Keystroke Inference with Smartwatch", in *the Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, Pages 1273-1285.

5 He Wang, Ted Tsung-Te Lai, Romit Roy Choudhury, "MoLe: Motion Leaks through Smartwatch Sensors", in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, Pages 155-166.

6 Raphael Spreitzer, "PIN Skimming: Exploiting the Ambient-Light Sensor in Mobile Devices", in *Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*.

7 Philip Marquardt, Arunabh Verma, Henry Carter, Patrick Traynor, "iPhone: Decoding Vibrations From Nearby Keyboards Using Mo-

bile Phone Accelerometers", in *Proceedings of the 18th ACM conference on Computer and communications security*, Pages 551-562.

8 Amit Kumar Sikder, Hidayet Aksu, and A. Selcuk Uluagac, "6thSense: A Context-aware Sensor-based Attack Detector for Smart Devices", in *the Proceedings of the 26th USENIX Security Symposium, 2017.*

9 Yan Michalevsky and Dan Boneh, "Gyrophone: Recognizing Speech from Gyroscope Signals", *Proceedings of the 23rd USENIX Security Symposium, 2014.*