AccelPrint: Imperfections of Accelerometers Make Smartphones Trackable

Sanorita Dey[1,4], Nirupam Roy[1,4], Wenyuan Xu[2], Romit Roy Choudhury[1] and Srihari Nelakuditi[3]

[1]University of Illinois at Urbana-Champaign
{sdey4, nroy8, croy}@illinois.edu
[2]University of South Carolina and Zhejiang University
wyxu@cse.sc.edu (Corresponding Author)
[3]University of South Carolina
srihari@cse.sc.edu
[4]Co-primary authors

***Abstract*—As mobile begins to overtake the fixed Internet access, ad networks have aggressively sought methods to track users on their mobile devices. While existing countermeasures and regulation focus on thwarting cookies and various device IDs, this paper submits a hypothesis that smartphone/tablet accelerometers possess unique fingerprints, which can be exploited for tracking users. We believe that the fingerprints arise from hardware imperfections during the sensor manufacturing process, causing every sensor chip to respond differently to the same motion stimulus. The differences in responses are subtle enough that they do not affect most of the higher level functions computed on them. Nonetheless, upon close inspection, these fingerprints emerge with consistency, and can even be somewhat independent of the stimulus that generates them. Measurements and classification on 80 standalone accelerometer chips, 25 Android phones, and 2 tablets, show precision and recall upward of 96%, along with good robustness to real-world conditions. Utilizing accelerometer fingerprints, a crowd-sourcing application running in the cloud could segregate sensor data for each device, making it easy to track a user over space and time. Such attacks are almost trivial to launch, while simple solutions may not be adequate to counteract them.**

I. Introduction

With more than $700,000$ apps available in the Google Play and App Store [6], [52], smartphones and tablets have emerged as the most popular platforms to assist our daily activities and to exchange information over the Internet. Most apps are offered as free with ads, which allows ad networks to collect data for tracking users and their online habits. While such tracking can be lucrative for advertising companies [47], it raises major privacy concerns for users.

Cookies were one of the most widely used mechanisms to track users. To address the privacy concern of tracking users, a "cookie law" has been enforced in the US and Europe [18], which requires apps to obtain user-permission before uploading cookies or any other identifiers to the cloud. Nevertheless, research [28] shows that stealing of various IDs, such as the IMEI (device ID), IMSI (subscriber ID), or ICC-ID (SIM card serial number), is still rampant in apps. Some recent proposals [27], [69] have designed solutions to thwart ID-theft. Nevertheless, we have to be vigilant since the unavailability of cookies and various IDs is likely to motivate advertisers to find new ways of linking users to their app usage habits or browsing histories, if past experience is an indication. Commercial advertising companies, such as BlueCava and Iovation, have already started to identify devices and link users based on browser configuration, screen resolution [47], etc.

In this paper, we explore cookieless methods to identify devices. Inspired by past work on device fingerprinting [37], [57], [64], where WiFi chipsets were shown to exhibit unique clock skews and frequency offsets, we ask the question: *could sensors in today's smartphones also have unique fingerprints?* Hardware imperfections are likely to arise during the manufacturing process of sensors, suggesting the existence of fingerprints. However, sensors, such as accelerometers, are known for generating noisy readings. Therefore, can sensors' fingerprints be consistently measured for device identification? In the pursuit of this question, we gathered, over time, around 80 standalone accelerometer chips used in popular smartphones, subjected each of them to vibrations from a single vibration motor (common in today's phones), and analyzed the large volume of data received from each of them. We found that while high level operations

NDSS '14, 23-26 February 2014, San Diego, CA, USA
 ISBN 1-891562-35-5
http://dx.doi.org/10.14722/ndss.2014.23059

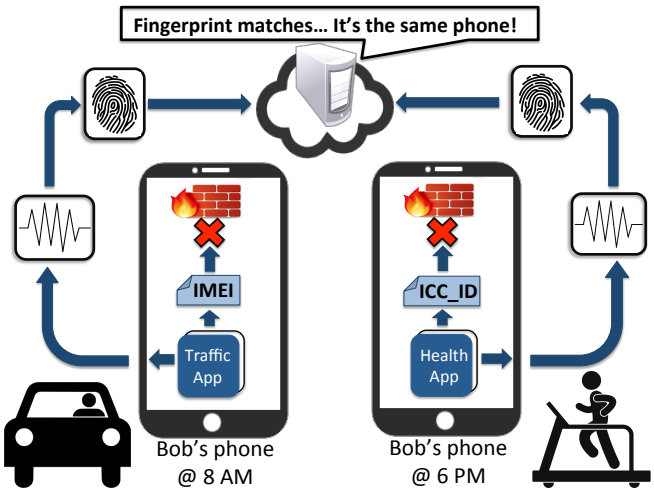

Fig. 1: Example threat: Bob uses traffic and health apps, supported by the same cloud backend. Even when device IDs are blocked, exporting a slice of sensor data enables the cloud to infer that it is the same user.

on the accelerometer signals yielded similar results, e.g., all the chips were comparable in counting the number of walking steps, an appropriately designed high dimensional feature-vector exhibited strong diversity, a fingerprint.

Our initial skepticism that this fingerprint is an outcome of non-identical vibrations was dispelled when a given accelerometer repeatedly exhibited the same distinct pattern. Moreover, we found that the fingerprints persist even if the vibrations are subjected in less controlled settings, e.g., when the user is naturally holding an accelerometer-equipped phone. Even different phone cases made of rubber or plastic did not affect much, so long as the system was trained on those casings. We have also conducted experiments subjecting smartphones to rotational motion instead of vibration, which too affirmed accelerometer diversity. Finally, our attempts to scrub off the fingerprint (without affecting the high level functions such as step-count) did not meet immediate success. Inducing small amounts of noise in the accelerometer signal still preserved the fingerprint; adding too much noise affected the activity and gesture recognition applications. This paper reports our effort to verify the existence of accelerometer fingerprints, and draws attention to new kinds of threats that may arise as a consequence.

Figure 1 illustrates one possible threat. Consider a common scenario where multiple motion-sensing apps, such as a road traffic estimator, a calorie counter, a gesture-based gaming app, etc., all implanted with third-party ads. While the cookie law and some recent proposals [27], [69] may thwart attackers from conveying cookies or various IDs, we observe that sensor data is not subjected to scrutiny since it is legitimately required by apps. Thus, if the sensor data can be used to identify devices, the advertising companies can easily bypass the cookie law, and track the users by sensor fingerprints. Put differently, an accelerometer fingerprint can serve as an electronic cookie, empowering an adversary to consolidate data per user, and track them over space and time. Alarmingly, such a cookie is hard to erase, unless the accelerometer wears out to the degree that its fingerprint becomes inconsistent. We have not noticed any evidence of this in the 9 months of experimentation with 107 accelerometers.

The notion that sensors can offer side-channel information is not new. Past work has demonstrated how accelerometers can leak information in smartphones – for instance, from accelerometer data gathered during typing, authors in [15], [16], [43] have shown that the typed characters, such as PIN numbers, can be inferred. Even swiping motion patterns can be estimated [10]. While disabling the accelerometer during a sensitive operation (e.g., typing PINs) is a plausible solution, the same does not apply in our case because even a small slice of the sensor reading is adequate to extract the fingerprint. Another alternative could be to perform the computations locally on the phone and only send the higher level results to the cloud. However, some operations are far too CPU-heavy to be performed on-phone, while others require matching against large databases that are expensive to download to the phone. Pre-processing the readings and scrubbing off the fingerprint is probably the appropriate approach, however, as we find later, this requires deeper investigation in the future. Scrubbing without an understanding of the app is risky – an app that needs high fidelity readings could easily be affected upon over-scrubbing.

A natural question on sensor fingerprints pertains to scalability, i.e., *is the fingerprint unique against millions of sensors?* We admittedly have no proof of such large scale, neither a theoretical basis to justify our claim. We have only attempted to lease/gather as many devices as possible, amounting to: (1) 80 stand-alone accelerometer chips of three types (used in the latest smartphones and tablets, including the Samsung Galaxy S III and Kindle Fire). (2) 25 Android phones composed of a mix of Samsung S3, Galaxy Nexus, and Nexus S. (3) 2 Samsung tablets. Each of the standalone chips were plugged into the same customized circuit board connected to an external vibration motor to provide the motion stimulus. As a result, the recorded accelerometer readings are free of any potential effects caused by the OS version and the middleware of smartphones. The Android phones and tablets were used as is; the stimulus induced by programming its on-board vibration motor.

The sensor fingerprint is designed as a vector of 36 features drawn from the time and frequency domain of accelerometer signals. A Bagged Decision Tree [20] is used for ensemble learning and classification. Results show that among these sensors, classification precision and recall reach upwards of 96%. Moreover, the fingerprints proved to be robust, visible even through natural hand-held positions, and even for various casings, including one of soft rubber.

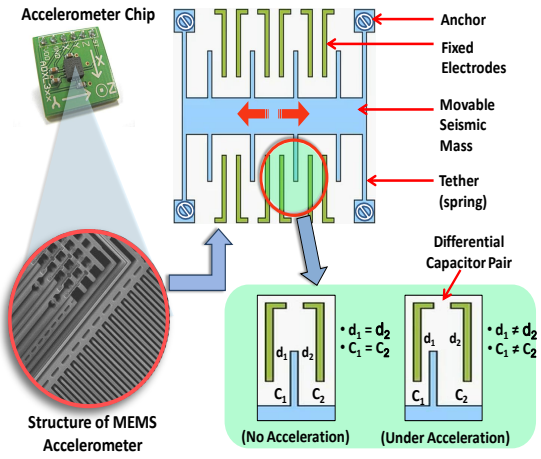

Fig. 2: The internal architecture of MEMS accelerometer chip used in smartphones.

While more extensive evaluation is warranted to verify the hypothesis (perhaps in an actual manufacturing pipeline), we believe that our results are still valuable. To the best of our knowledge, this is the earliest work that suggests and verifies (in a lab setting) that accelerometers in modern smartphones are identifiable. We call the overall system, `AccelPrint`.

II. Accelerometers: A Closer Look

This section presents a brief background on accelerometers to qualitatively reason about the source of fingerprints. Then, we describe our experiment framework and present early evidence of accelerometer fingerprints. Detailed results and associated issues are presented in the evaluation section.

A. Hardware Imperfections

Accelerometers in smartphones are based on Micro Electro Mechanical Systems (MEMS) that emulate the mechanical parts through micro-machining technology [8]. Figure 2 shows the basic structure of an accelerometer chip, composed of several pairs of fixed electrodes and a movable seismic mass. The distances d_1 and d_2 represent the small gaps that vary due to acceleration and form a differential capacitor pair. The chip measures the acceleration according to the values of these differential capacitor pairs. It is the lack of precision in this electro-mechanical structure that introduces subtle idiosyncrasies in different accelerometer chips. Even slight gaps between the structural parts (introduced during the manufacturing process) can change the capacitance [8]. Moreover accelerometer chips use Quad Flat Non-leaded (QFN) or Land Grid Array (LGA) packaging, another potential source of imperfections [22].

According to the official data sheets, the target applications for smartphone accelerometers are gesture recognition, display rotation, motion-enabled games, fitness monitoring, etc. These applications primarily depend on the relative change in the accelerometer readings as opposed to their absolute values. Therefore, while subtle imperfections in the accelerometer chips can lead to different acceleration values, they may not affect the rated performance of the target applications. However, these discrepancies may be sufficient to discriminate between them.

B. Evidence of Fingerprints

To gain early evidence on the existence of fingerprints, we conducted an experiment using 6 stand-alone accelerometer chips of 3 types: (i) MPU-6050; (ii) ADXL-345; and (iii) MMA-8452q. MPU-6050 is a MEMS chip [5] used in many mobile devices, including the Galaxy S III and Kindle Fire. The ADXL-345 is a small, thin, ultra-low power 3-axis accelerometer [1] with a high resolution of 13 bits and scaling up to $\pm$ 16g (where g is acceleration due to gravity). This is mainly used for tap/swipe sensing and activity recognition. MMA-8452q is a 12 bit digital 3-axis low-power capacitive accelerometer [4], available in QFN packaging, and configurable to $\pm$2g/$\pm$4g/$\pm$8g through high-pass filters. The mix of chips included in the experiment are 2 MPU-6050 from two different vendors (SparkFun and Amazon), 3 ADXL-345, and 1 MMA-8452q. We setup the Arduino Uno R3 boards [2] to collect accelerometer readings from the chips. We use an external vibration motor – the model used in most smartphones – to stimulate the accelerometer with a specific vibration duty-cycle, controlled through the Arduino board. Fig. 3 shows the experimental setup.

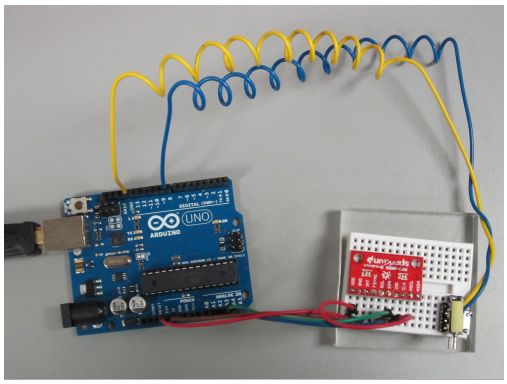

Fig. 3: Experimental setup with the Arduino board on the left, the red accelerometer chip on the breadboard, and the vibration motor connected over the wire.

Each of the six stand-alone chips are stimulated with an identical vibration sequence and their accelerometer readings are recorded. Figure 4a shows the *root sum square* (RSS) of the three axes values against time. The plots on each column are distinct but the elements in the top two rows look similar (i.e., tagged "A" and "B", "C" and "D"). To separate them out, Figure 4b plots the mean RSS values against their standard deviations (i.e., in a 2-dimensional plane). Each experiment on a chip yields a data point on the graph and the points from multiple experiments on the same chip exhibits a cluster. The top two rows that appear similar in Figure 4a

begin to separate out on this 2-dimensional plane, although some overlap still remains. Of course, other features might be more effective in reducing the overlap.

To further distinguish the 6 chips from each other, we considered a feature called *skewness*, which measures the asymmetry of a probability distribution. Figure 4c shows the skewness of the accelerometer readings of the two similar MPU-6050 chips (tagged "C" and "D"). Evidently, one consistently shows a higher skewness over the other even though they are the of same make and model. This suggests that chips that appear indistinguishable on one dimension may be well separated on others. However, these three features, RSS mean, std. deviation, and skewness, alone are not sufficient to discriminate all accelerometers. Therefore, recruiting an appropriate set of feature vectors and projecting the accelerometer signals on them may demonstrate that accelerometers could indeed be unique.

An accelerometer fingerprint (under controlled vibration sequences) may not necessarily translate to a smartphone fingerprint in the real world. First, the OS running on the phone, application API, and CPU load, can all influence the sensor readings. Second, considering that fingerprinting is based on subtle features in response to brief vibrations, the surface on which the device is placed, or its casing, may also matter. While these make fingerprinting a naturally-used smartphone more challenging compared to a standalone accelerometer, we observe that additional sensors on the phone could be harnessed as well. A gyroscope, barometer, and accelerometer may together exhibit a fingerprint robust to OS versions, CPU-load, and surfaces. While we leave this exploration to future work, in this paper we show that accelerometers alone can achieve reasonable smartphone fingerprinting under uncontrolled conditions. Naturally, this makes the threats imminent.

III. Application Scenarios

A. Threat Model

We consider an adversary that aims at identifying smartphones but cannot gain access to unique device IDs (e.g., IMEI or ICC-ID). This can be because these IDs are protected by monitoring strategies [27], [28]. Thus, the adversary tries to obtain the fingerprint of the built-in sensors (e.g., accelerometers). We assume that the adversary is able to interact with apps on the smartphone, has access to accelerometer data, and can communicate over networks.

Smartphone Access. We assume that an adversary can access apps that are either installed legitimately by a user or affected by malware. In either case, the adversary can interact with the smartphone through the apps over the communication networks. For instance, an adversary could be an advertiser (e.g., ad networks) that wants to obtain users' personal data for supplying targeted ads to boost the likelihood of purchase. The current practice of free apps with ads is the following. Advertisers provide prepackaged developer kits (e.g., iApp) which allows app-developers to get revenue by including a few lines of code into their apps. The code not only displays ads in the app, but also allows to collect data from the device and share it with ad networks.

Sensor Access. We assume that an adversary is able to collect raw sensor readings directly. Such an assumption is easy to satisfy, because among all smartphone sensors, only the location sensor requires explicit user access permission on both Android and the iPhone platforms and other sensors (e.g., accelerometers and gyroscope) can be accessed without notifying users. Even if in the future, explicit permission is required to access sensors, the apps could be legitimately granted permission to sensors and the adversary may inherit such a permission for accessing sensors.

Packaged Sensors. Since it is difficult to replace the sensors inside a smartphone, we assume that throughout the operational lifetime of a smartphone, the sensors on the smartphone remain unchanged.

B. Attack Scenarios

Once an adversary gains access to a smartphone, she can use the official APIs to acquire sensor readings and upload a short segment of raw readings to the cloud for fingerprint extraction. Alternatively, with the increasing computational capability of smartphones, the fingerprint of sensors can be extracted locally on the phone and only the sensor fingerprints are uploaded to the cloud. Given that sensors inside a smartphone are rarely replaced, their innate imperfections can create, to some extent, a permanent fingerprint of a smartphone. One major consequence is that such a built-in and consistent fingerprint can act as a trackable identifier of the smartphone's owner.

For instance, a health-conscious and commuting user (hereafter Bob) may install apps for monitoring his daily activities and for traffic condition. All these apps rely on inertial sensors (e.g., accelerometer, gyroscope) and could be loaded with ads supplied by the same ad networks. As a result, the ad networks can collect detailed data of Bob along with Bob's sensor readings/fingerprints. When the ad networks observe Bob's sensor fingerprints for the first time, they create a profile for Bob, which will be expanded with new data from Bob. Even after Bob uninstalls all these apps, the fingerprints and Bob's profile remain in the digital world. The data collection of Bob may be paused briefly, until Bob installs a new app with ads. Then, the ad networks extract the sensor fingerprint and index through the profile database to find Bob, and resume the data collection.

IV. AccelPrint Design

This section describes the 3 sub-modules of the overall system: (1) Accelerometer data collection; (2) Fingerprint generation; (3) Fingerprint matching.

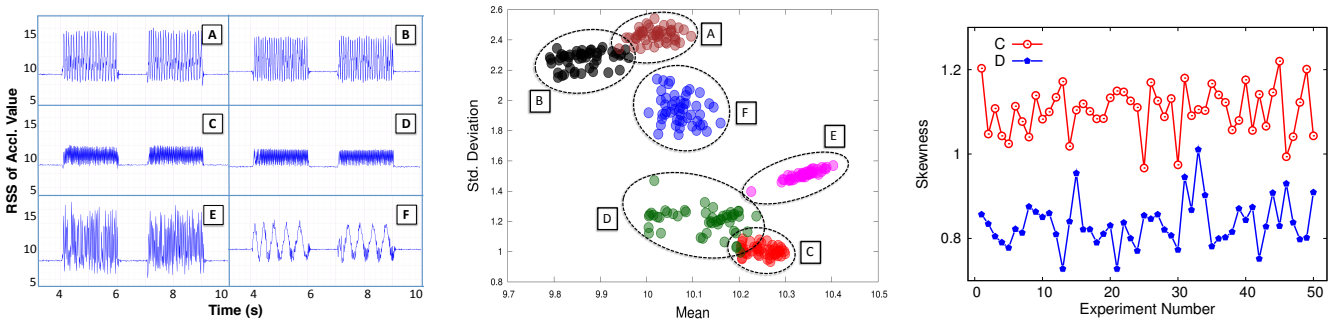

Fig. 4: Accelerometer responses of 6 chips for the same stimulation: (a) Using *Root Sum Square* (RSS) of the three axes over time offers some differentiation among chips; (b) Clustering on 2 dimensions – RSS mean and deviation – improves separation; (c) Clusters that overlap with mean/deviation, separate out further using a *Skewness* feature.

A. Accelerometer Data Collection

We define the accelerometer fingerprint as the response it yields to any predefined motion stimulus. We found that the vibration motor internal to a smartphone – mainly used to "buzz" the device – generates consistent motion stimuli, and can be programmed to ON/OFF states at fine time scales. Hence, we collect accelerometer data during time windows when the vibration motor is ON, and call this raw data a *trace*. Of course, the vibration motor need not to be explicitly turned on for trace collection (or else a malware may raise the user's suspicion due to frequent vibrations). Instead, the malware could opportunistically collect the accelerometer data whenever the vibration motor is active, perhaps due to an incoming email, SMS, phone-rings, or other alerts and push notifications.

A natural question is *how can one detect when the vibration motor is active, given that no standard Android API is available to check its ON/OFF status?* `AccelPrint` uses the accelerometer data itself to identify portions during which the vibration motor was ON. This is feasible mainly due to 2 factors: First, the to-and-fro motion generated by a vibration motor is faster than any normal human activity. Second, based on our analysis on 6 types of Android devices (4 smartphones and 2 tablets), the effect of a vibration motor is significantly higher on the Z-axis irrespective of the device orientation. This is because a motor is typically mounted on the phone such that it has greater movement freedom along the Z-axis. Leveraging this observation, our detection algorithm calculates the derivatives of the acceleration in all 3 axes and compares them against empirically designed thresholds. We tested our scheme by turning on the vibration motor at random duty cycles (we used the "fastest" sampling mode in Android). Figure 5 shows the results – the detection is reliable across various user activities, including when driving a car, placed on a table top, walking, running, etc.

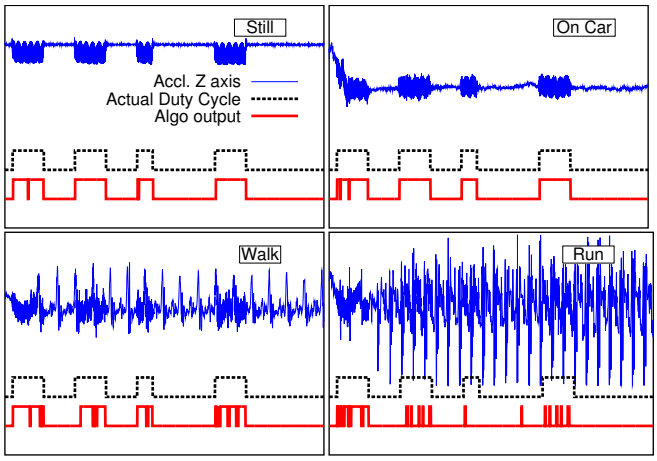

Fig. 5: Identifying when the vibration motor is ON from the accelerometer readings directly.

B. Fingerprint Generation

Trace Pre-processing. Instead of extracting features from a raw trace, `AccelPrint` pre-processes the trace to obtain two sets of intermediate data: one represents how often an accelerometer reading was recorded and one represents the absolute value of accelerometer readings. Let $\{s_x(k),\ s_y(k),\ s_z(k)\}$ be the kth acceleration along x, y, and z axes, and $T(k)$ be the timestamps. `AccelPrint` calculates sampling intervals $I(k)$ and the root sum square (RSS) of accelerometer readings $S(k)$ as follows.

$$\begin{cases} I(k) = T(k+1) - T(k) \\ S(k) = \sqrt{s_x^2(k) + s_y^2(k) + s_z^2(k)} \end{cases}$$

Since $\{s_x(k),\ s_y(k),\ s_z(k)\}$ are not sampled at a fixed interval, the derived values $\{T(k), S(k)\}$ are not equally-spaced. This makes the frequency domain characteristics difficult to compute. Hence, `AccelPrint` employs a cubic spline interpolation [44] to construct new data points such that $\{T(k), S(k)\}$ are now equally-spaced.

Feature Selection. We extract 40 scalar features in both time and frequency domains using LibXtract [3], a popular feature extraction library. The time domain features are calculated using $\{T(k), S(k)\}$ prior to interpolation, and the frequency domain features are drawn from the interpolated

version. Since we consider features for both $S(k)$ and $I(k)$ (where $I(k)$ is the interval between samples), a total of 80 features are available for use. To select features, we ranked features using the FEAST toolbox [7] and utilized the joint mutual information criterion for ranking (known to be effective for small training data [14]). From the results, we select the top 8 time domain features (see Table I) and top 10 frequency domain features (see Table II). In total, 36 features are used to construct the fingerprint.

TABLE I: List of Time Domain Features. Vector x is the time domain representation of the data. N is the number of elements in x.

Feature Name	Description
Mean	$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x(i)$
Std-Dev	$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x(i)-\bar{x})^2}$
Average Deviation	$D_{\bar{x}} = \frac{1}{N}\sum_{i=1}^{N}\lvert x(i)-\bar{x}\rvert$
Skewness	$\gamma = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{(x(i)-\bar{x})}{\sigma}\right)^3$
Kurtosis	$\beta = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{(x(i)-\bar{x})}{\sigma}\right)^4 - 3$
RMS Amplitude	$A = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x(i))^2}$
Lowest Value	$L = (Min(x(i))\lvert_{i=1\ to\ N})$
Highest Value	$H = (Max(x(i))\lvert_{i=1\ to\ N})$

TABLE II: List of Frequency Domain Features. Vector y is the frequency domain representation of the data. Vectors y_m and y_f hold the magnitude coefficients and bin frequencies respectively. N is the number of elements in y_m and y_f.

Feature Name	Description
Spec. Std Dev	$\sigma_s = \sqrt{\left(\sum_{i=1}^{N}(y_f(i))^2 * y_m(i)\right) / \left(\sum_{i=1}^{N} y_m(i)\right)}$
Spec. Centroid	$C_s = \left(\sum_{i=1}^{N} y_f(i) y_m(i)\right) / \left(\sum_{i=1}^{N} y_m(i)\right)$
Spec. Skewness	$\gamma_s = \left(\sum_{i=1}^{N}(y_m(i)-C_s)^3 * y_m(i)\right) / \sigma_s^3$
Spec. Kurtosis	$\beta_s = \left(\sum_{i=1}^{N}(y_m(i)-C_s)^4 * y_m(i)\right) / \sigma_s^4 - 3$
Spectral Crest	$CR_s = (Max(y_m(i))\lvert_{i=1\ to\ N}) / C_s$
Irregularity-K	$IK_s = \sum_{i=2}^{N-1}\left\lvert y_m(i) - \frac{y_m(i-1)+y_m(i)+y_m(i+1)}{3}\right\rvert$
Irregularity-J	$IJ_s = \frac{\sum_{i=1}^{N-1}(y_m(i)-y_m(i+1))^2}{\sum_{i=1}^{N-1}(y_m(i))^2}$
Smoothness	$S_s = \sum_{i=2}^{N-1}\left\lvert 20.log(y_m(i)) - \frac{\left(20.log(y_m(i-1))+20.log(y_m(i))+20.log(y_m(i+1))\right)}{3}\right\rvert$
Flatness	$F_s = \left(\prod_{i=1}^{N} y_m(i)\right)^{\frac{1}{N}} / \left(\left(\sum_{i=1}^{N} y_m(i)\right)/N\right)$
Roll Off	$R_s = \frac{SampleRate}{N} * n\Big\rvert_{\sum_{i=1}^{n} y_m < Threshold}$

Formally, for a trace i, we denote $\mathcal{F}(I)_i$ and $\mathcal{F}(S)_i$ as the set of selected features of $I(k)$ and $S(k)$, respectively. The fingerprint of this trace is then represented by $< \mathcal{F}(I)_i, \mathcal{F}(S)_i >$.

C. Fingerprint Matching

`AccelPrint` uses supervised learning to classify smartphone accelerometers, beginning with a training phase followed by testing (or classification). During training, n traces from a smartphone are collected for extracting fingerprints, and the n sets of features $< \mathcal{F}(I)_i, \mathcal{F}(S)_i >_{i\in[1,n]}$ are used to train the classifier. For m smartphones, $n \times m$ sets of features can be used to train the classifier all together. In addition, given n set of features that constitute the fingerprint of a new smartphone, the classifier database can be updated to incorporate the new smartphone. We employ an ensemble classification approach for training mainly to achieve robustness over any single classification approach [12], [21], [42], [54]. Among various ensemble techniques possible, we use Bagged Decision Trees [13] for ensemble learning.

During the testing phase, `AccelPrint` collects a trace, extracts a set of features $< \mathcal{F}(I), \mathcal{F}(S) >$, and inputs to the classifier. The classifier either outputs a positive match with one of the phones that it has been trained with, or indicates an "alien", implying that this accelerometer is not from any of the phones used for training. In such a case, `AccelPrint` initiates a training request that collects n traces from the alien smartphone, inserts a new entry to the classifier database, and re-trains the system. Although false negatives could occur, additional side-information could be leveraged to exercise caution before re-training. For instance, enforcing the rule that the classifier can be re-trained only when the trace is the first one collected by an app since installation, could improve the confidence in re-training.

To distinguish an alien device from the known devices, we apply a threshold on the *classification score* – if the classification score is less than the threshold, then the trace is declared "alien". Figure 6 plots the classification scores for both alien and pre-registered phones (the first half of the X-axis are traces drawn from alien devices, and the vice versa). Observe that the alien phones generally present a relatively low score and a threshold for reliable segregation is not hard to find. In `AccelPrint`, we have picked the threshold to be 0.6.

V. Performance Evaluation

We have evaluated the performance of `AccelPrint` using 80 stand-alone accelerometer chips, 25 smartphones and 2 tablets. The key questions we investigated and the corresponding findings are summarized below.

- *How much training is needed to learn the fingerprint?* We find that 30 seconds of accelerometer trace is sufficient to model a device's fingerprint.
- *Does the fingerprint manifest only at the fastest sampling rate?* No, even at slower sampling rates, devices exhibit distinguishing features. However, the performance is slightly better at faster rates.

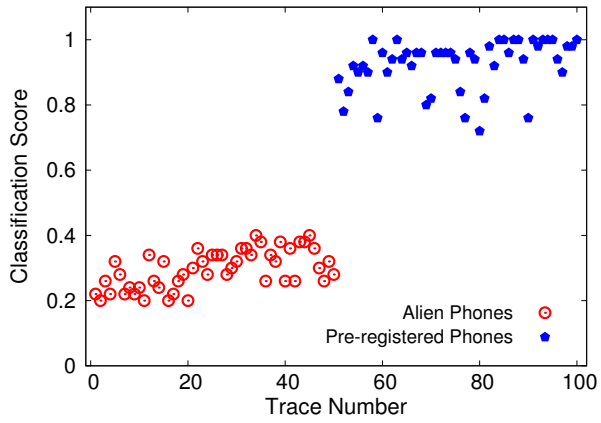

Fig. 6: The threshold for segregating alien phones can be chosen from a wide range, indicating robustness.

- *Does the system need to be aware of the surface on which a device is placed?* No. Whereas training on a variety of surfaces improves the performance, the system itself is surface-agnostic.
- *Can we mask the fingerprint of a device with a case?* The fingerprints of a device with and without a case are different. However, similar to surfaces, by training with and without a case, a device can be classified with high precision.
- *Is the system sensitive to CPU load?* Somewhat. If the difference in CPU load at the time of training and testing is less than 40%, it does not significantly affect the performance of `AccelPrint`.
- *Can we fingerprint a device without vibration?* Yes. Even when the devices are kept still, their accelerometers yield somewhat distinct readings allowing them to be fingerprinted, albiet with lower precision. However, a device can be fingerprinted with rotational motion as well as that with vibration.

We now begin by describing the experimental setup and the performance metrics used for evaluation.

A. Experimental Setup

We have conducted experiments with 80 accelerometer chips of 3 types, 25 Android phones of 5 different models, and 2 tablets. We collect 50 accelerometer traces for each of the 80 chips and 27 phones/tablets, a total of 5350 traces. [1] The stand-alone accelerometer chip setup is described below.

1) Stand-alone Accelerometer Chip Setup: We have conducted experiments with 3 types of accelerometer chips, 60 MPU-6050, 10 ADXL-345, and 10 MMA-8425q. We prepare a setup to collect data from all stand-alone accelerometer chips under the same environment. The setup has three main components: 1) the breadboard that holds the accelerometer chips and the vibrator motor, 2) the Arduino that is connected to various components on the breadboard with wires and 3) the server that stores the collected data and it is connected to the Arduino with a USB cable.

Arduino has an 8-bit RISC-based microcontroller [9] as its main component and provides 14 input/output pins to connect to the external circuits. We control the on-board vibration motor with Arduino so that it will be turned on and off periodically. In addition, the Arduino can connect to a server with a USB interface, and communicate with the stand-alone chips via serial communication. Thus, Arduino helps transfer data from the stand-alone chips to the server.

We used a breadboard to arrange the accelerometer chip and the vibrator motor as an evaluation unit. The connections between various components on the breadboard and the Arduino are made with jump wires. We also attached a small piece of acrylic board under the breadboard to make the weight of this evaluation unit comparable to that of a smartphone. The on-board vibrator motor generated stimulation to the accelerometer chip. We chose a lightweight vibrator motor similar to the model used in a majority of our experimental smartphones. This motor was firmly soldered to the breadboard and connected to one of the digital pins of the Arduino board. The only removable component in this setup was the accelerometer chip. Because the accelerometer is tiny and difficult to operate, we procured the chips mounted on breakout boards [25] so that we can access the pins of the chip through the header pins soldered with the breakout board. This setup is shown in Fig. 3.

2) Smartphone Setup: We have experimented with 25 Android phones of five different models and 2 tablets: i) 8 Nexus One; ii) 7 Samsung Galaxy Nexus; iii) 6 Samsung Galaxy S3 iv) 2 Nexus S; v) 1 HTC Incredible Two; and vi) 1 HTC MyTouch vii) 2 Samsung Galaxy Tab 2. The sampling mode is set to "Fastest", and to stimulate the accelerometer, we use internal vibration motor of the devices.

3) Data Collection Setup: Using each of the chips and devices, we have conducted experiments in our lab to gather sensor readings. Either the internal vibration motors or the stand-alone motor is used to stimulate the accelerometer for 2 seconds and the accelerometer readings are recorded with the sampling mode set to "Fastest" by default. We refer to this 2 seconds of accelerometer data as *trace*.

To exclude any possibility that the fingerprints may be an outcome of a unique physical arrangement of the motor and stand-alone chips, we collect traces in a round robin manner. In the first round, we collect 10 traces from each chip and smartphone. Once 10 traces from each device are collected, then we move to the next round and similarly collect 10 traces from each device. Thus, in total, it takes 5 rounds to collect 50 traces from each and every device.

[1] The raw data and the source code for this paper can be found at http://web.engr.illinois.edu/~sdey4/AccelPrintDataSourceCode.html.

To obtain statistically significant results, we evaluate our system under each setting 10 times. Unless specified, each time we randomly chose 15 traces out of 50 traces as our training sample set, and used the rest as the testing sample set. Thus, in total 3745 randomly chosen traces were used for testing and 1605 traces were used for training each time.

B. Performance Metrics

Let k be the total number of devices or classes. Given an accelerometer trace, `AccelPrint` classifies it as belonging to one of these classes. Then, based on the ground truth, for each class i, we define TP_i as the true positives for class i, i.e., the number of traces that are correctly classified as i. Similarly, FN_i, and FP_i, respectively refer to the number of traces that are wrongly rejected, and wrongly classified as i. Now, we can define the standard multi-class classification metrics, *precision*, and *recall*, as follows.

$$\text{precision}_i = \frac{(TP_i)}{(TP_i + FP_i)}$$

$$\text{recall}_i = \frac{(TP_i)}{(TP_i + FN_i)}$$

Then, we can compute the average precision and recall.

$$\text{average precision} = \frac{\sum_{i=1}^{k} \text{precision}_i}{k}$$

$$\text{average recall} = \frac{\sum_{i=1}^{k} \text{recall}_i}{k}$$

To evaluate the overall performance of the multi-class classification in the presence of alien devices (untrained devices), we use accuracy as the metric. Given that a multi-class classifier is trained by n classes and is tested by n classes and m aliens, we define accuracy as below.

$$\text{accuracy} = \frac{(\sum_{i=1}^{n} TP_i + \sum_{j=1}^{m} TN_j)}{N}$$

where N is the total number of testing traces, TP_i is the true positive for class i and TN_j as the true negative for alien class j, i.e., alien class j being rejected by the classifier.

C. Overall Performance

In the first set of experiments, we trained the system with 15 traces and tested it with the rest 35 traces from each of the 107 chips/phones/tablets. The resulting average classification score for each device is shown on a heat map (aka. a confusion matrix) in Figure 7. In the confusion matrix plot, the darker the shade, the higher the classification score. Evidently, the diagonal cells are the darkest, implying that the traces from device i were indeed classified as class i. While there are a few gray cells appearing outside the diagonal cells, instances of misclassification are rare. This is because the classifier picks the device with the maximum score, as long as the score is greater than a predefined threshold (used for segregating alien phones).

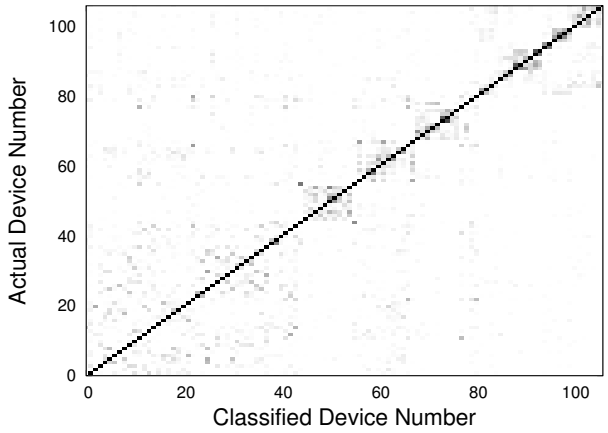

Fig. 7: Overall performance: confusion matrix.

The confusion matrix contains some clusters, e.g, a cluster of devices numbered from 0 to 49. Note that the device numbers 0 to 79 represent the standalone chips whereas the device numbers 80 to 104 map to the 25 smartphones, and the last two devices correspond to the two tablets used in this experiment. In particular, the devices numbered from 0 to 49 represent the standalone chips of the same brand, the same model, and the same manufacturing company (i.e., MPU 6050). Therefore, they are more likely to be confused with each other than another device. Still, the following results show that a device can be identified with high precision.

Zooming into the results, we compute the precision_i and recall_i for each class i, and plot the CDF of their distributions in Figures 8a, 8b, 9a and 9b for chips and smartphones. Even with 5 training traces (amounting to 10 seconds of training) for both chips and smartphones, both precision and recall are above 75% for all classes. The tail for both precision and recall gets shortened as the training size increases to 10, with none of the classes having precision below 85%. This is true for the smartphones as well as the MEMS chips. With training size 15, the worst case precision improves to 87%, while the average precision and recall are both above 99%. Since 30 seconds of training traces are not unreasonable (a malware in a phone could silently collect data from time to time and accumulate up to 30 second traces when an incoming call rings the phone), we set the number of training traces to 15 in the rest of the evaluation.

Next, we consider several factors that could affect our ability to model fingerprints and classify devices. The rate at which an app samples the accelerometer readings depends on the configured mode as well as the CPU load. The surface on which the phone is placed may influence the vibration sensed by the accelerometer. In view of operations in uncontrolled environments, we study the impact of these factors on the fingerprint classification performance.

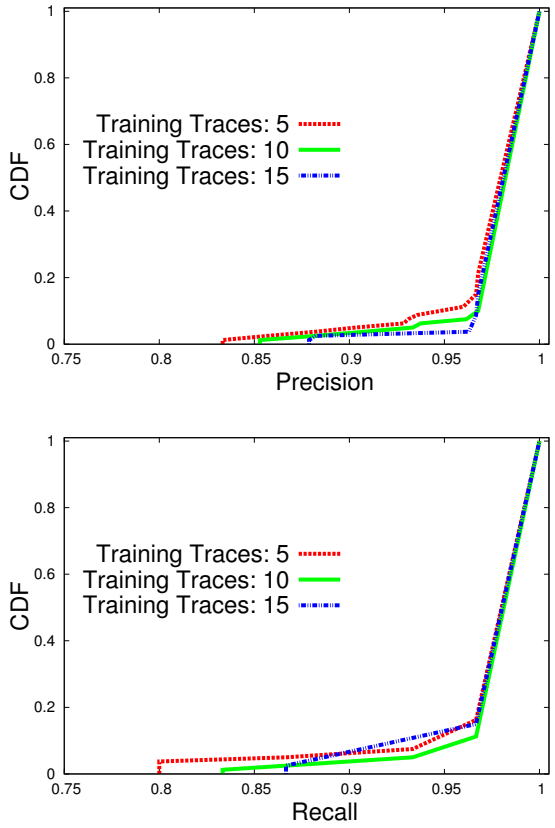

Fig. 8: Overall performance for chips: (a) precision; (b) recall.

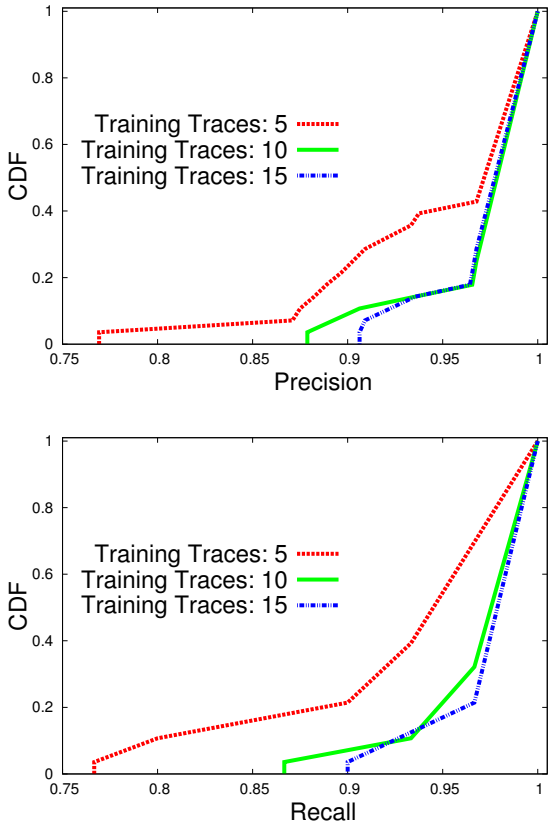

Fig. 9: Overall performance for smartphones: (a) precision; (b) recall.

D. Significance of Sampling Rate

The Android OS allows four different sampling rates for the accelerometer. These with decreasing rates are: i) Fastest; ii) Game; iii) UI; and iv) Normal. The sampling rate of the Fastest mode on our devices varies from around 100 Hz to 20 Hz, depending upon the hardware/software specification and the activity level. However, the rate of the Normal mode remains the same for all the devices (around 4 Hz).

To study the effect of sampling rates on fingerprinting, we conduct experiments with each of the four modes. The results of these experiments are shown in Fig. 10, from which we observed that the faster the sampling rate is, the higher the precision and recall are. Nevertheless, even at the slowest rate (i.e., the Normal mode at 4 Hz), precision and recall are both above 80%. This indicates that with only 4 accelerometer samples per second, different devices can be distinguished with reasonable amount of precision. Of course, a faster sampling rate does improve the likelihood to distinguish devices, and subtle differences between accelerometers can be discerned with a higher precision.

E. Impact of CPU Load

To understand the impact of CPU load on the fingerprints of devices, we create a background process using Android

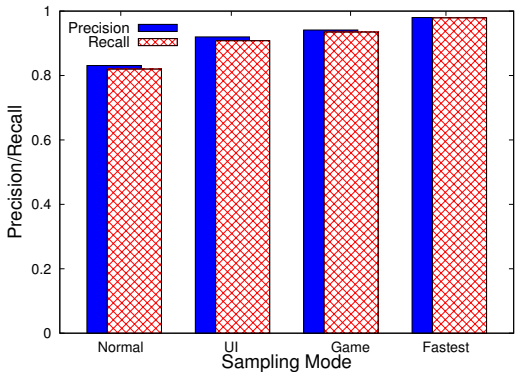

Fig. 10: Performance with different sampling rates

IntentService class to control the CPU load and measure its effect on `AccelPrint`. The background process works in a duty cycle and alternates between awaken and sleeping status. For the ease of discussion, we define the percentage of time that the background service remains awake as "load level". To measure the impact of load, we first train our system with 0 load level. Then we test it with traces collected at four different load levels (0%, 20%, 40% and 60%). Similarly, we trained the classifier with traces collected at 20%, 40% and 60% load levels, respectively, and tested on the rest of load levels. We depicted the precision of `AccelPrint` in all training-testing scenarios as a heat diagram (shown

in Figure 11). In the heat diagram, the darker the region is, the higher the precision is. We observed that when we trained and tested the system with the traces collected at the same CPU load (diagonal region), we achieve a high precision, whereas as we increased the load level difference between train and test cases, the precision reduced. This is because at higher loads, some of the accelerometer readings get skipped, yielding different set of features. Nevertheless, the precision is above 80% when the load difference is within 40%. Overall, these results show that when the difference of load levels at the time of training and testing is similar, `AccelPrint` can distinguish devices with a high precision. That is, modest load difference does not significantly affect the performance of `AccelPrint`.

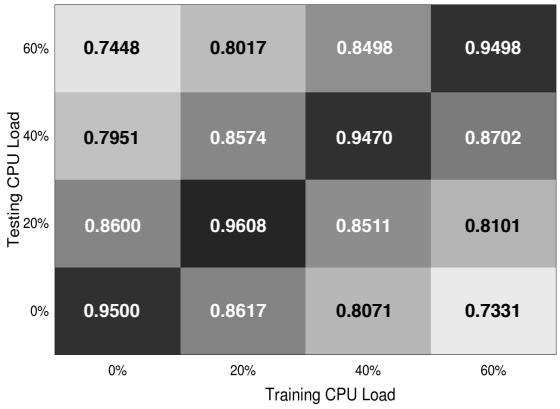

Fig. 11: Precision with varying CPU loads

F. Impact of Smartphone Casing

People typically use cases for phones and hence it is pertinent to understand how a smartphone's case affects its accelerometer's response to vibration. Commonly used cases include two types: the hard covers made of plastic and the soft covers made of rubber/leather. We conducted experiments using both types of covers and collected accelerometer readings of phones with and without those covers. The results of the experiments are shown in Figure 12.

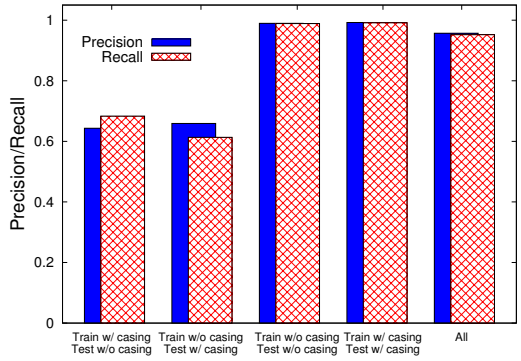

Fig. 12: Performance with and without phone cases

When the training is done using the traces of the phones without cases and then the testing is done with the phones with cases, and vice versa, the precision and recall is below 80%. Thus, a phone's case does influence its accelerometer's response to vibration and change the fingerprints of the phone. However, when the system is trained and tested on traces that were collected on the phones with a case, `AccelPrint`'s performance is not affected, which means having the case itself did not affect the classification of a phone. Furthermore, when the system is trained on a mix of traces that were collected on phones with and without cases, `AccelPrint` can classified all phones with high precision and recall. Considering that people do not change their phone's case often, the fingerprint of its accelerometer remains the same and can be utilized to identify the phone.

G. Impact of Surface

The amount of stimulation generated by a smartphone's vibration motor may depend upon the surface on which it is placed. To measure the impact of surface on device fingerprints, we collected accelerometer readings while keeping smartphones on four types of surfaces: a wooden table, a carpeted floor, a sofa cushion and on top of a palm. We trained `AccelPrint` with traces from one surface and tested with traces from all surfaces. We repeated this process for all four types of surfaces. We have also trained the system with a mix of traces that were collected on all four surfaces (we kept the number of training traces for each surface the same), and tested upon traces collected on all surface traces. The results of these experiment are shown in Figure 13.

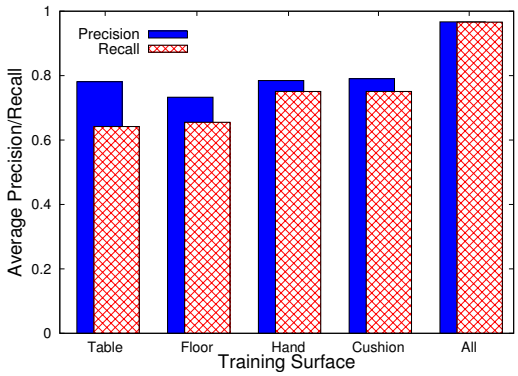

Fig. 13: Performance with different surfaces

When `AccelPrint` is trained by placing the phones on a table and then tested by placing them also on the carpet, on the cushion, and in a hand, it classifies all phones with an average precision of around 80% and recall close to 60%. The reduced precision and recall are caused by the different hardness of the surfaces. Compared to a table, a cushion is softer and can absorb a larger amount of vibration, and hence accelerometer readings are affected accordingly. If the training set includes enough diversity, e.g., when we train the system with traces from each surface, `AccelPrint` can classify each phone with 98% precision and recall,

regardless of the surface types. This is encouraging, since AccelPrint can achieve a high performance without having to explicitly specify the surface while testing a trace. In other words, AccelPrint is surface-agnostic.

H. Scalability of AccelPrint

In an academic lab setup, it is difficult to test a system with a very large set of devices. However, to get a sense of how well the system scales, we conduct an experiment where we increase the number of devices considered gradually and measure the performance of the system at each stage. In the first stage, we consider only 20 randomly chosen devices, train and evaluate the system with their traces. Next, we increase this number to 40 devices and we again evaluate our system. This way we keep increasing the number of devices in each stage and measure the system's performance.

Table III shows how accuracy changes with the increasing number of devices. From this table, we can observe that although the number of devices increases, the accuracy of the system does not change significantly. Figure 14 shows the precision and recall of the system for different number of devices considered. This figure also shows that the system performance does not change much for the larger set. These results provide encouraging signs that AccelPrint is likely scalable to a large number of devices.

TABLE III: Accuracy for increasing number of known devices

Number of devices	Accuracy
20	0.9917
40	0.9958
60	0.9956
80	0.9908
100	0.9883
107	0.9907

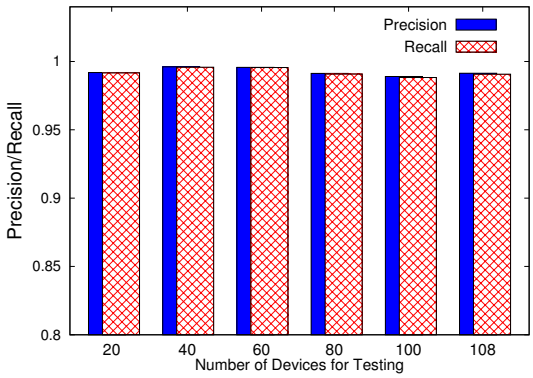

Fig. 14: Precision/Recall for known devices

I. Impact of Alien Devices

In real life, it is very likely that AccelPrint needs to classify the traces of the devices, i.e., alien devices, for which it is not trained beforehand. To understand how it performs for the traces collected from alien devices, we conduct the following experiment. Out of 107 devices, we randomly choose 20 devices for training and those 20 devices are never used for testing. Out of the rest 87 devices, first we randomly choose 20 alien devices for testing, and evaluate the performance of the system. In the next stage, we choose total 40 alien devices and repeat the same process. Thus, we include all the alien devices gradually and obtain a measure of the scalability of AccelPrint.

Table IV show the overall accuracy with the increasing number of alien devices, whereas figure 15 shows the precision and recall for the same. These results indicate that AccelPrint can successfully reject the alien devices using the threshold value of the classification score. Further, the overall performance of the system does not change much with the increasing number of alien devices.

TABLE IV: Accuracy for increasing alien devices

Number of devices	Accuracy
20	0.9917
40	0.9950
60	0.9983
80	0.9963
87	0.9883

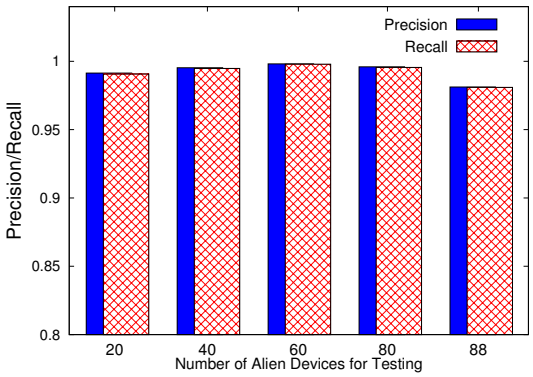

Fig. 15: Precision/Recall for alien devices

J. Impact of Stimulation

The experiments we described thus far employ vibration to stimulate the accelerometer. So it is natural to raise three questions. 1) Are we fingerprinting the accelerometer or the combination of accelerometer and vibration motor? 2) Can a device be fingerprinted with a different stimulation other than vibration? 3) How about fingerprinting a device without any stimulation? In this section, we attempt to answer these

questions. First, we subject smartphones to rotational motion and show that it works as well as vibration in fingerprinting devices. This experiment, in addition to the experiments with stand-alone accelerometer chips, confirm that accelerometer can be fingerprinted (even if vibration motor contributes to the fingerprint of a smartphone). Second, we perform experiments keeping devices still and show that they can be discriminated to some extent without any stimulation.

1) Fingerprinting with external rotational motor: In this setup, we use an external DC motor and an Arduino controller [2] to stimulate the smartphones through rotational motion. The Arduino controller rotates the DC motor in the same 2-sec pattern as above to maintain consistency. The DC motor is attached to a platform that is used to hold the smartphone. The entire setup is shown in Figure 16. In this experiment we use only the smartphones as the size of the platform was not suitable for the tablets. Here as we use only one common motor for all the smartphones, the rotational motor has no impact on the fingerprint of the smartphones. However this is worth mentioning that we also collect traces from this setup in a round robin fashion as stated previously.

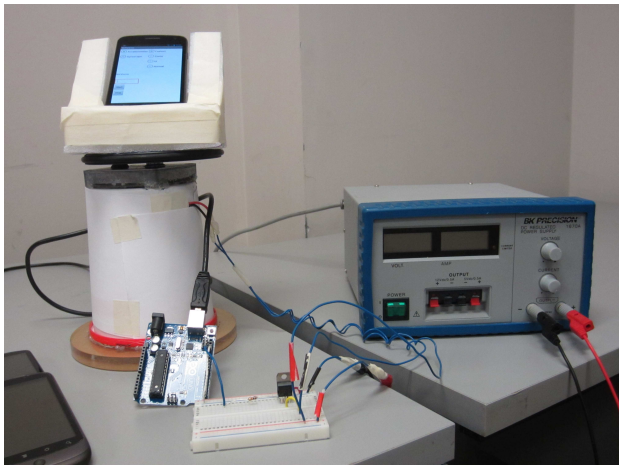

Fig. 16: Smartphone with external motor for rotation.

The precision and recall of this setup is shown in Fig. 17. We compute the precision and recall for all the smartphones individually and then plot their CDF in Figures 17a and 17b. Here for 5 training traces (amounting to 10 seconds of training), both precision and recall are above 84% for all classes. As we increase the number of training traces, the tail for both CDFs get shortened. If we consider 15 training traces, then all the smartphones are above 90%. For training size 15, the average precision and recall are both above 97%.

2) Fingerprinting without stimulation: To understand if the fingerprint can be extracted without any stimulation, we conduct an experiment where we put the smartphones still on the table and collect the traces in this setup.

The precision and recall of this setup is shown in Fig. 18. As done in the previous section, here also we compute the precision and recall for all the smartphones individually to plot the CDF of their distributions in Figures 18a and 18b. Here for 5 training traces, both precision and recall are above 65% for all classes. Here even if we increase the number of training traces, the tail for both CDFs do not get shortened as fast as in Figure 9. We surmise this because accelerometer chip has a mechanical part and an electronic part. Without stimulation, the movable mechanical part does not play any role in the fingerprint and we cannot capture the imperfection of the movable part. That is why the system performs worse compared to that with external stimulation.

To summarize, our evaluation using 107 different types of stand-alone chips, smartphones, and tablets shows that they can be identified robustly leveraging the fingerprints of their accelerometers. While even larger study is needed to confirm the scalability of our findings, to the best of our knowledge, this is the first work to attempt device identification based on fingerprints of accelerometers.

VI. Related Work

A. Device Fingerprinting

Fingerprints are originally used as a biometrics technology to identify human beings [58], [62]. The concept was applied to device identification as early as in 1960s, when a "specific emitter identification" system that utilizes externally observable characteristics of signals was developed to distinguish radars [61]. Later, the similar technology was used to identify transmitters in cellular networks [26], [39], [56]. Since then, much effort has been devoted to identifying network devices by building a fingerprint out of their software or hardware.

In terms of software-based fingerprint, MAC address was exploited to detect the presence of multiple 802.11 devices [32], [66]. The combination of chipsets, firmware and device drivers [30], timing interval of probe request frames [19], or the patterns of wireless traffic [50] were also used to identify devices. An open source toolkit for network administrators [40] utilizes software configuration for network discovery. The downside of these methods is that fingerprints will be different once computer configuration or traffic behavior changes.

Another approach of the software based fingerprinting is to exploit the browser properties and plugins to figure out unique identifiers [23]. Researches [48], [68] shows that the log files of Bing and Hotmail and web browser history can potentially reveal the identity of the client. The web viewing time [31] or the benchmark execution time of the javascript [46] can also help to fingerprint a device.

Hardware-based approaches rely on stable fingerprints. Network devices have different clock oscillators that create stable and constant clock skews [53], [60], which can be estimated remotely using TCP and ICMP timestamps for device fingerprinting [37]. Radio frequency (RF) fingerprinting

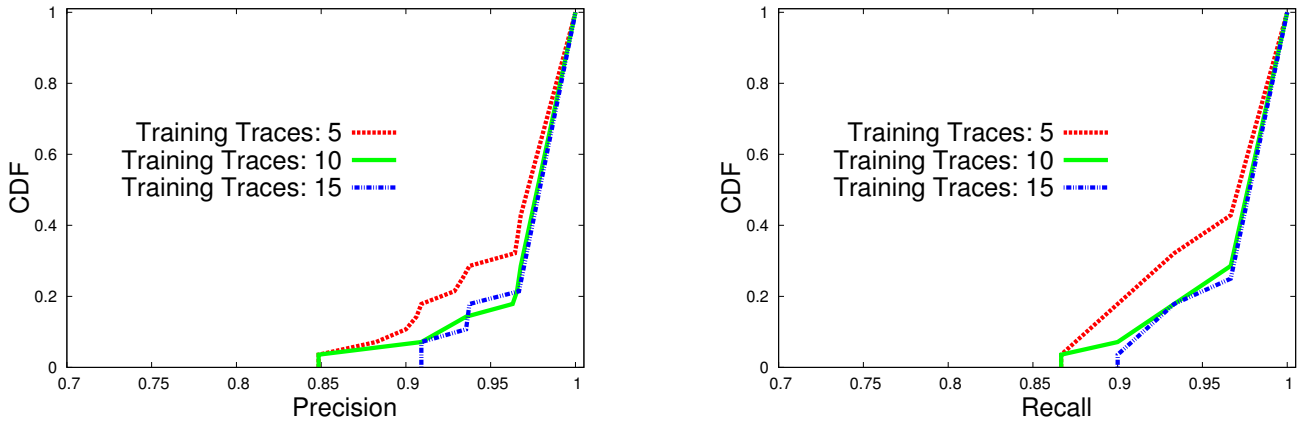

Fig. 17: Precision and recall of smartphones with external motor for rotation

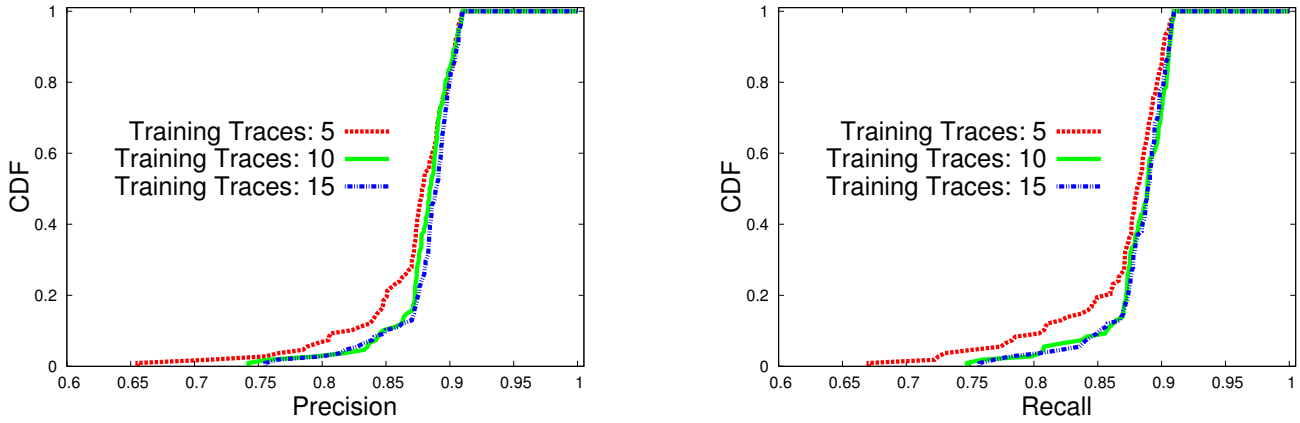

Fig. 18: Precision and recall of smartphones without any stimulation

has been extensively studied to identify wireless transmitters and can be divided into two categories: channel-based and device-based. Channel-based methods estimate the channel impulse response that characterizes multipath effect [51] and attenuation [29], [70] between a transmitter and a receiver for RF fingerprinting. Device-based methods rely on the distinct radiometrics of transmitters at the waveform [33]–[35], [41], [57], [63] or modulation [64] levels. Wired Ethernet NICs can also be identified by analyzing their analog signals [55].

Our work is inspired by the aforementioned device fingerprinting work. Instead of wireless or wired transmitters, we focus on fingerprinting smartphones utilizing the imperfections of on-board sensors.

B. Privacy and Side Channel

Sensor-rich smartphones and tablets are increasingly becoming the target of attacks for harvesting sensitive data [24]. Enck et al. [27], [28] showed the potential misuse of users' private information through third-party applications, and Schlegel et al. [59] demonstrated that a smartphone's microphone can be used maliciously to retrieve sensitive data.

Since Cai et al. pointed out that smartphones built-in sensors (e.g., GPS, microphone and camera) can be used as a side channel to record user actions by stealthily sniffing on them [17], several systems (e.g., TouchLogger [15], ACCessory [49], Taplogger [67]) have been built. They have shown that collecting data from an accelerometer or a gyroscope alone is enough to infer the sequences of touches on a soft keyboard. Cai et al. [16] compared gyroscopes and accelerometers as a side channel for inferring numeric and soft-keyboard inputs. They found that inference based on the gyroscope is more accurate than the accelerometer. Milluzo et al. went one step ahead to develop TapPrint [45] that uses gyroscopic and accelerometer reading in combination to infer the location of tapping on tablet and smartphone keyboards. In addition, it was shown that accelerometer readings can be used to infer not only PINs but also

Android's graphical password patterns [11].

Inferring keystrokes on a regular keyboard has attracted much attention. Electromagnetic waves [65], acoustic signals [71], timing events [38], and specialized software [36] were exploited to intercept the keystrokes with high accuracy. It is also possible to infer keystrokes using the accelerometer readings from an iPhone placed two inches away from the keyboard.

Instead of treating sensors as a side channel, we focus on the built-in fingerprint of a smartphone for device identification.

VII. Limitations and Discussion

(1) Scalability. Accelerometer fingerprints may not need to be globally unique to pose a threat. For instance, if a smartphone accelerometer in the US proves to be identical to another in Taiwan, the backend adversary may still be able to disambiguate using the device's cell tower location. Put differently, broad location, device type, and other contextual factors can relax the stringency on uniqueness. Moreover, combining additional sensors within the fingerprint, such as the gyroscope and the microphone, can further increase the ability to discriminate. From crude measurements, we have observed that the gyroscope also responds to stimuli from the phone's vibration motor. For the microphone, it may be feasible to play a fixed audio file through the speakers, and the recording processed for the fingerprint.

(2) Scrubbing the Fingerprint. In an attempt to scrub the fingerprint, we first attempted to compute the resting acceleration of each device, i.e., the acceleration value when the phone is completely at rest on a pre-defined location. Given that the resting values are different across phones, we equalized the RSS values by suitably adding or subtracting from the signal. Still, the fingerprinting accuracy did not degrade since the uniqueness probably arose from a wide range of features. Equalizing across all these features is certainly difficult. Alternatively, we added 0dB white Gaussian noise to the signal, but observed only a marginal drop in precision and recall (to 93%). Upon adding 5dB of noise, the performance dropped sharply, but other higher level operations were also affected severely. Finally, we used a low pass filter to eliminate the high-frequency components of the signal, but again was not able to remove the fingerprint without affecting the application. We opine that fingerprint scrubbing requires closer investigation, and will be a critical next step to `AccelPrint`.

(3) Influence of the version of Operating Systems. We have used the Android operating system (ice cream sandwich and gingerbread) for all the smartphones. Between all phones using the identical OS version, the fingerprints are still discernible, implying that `AccelPrint` is not affected by the OS versions.

VIII. Conclusion

This paper shows that accelerometers possess unique fingerprints. As standard components inside smartphones and tablets, accelerometers' fingerprints create new threats in mobile apps — tracking users without cookies or device IDs. The fingerprints stem from the core of accelerometers: an electro mechanical moving part holds the key to sensing. The manufacturing of such moving parts are susceptible to imperfections, bringing about diversity in the behavior of accelerometers. This diversity is not conspicuous from a higher level since various operations such as step-counts and display rotations are tolerant to noise. However, when the properties of these imperfections are deliberately extracted, they lead to a sensor fingerprint, adequate to identify a device, and even an user. Our results on 80 standalone accelerometer chips, 25 Android phones, and 2 tablets offer confidence that such fingerprints exist, and are visible even in real, uncontrolled environments. While commercial-grade measurements are necessary towards a conclusive result, we believe that our lab findings are still an early and important step for understanding sensor fingerprints and their consequences at large.

Acknowledgment

This work has been funded in part by NSF CNS-0845671 and NSF GEO-1124657.

References

[1] "Adxl-345 3-axis digital accelerometer," http://www.analog.com/static/imported-files/data_sheets/ADXL345.pdf.

[2] "Arduino unosetup," http://www.arduino.cc/en/Main/ArduinoBoardUno/.

[3] "LibXtract: Feature Extraction Library Documentation," http://libxtract.sourceforge.net//.

[4] "Mma-8452q 3-axis digital acelerometer," http://www.freescale.com/files/sensors/doc/data_sheet/MMA8452Q.pdf.

[5] "MPU6050: Triple Axis Accelerometer and Gyroscope," http://www.invensense.com/mems/gyro/documents/PS-MPU-6000A.pdf.

[6] "Apple updates ios to 6.1," http://www.apple.com/pr/library/2013/01/28Apple-Updates-iOS-to-6-1.html, 2013.

[7] adam pocock and gavin brown, "Feast," 2012, http://www.mloss.org/software/view/386/.

[8] M. Andrejasic, "Mems accelerometers," *Seminar*, March 2008.

[9] Atmel, "Atmega328 microcontroller," http://www.atmel.com/devices/atmega328.aspx.

[10] 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*, ser. ACSAC '12, 2012, pp. 41–50.

[11] ——, "Practicality of accelerometer side channels on smartphones," in *Proceedings of the 28th Annual Computer Security Applications Conference*. ACM, 2012, pp. 41–50.

[12] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 1049–1060, 2001.

[13] ——, "Bagging predictors," in *Machine Learning*, 1996, pp. 123–140.

[14] G. Brown, A. Pocock, M.-J. Zhao, and M. Lujan, "Conditional likelihood maximisation: A unifying framework for information theoretic feature selection," *The Journal of Machine Learning Research*, vol. 13, pp. 27–66, 2012.

[15] L. Cai and H. Chen, "Touchlogger: inferring keystrokes on touch screen from smartphone motion," in *Proceedings of the 6th USENIX conference on Hot topics in security*, 2011.

[16] ——, "On the practicality of motion based keystroke inference attack," in *Trust and Trustworthy Computing*. Springer, 2012, pp. 273–290.

[17] L. Cai, S. Machiraju, and H. Chen, "Defending against sensor-sniffing attacks on mobile phones," in *Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds*. ACM, 2009, pp. 31–36.

[18] E. P. . Council, "Directive 2002/58 on privacy and electronic communications," http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2002:201:0037:0047:EN:PDF, 2011.

[19] L. C. C. Desmond, C. C. Yuan, T. C. Pheng, and R. S. Lee, "Identifying unique devices through wireless fingerprinting," in *Proceedings of ACM WiSec*, 2008.

[20] T. Dietterich, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[21] ——, "Ensemble methods in machine learning," *Multiple Classifier Systems*, vol. 45, no. 1, pp. 1–15, January 2000.

[22] C. T. Dr. Craig Hillman, "Manufacturing and reliability challenges with qfn," *SMTA DC Chapter*, vol. 45, no. 1, pp. 1049–1060, February 2009.

[23] P. Eckersley, "How unique is your web browser?" in *Privacy Enhancing Technologies*. Springer, 2010, pp. 1–18.

[24] M. Egele, C. Kruegel, E. Kirda, and G. Vigna, "PiOS: Detecting privacy leaks in iOS applications," in *Proceedings of the Network and Distributed System Security Symposium*, 2011.

[25] S. Electronics, "Mpu6050 breakout board," https://www.sparkfun.com/products/11028.

[26] M. Electronix, "TxID Transmitter FingerPrinter," http://www.motron.com/TransmitterID.html/.

[27] W. Enck, P. Gilbert, 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," in *Proceedings of USENIX OSDI*, 2010, pp. 1–6.

[28] W. Enck, D. Octeau, P. McDaniel, and S. Chaudhuri, "A study of android application security," in *Proceedings of the USENIX security symposium*, 2011.

[29] D. Faria and D. Cheriton, "Detecting identity-based attacks in wireless networks using signalprints," in *Proceedings of ACM WiSec*, 2006, p. 43âĂŞ52.

[30] J. Franklin, D. McCoy, P. Tabriz, and V. Neagoe, "Passive data link layer 802.11 wireless device driver fingerprinting," in *USENIX Security, Vancouver, BC, Canada*, August 2006.

[31] J.-L. Gassee and F. Filloux, "Measuring time spent on a web page," http://www.mondaynote.com/2009/05/24/measuring-time-spent-on-a-web-page/.

[32] F. Guo and T. Chiueh, "Sequence number-based mac address spoof detection. recent advances in intrusion detection: 8th international symposium," in *8th International Symposium, Raid*, 2005,2006.

[33] J. Hall, "Detection of rogue devices in wireless networks," *PhD thesis*, 2006.

[34] A. Y. H.C. Choe, C.E. Poole and H. Szu, "Novel identification of intercepted signals from unknown radio transmitters," *SPIE*, vol. 2491, no. 504, 2003.

[35] M. B. J. Hall and E. Kranakis, "Radio frequency fingerprinting for intrusion detection in wirless networks," in *Defendable and Secure Computing*, 2005.

[36] K. S. Killourhy and R. A. Maxion, "Comparing anomaly-detection algorithms for keystroke dynamics," in *Proceedings of IEEE DSN*. IEEE, 2009, pp. 125–134.

[37] T. Kohno, A. Broido, and K. Claffy, "Remote physical device fingerprinting," in *IEEE Symposium on Security and Privacy, Washington, DC, USA*, September 2005.

[38] D. Kune and Y. Kim, "Timing attacks on pin input devices," in *Proceedings of the 17th ACM conference on Computer and communications security*. ACM, 2010, pp. 678–680.

[39] L. Langley, "Specific emitter identification (sei) and classical parameter fusion technology," in *Proceedings of WESCON*, 1993.

[40] G. Lyon, "Nmap network mapper," http://www.nmap.org./.

[41] J. H. M. Barbeau and E. Kranakis, "Detecting impersonation attacks in future wireless and mobile networks," in *Proceedings of MADNES*, 2006.

[42] A. T. M. Jahrer and R. Legenstein, "Combining a predictions for accurate recommender systems," in *Proceedings of ACM SIGKDD*, 2010, pp. 693–702.

[43] P. Marquardt, A. Verma, H. Carter, and P. Traynor, "(sp) iPhone: decoding vibrations from nearby keyboards using mobile phone accelerometers," in *Proceedings of ACM CCS*, 2011, pp. 551–562.

[44] S. McKinley and M. Levine, "Cubic spline interpolation," *College of the Redwoods*, vol. 45, no. 1, pp. 1049–1060, January 1998.

[45] E. Miluzzo, A. Varshavsky, S. Balakrishnan, and R. R. Choudhury, "Tapprints: your finger taps have fingerprints," in *Proceedings of ACM Mobisys*, 2012, pp. 323–336.

[46] K. Mowery, D. Bogenreif, S. Yilek, and H. Shacham, "Fingerprinting information in javascript implementations," in *Proceedings of Web*, vol. 2, 2011.

[47] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna, "Cookieless monster: Exploring the ecosystem of web-based device fingerprinting," *2013 IEEE Symposium on Security and Privacy*, vol. 0, pp. 541–555, 2013.

[48] L. Olejnik, C. Castelluccia, A. Janc *et al.*, "Why johnny can't browse in peace: On the uniqueness of web browsing history patterns," in *5th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs 2012)*, 2012.

[49] 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.

[50] J. Pang, B. Greenstein, R. Gummadi, S. Seshan, and D. Wetherall, "802.11 user fingerprinting," in *Proceedings of MobiCom*, 2007, pp. 99–110.

[51] N. Patwari and S. K. Kasera, "Robust location distinction using temporal link signatures," in *Proceedings of MobiCon*, 2007.

[52] I. Paul, "Google play store: 800,000 apps and overtake apple appstore!" http://www.rssphone.com/google-play-store-800000-apps-and-overtake-apple-appstore/, 2012.

[53] V. Paxson, "On calibrating measurements of packet transit times," in *SIGMETRICS*, 1998.

[54] G. R. Caruana, A. Niculescu-Mizil and A. Ksikes, "Ensemble selection from libraries of models," in *Twenty-first international conference on Machine Learning*. ICML, March 2004, p. 18.

[55] M. M. R. Gerdes, T. Daniels and S. Russell, "Device identification via analog signal fingerprinting: A matched filter approach," in *Proceedings of NDSS*, 2006.

[56] M. Reizenman, "Cellular security: better, but foes still lurk," in *Spectrum, IEEE*, 2000.

[57] K. Remley, C. Grosvenor, R. Johnk, D. Novotny, P. Hale, M. McKinley, A. Karygiannis, and E. Antonakakis, "Electromagnetic signatures of wlan cards and network security," in *ISSPIT*, 2005.

[58] A. Ross and A. Jain, "Information fusion in biometrics," *Pattern Recognition Letters*, vol. 24, no. 13, pp. 2115–2125, June 2003.

[59] R. Schlegel, K. Zhang, X. Zhou, M. Intwala, A. Kapadia, and X. Wang, "Soundcomber: A stealthy and context-aware sound trojan for smartphones," in *Proceedings of the 18th Annual Network and Distributed System Security Symposium (NDSS)*, 2011, pp. 17–33.

[60] D. T. Sue B. Moont, Paul Skelly, "Estimation and removal of clock skew from network delay measurements," in *Proceedings of InfoCom*, 1999.

[61] K. Talbot, P. Duley, and M. Hyatt, "Specic emitter identification and verification," in *Technology Review*, 2003.

[62] P. Tuyls and J. Goseling, "Capacity and examples of template-protecting biometric authentication systems," in *ECCV*, 2004.

[63] O. Ureten and N. Serinken, "Rf fingerprinting," *Electrical and Computer Engineering Canadian Journal*, vol. 32, no. 1, pp. 27–33, 2007.

[64] M. G. S. O. Vladimir Brik, Suman Banerjee, "Wireless device identification with radiometric signatures," in *Proceedings of Mobicom*, 2008.

[65] M. Vuagnoux and S. Pasini, "Compromising electromagnetic emanations of wired and wireless keyboards," in *Proceedings of USENIX security*, 2009, pp. 1–16.

[66] J. Wright, "Detecting wireless lan mac address spoofing," *White Paper*, 2003.

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

[68] T.-F. Yen, Y. Xie, F. Yu, R. P. Yu, and M. Abadi, "Host fingerprinting and tracking on the web: Privacy and security implications," in *Proceedings of NDSS*, 2012.

[69] H. Yin, D. Song, M. Egele, C. Kruegel, and E. Kirda, "Panorama: capturing system-wide information flow for malware detection and analysis," in *Proceedings of the 14th ACM conference on Computer and communications security*. ACM, 2007, pp. 116–127.

[70] R. M. Z. Li, W. Xu and W. Trappe, "Securing wireless systems via lower layer enforcements," in *Proceedings of ACM Wise*, 2006, pp. 33–42.

[71] L. Zhuang, F. Zhou, and J. D. Tygar, "Keyboard acoustic emanations revisited," in *Proceedings of ACM CCS*. ACM, 2005, pp. 373–382.