Leveraging Sensor Fingerprinting for Mobile Device Authentication

Thomas Hupperich, Henry Hosseini, and Thorsten Holz

Horst Görtz Institute for IT-Security (HGI), Ruhr-Universität Bochum, Germany
firstname.lastname@rub.de

Abstract. Device fingerprinting is a technique for identification and recognition of clients and widely used in practice for Web tracking and fraud prevention. While common systems depend on software attributes, sensor-based fingerprinting relies on hardware imperfections and thus opens up new possibilities for device authentication. Recent work focusses on accelerometers as easily accessible sensors of modern mobile devices. However, it has remained unclear if device recognition via sensor-based fingerprinting is feasible under real-world conditions.
In this paper, we analyze the effectiveness of a specialized feature set for sensor-based device fingerprinting and compare the results to feature-less fingerprinting techniques based on raw measurements. Furthermore, we evaluate other sensor types—like gravity and magnetic field sensors—as well as combinations of different sensors concerning their suitability for the purpose of device authentication. We demonstrate that combinations of different sensors yield precise device fingerprints when evaluating the approach on a real-world data set consisting of empirical measurement results obtained from almost 5,000 devices.

Keywords: Device Fingerprinting, Sensor Fingerprinting, Device Authentication

1 Introduction

Many providers of modern Web services aim to recognize the device a user accesses their services from. An emerging functionality is the detection whether a user has changed the device, e. g., owns a new smartphone. The main target of this is authentication of a user's hardware to detect malicious activity like account theft: If a user logs in from a device never used before, this might be a hint that the login credentials have been stolen and are abused for malicious purposes. If a user logs in from an authenticated device which is known to be the user's device, it is probably a legitimate login. Google+ already implements such a detection: If a group member performs a login from a device never seen before and this login is deemed suspicious, a security alert is raised resulting in an email to the group's administrator. Facebook keeps track of its users' devices and aims to associate all systems belonging to a single user. Hence, detecting whether a login is performed either from a known or a new device is essential for fraud detection and account theft. Authenticating a device—and consequently binding an action to a specific device—can be an important step to achieve this security goal.

For this purpose, often the browser is fingerprinted at login time. *Fingerprinting* describes the process of obtaining a set of characteristic attributes from a system and assembling them to features which can be used to recognized or identify unique systems among all others. This technique usually complements cookie-based recognition which has been state of the art for many years. In the course of browser fingerprinting, software attributes like user-agent and installed plugins are leveraged [1, 10, 18, 23]. Previous research found that software-based device fingerprinting performs reasonably well for highly customized commodity systems like desktop computers, mainly since the configurations of these devices vary significantly [8, 26]. In contrast, mobile devices like smartphones and tablets are highly standardized. Still, it is possible to gather characteristic attributes of such systems and even about its user using Web technologies only [15]. However, device fingerprinting is strongly dependent on software attributes.

For device authentication, the fingerprint should be as immutable as possible and thus it should be hardware-based. As cookies may be deleted and software can be changed, device authentication should not rely on these factors. A hardware-based fingerprint should stay the same if a users decides to use another browser or even installs a different operating system. A devices' sensors seem to be suitable for this purpose and offer essential advantages:

1. Sensors are easily accessible: accelerometers and gyroscope data can be obtained even via JavaScript without special permissions needed.
2. Sensors yield measurable hardware imperfections which can be leveraged for fingerprinting a device.
3. These imperfections are immune to most software changes.

Due to their manufacturing processes, hardware sensors exhibit imperfections which cause minimal yet measurable deviations between every single sensor [5]. Hence, several sensors provide distinguishable measurements for the same events, making them a suitable source for device fingerprinting. Dey et al. proposed a system called AccelPrint introducing a thorough feature set setting new standards for accelerometer fingerprinting [7]. However, mobile devices typically contain several sensors and an open challenge is to figure out which sensor (or combination of sensors) yields the best device fingerprint in practice. Furthermore, the performance of such sensor-based device fingerprinting techniques was only analyzed in lab settings so far, thus it remains unclear if these techniques could actually be applied in practice, e. g., for device authentication.

In this paper, we address these open research gaps and focus on two different aspects of sensor-based fingerprinting: First, we evaluate the features proposed by AccelPrint on a data set containing almost 5,000 devices. This data set includes more than eight million accelerometer events collected by an app we developed and enables us to review the performance of such an approach in real-world conditions: While the mathematical features introduced by Dey et al. enable device recognition based on accelerometer data under scientific/ideal circumstances in a lab, our goal is to shed light on how precise sensor-based fingerprinting can be in the real world and what limitations such an approach yields in practice. We also compare the recognition precision of the introduced features and the raw measurement data to determine whether there is a realistic need for these features. Second, we study other sensors available on modern devices (e. g., gyroscope and magnetic field sensors) and assess how device fingerprinting techniques can be improved by leveraging this information. We extend current research by investigating how the seven most common sensor types can be used for fingerprinting devices on a hardware level and empirically verify our proposed approach. Our analysis is based on five different machine learning algorithms and three data preparation processes in order to perform a comprehensive feasibility study. We evaluate the precision at which a unique device and a device model can be recognized.

In summary, we make the following contributions:

- We examine the performance and necessity of the state-of-the-art feature set for accelerometer fingerprinting on a large, real-world data set.
- We investigate how other kinds of sensors available on modern devices can extend hardware-based fingerprinting for the goal of device authentication.
- We show how sensor data from several sensors can be combined to achieve a better device recognition precision.

2 Sensor-based Device Authentication

In contrast to *user authentication* which aims to prove a user's identity, we target to confirm a specific device (e. g., a unique smartphone) with *device authentication*. The overall goal is to bind an action performed by a user to a specific system (device) which is used to perform this action. Hence, if a device is authenticated, one can be sure that a specific user action was performed using exactly this device.

Use cases include online banking, handling of suspicious logins, and password reset requests. If a user of an online platform has forgotten his password and requests a new one, he usually has to answer a security question. Instead of proving his identity with the knowledge of the answer, he could authenticate his device which would make this an authentication by possession. This also applies for suspicious logins: large Web service providers keep track of the devices which are used to access their services and consequently check if a user performs a login from a known or a never-seen-before system. If a login attempt seems suspicious, a user could authenticate his device to prove his identity. Another use case is online banking: In Europe and in several countries around the world, online banking transactions—no matter if Web-based or app-based—need to be confirmed via a **t**rans**a**ction **n**umber (TAN). Additionally to this established method, device authentication could be performed for crucial actions like transactions above a certain amount or voiding a lost credit card. This way, the bank can be sure from which device this action was performed.

In a practical attack, an attacker may get hold of an original SIM card or a replacement card of a victim's phone number and abuse it (e. g., for app-based banking as the phone number is commonly used as identifier). Implementing a hardware-based mechanism for device authentication may remedy this fault: Binding transactions to hardware—in this case a user's mobile device—enables the detection of such fraud attempts as the service provider is capable to recognize that the attacker's action is not performed on the user's device. With hardware-based device authentication, SIM card theft and spoofing may be detected before a crucial action can be carried out by an attacker.

In any case, the hardware of the device to be authenticated needs to be *fingerprinted* as relying on software fingerprints may not be robust enough for this purpose. We differentiate between two types of use cases:

1. Web context: The provider operates an online platform and fingerprint techniques are restricted to Web technologies like HTML5 and JavaScript.
2. App context: The provider has deployed an app for using the service. As such an application may possess more permissions than a browser, it is able to access more of the device's resources (namely its sensors) for fingerprinting.

Although device authentication is not user authentication, it may be used as a second factor for user authentication as it constitutes that a specific user *owns* a specific device. This can be used as second factor, e. g. besides a *knowledge*-based authentication like passwords.

2.1 Device Registration

In order to use a device's sensor fingerprint for authentication, it needs to be registered first. The provider obtains the fingerprint belonging to a device which is to be registered and stores it in the fingerprint database.

During this registration process, the device needs to stay still for some seconds. In this time, the sensors' manufacturing imperfections are measured resulting in the device's sensor fingerprint. These specific measuring errors are an inherence factor of the device. In contrast to knowledge and ownership/possession factors of authentication, one could refer to these hardware peculiarities as "biometrics of hardware", thus to be considered as authentication by inherence.

The registration procedure is crucial and needs to be secured against adversaries. An attacker could try to register a device for a targeted user account as legitimate user device and consequently authorize banking transactions or perform successful logins or password resets. Therefore, the registration of a new device must only be possible after a successful user authentication, e. g., login at a provider's website. For example, a user may login to his online banking account and register a new device which needs to be confirmed via email. Only when such a second channel is used, the registration process can be performed, so that an attacker is not able to register a device without the user's knowledge and confirmation. Hence, the registration should be on-demand only. Additionally, for banking scenarios the device registration could be confirmed by a device-independent TAN method to avoid malicious registrations.

2.2 Device Authentication

Once the registration is done, a provider is able to distinguish and recognize devices based on their sensor fingerprints. In practice, this additional authentication could be performed to authorize crucial transactions in online banking (e. g., transactions above a certain threshold) or password resets at online platforms. It could also be used to verify a login attempt which is considered suspicious by common methods to clarify whether it is a legitimate login or a possible attack.

In any of these cases, a user would have to let the phone lay still for a few seconds, e. g., by laying it on a table. Previous work has proven that sensor imperfections can be measured in a duration of less than 30 seconds [7]. During this time, the device's sensors are fingerprinted again by measuring their hardware imperfections. The fingerprint can then be checked against previously registered devices by the provider, resulting either in a match which represents a legitimate user action or a reject possibly indicating illicit behavior. Figure 1 illustrates this procedure.

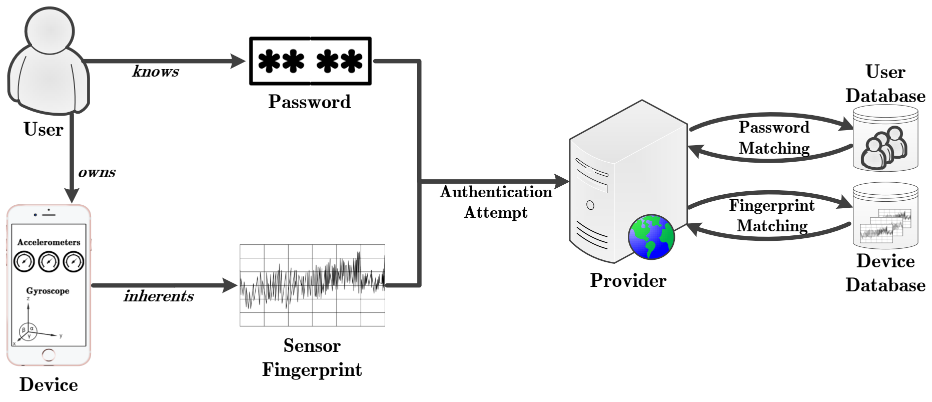

Fig. 1. Sensor-based device authentication for user authentication

In practice, if a device cannot be recognized exactly, instead of failing the authentication immediately there could be a fallback solution: The unique device may not be determined but at least the device model could be recognized from the sensor data. So, instead of being sure that a user performs an action from a specific device, at least information about the device type and model are available as enrichment for other fingerprint mechanisms. We included this scenario in our experiments as well.

As the sensor data is transferred to the provider during authentication, an attacker could try to replay a specific device fingerprint instead of her own to perform a successful authentication without the previously registered device. However, obtaining the victim's sensor fingerprint or even mimic the devices' sensors' peculiarity is hard to achieve in practice given that the sensor imperfections are hard to replicate. An attacker would either have to possess a special mobile device to intercept its sensor readings by manipulating system drivers or she has to setup a computer to simulate the targeted mobile device exactly. Consequently, if any other fingerprinting or system check is assembled by the provider, this has to be deceived as well, resulting in an increased effort for such a mimicry attack. Furthermore, sensor-based device authentication is an enhancement to other mechanisms and designed as reinforced user authentication. An attacker would still need to obtain user credentials or break other user authentication methods to successfully perform an attack.

3 Fingerprinting Sensors

Modern mobile devices contain a multitude of hardware sensors like accelerometers, gyroscopes and sensors for rotation, magnetic fields and gravity. Accelerometer and gyroscope sensor readings are usually accessible via JavaScript and therefore useful for Web-based fingerprinting and tracking. Although other sensors are accessible via native applications and may be accessible from within a Web browser in the future, recent research mostly addresses accelerometers [5,6, 14]. We investigate the effectiveness of the state-of-the-art features for accelerometer-based device recognition introduced by Dey et al. [7]. First, we compare the recognition precision utilizing these features to the recognition precision when using raw accelerometer data to provide insights on the usefulness of specialized features for device fingerprinting. Second, we extend current research by taking other sensor types into account to determine whether accelerometer-based results can be extrapolated. This includes common sensors of mobile devices as well as combinations of different sensors' data.

All these sensors exhibit hardware imperfections due to the manufacturing process which results in quivering measurement readings even for unmoved devices. These imprecisions usually affect the measurement value to a thousandth and are expected to be characteristic features for different sensors.

3.1 Data Set

The first step of our analysis is the preparation of a comprehensive data set of sensor measurements collected from a diverse set of mobile devices. We developed a sensor benchmarking app designed to collect raw sensor readings of accelerometers and other sensors from mobile devices in two stages: First, the user is instructed to put the mobile device on a flat table and leave it still to gather clean measurements for calibration. Second, the user is asked to turn the device in different directions, so we can collect readings when an actual interaction is happening. During both of these stages, the time window of each measurement is 2 seconds at the highest possible sampling frequency available, just like proposed by Dey et al. [7]. The app is available for Android and Blackberry phones and was distributed via the vendors' app stores. We made sure that users of the app were aware of the fact that they participated in a scientific study and that we collected information about the sensors in their mobile device. We did not store any personally identifiable information. Note that a user of the app is instructed to follow the two phases, but a user might not follow these instructions and thus the collected data might contain outliers or even wrong measurement readings. Therefore, significant movements have been detected as outliers and filtered out for our analyses. Minor movements may be included and represent real-world settings for device authentication as a user may have to authenticate her device being on the go.

We collected 41,610 benchmarks consisting of 58,280,607 raw measurement events in total from 7 different types of sensors and from nearly 5,000 devices. Every *event* yields a value for x, y and z axes coordinates as well as a timestamp to specify when the event was measured. A *benchmark* consists of all events which occurred within a 10 seconds time slot. Depending on the sensor type and model, there are different numbers of events per benchmark. The precise numbers of events, benchmarks, and devices per sensor are shown in Table 1. Although data from other sensors (e. g., proximity sensors) was collected by the app, only a minority of devices possess such sensors: Only a few benchmarks for these sensor types could be obtained and hence the data might not be substantive enough to make a claim about the recognition precision in general. For this reason, we take only those sensors into account having representative benchmark data available.

As different devices happen to integrate sensors manufactured by the same vendor, we show in Table 2 the number of different sensors in our data set. Taken from our representative data set, we see that there are many different vendors for general purpose accelerometers (275), but only a few for gravity sensors or linear acceleration sensors (each 37).

There is a unique identifier within the data set for every single device such that we can recognize specific devices as a ground truth. Furthermore, we store an identifier for every device model (e. g., "Google Nexus 5"). These two identifiers enable us to group the sensor measurement data by device model as well as by single devices. Hence, we are able to determine the effectiveness of fingerprinting features for recognizing device types (e. g., are hardware imperfections of iPhone 6 devices significantly different compared to hardware imperfections of Nexus 5 devices?) and for recognizing single devices (e. g., is it possible to tell one Nokia Lumia 930 apart from another?). In the following, we use the term *ModelID* to describe the identifier used to group data by device model and the term *DeviceID* for the identifier used to group data by single devices. For every sensor type in the data set, we group the data once per *DeviceID* and once per *ModelID*. For both groups, we compute the features described in Section 3.2 from the raw sensor readings obtained by our app. This builds up four data sets in total:

1. Raw sensor measurements grouped by *DeviceID* called $R_{DeviceID}$.
2. Feature set grouped by *DeviceID* defined as $F_{DeviceID}$.
3. Raw sensor measurements grouped by *ModelID* named $R_{ModelID}$.
4. Feature set grouped by *ModelID* which we define as $F_{ModelID}$.

Table 1. Numbers of events, benchmarks and devices per sensor type

Sensor Type	Events	Benchmarks	Devices
Acceleration	8,005,352	7,004	4,179
Magnetic Field	2,855,199	5,230	3,676
Orientation	8,047,497	6,228	4,963
Gyroscope	12,578,437	6,342	4,698
Gravity	9,061,253	5,726	4,374
Linear Acceleration	8,687,132	5,556	4,297
Rotation Vector	9,045,737	5,524	4,401

Table 2. Number of different sensor hardware models by sensor type

Sensor Type	No. of Sensor Models
Acceleration	275
Magnetic Field	179
Orientation	147
Gyroscope	100
Rotation Vector	43
Gravity	37
Linear Acceleration	37

3.2 Feature Set

As the second step of our analysis, we extract state-of-the-art features described below, originally proposed by Dey et al. [7], from the raw data records. We analyze if such features can be leveraged for other sensors as well and thus briefly introduce the feature set in the following.

Preliminary, we calculate the Root Sum Square (RSS) of the x, y, and z axes. Then, we extract the time domain features utilizing NumPy [33] and SciPy [17] libraries. In order to extract the frequency domain features, we have to transfer the raw sensor readings from time domain into frequency domain. For this purpose, we interpolated the RSS data. We applied a cubic spline interpolation as it addresses the accuracy of the minimal hardware deviations in our data set. For having less than 4 samples per measurement or lack of sensor readings from all three axes, 183 measurements had to be omitted during the interpolation phase. This might happen on hardware failure, broken sensors, or faulty drivers. After completing this task, we utilize the Fast Fourier Transformation (FFT) to transfer the interpolated measurements into the frequency domain. The frequency domain features are extracted from the transformed data. Finally, we vectorize the data to obtain sensor fingerprints utilizing the following features:

Time Domain Features

Mean is described as the result of dividing the sum of measurements to the number of samples in a specified time window: $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x(i)$

Standard Deviation describes how much the measurements deviate from the mean of all measurements in a specified time window. This feature provides the ability to consider noisy signals in our tests [32]: $\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x(i)-\bar{x})^2}$

Average Deviation provides the mean of the deviations of all samples in a specified time frame. By definition, only the absolute value of amplitudes are considered [32]: $D_{\bar{x}} = \frac{1}{N}\sum_{i=1}^{N}|x(i)-\bar{x}|$

Skewness measures the (lack of) symmetry of a distribution in a specified time frame. If the data set is symmetric, it looks even on the left and right side of the mean. Skewness can be positive or negative if the data set is more distributed to the left or right, respectively. The skewness of symmetric data is near zero [29]: $\gamma = \frac{1}{N}\sum_{i=1}^{N}(\frac{(x(i)-\bar{x})}{\sigma})^3$

Kurtosis states how much the data points are distributed near or far from the mean, i.e., whether a peak of data points exists near the mean or not. The ideal kurtosis is three according to our formula [29]: $\beta = \frac{1}{N}\sum_{i=1}^{N}(\frac{x(i)-\bar{x}}{\sigma})^4 - 3$

Root Mean Square (RMS) Amplitude measures the mean of all amplitudes over time. In order to calculate this feature, first all amplitudes are squared, so that both negative and positive values become positive. After calculating the mean of these values, they are scaled back to the right size by calculating the square root: $A = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x(i))^2}$ The RMS amplitude is normally equal to 70.7% of the peak amplitude [11].

Lowest Value is the smallest amount among the measurements in a specified time window: $L = Min(x(i))|_{i=1\ to\ N}$

Highest Value is the greatest value among the measurements in a specified time window: $H = Max(x(i))|_{i=1\ to\ N}$

Frequency Domain Features

Spectral Standard Deviation shows the spread of the frequencies in a spectrum relative to its mean along the frequency axis [13,28]: $\sigma_s = \sqrt{\frac{\sum_{i=1}^{N}(y_f(i))^2 * y_m(i)}{\sum_{i=1}^{N} y_m(i)}}$

Spectral Centroid can be considered as the middle point of the amplitude spectrum [3]: $\zeta_s = \frac{\sum_{i=1}^{N} y_f(i) y_m(i)}{\sum_{i=1}^{N} y_m(i)}$

Spectral Skewness measures the symmetry of the distribution of the spectral magnitude values relative to their mean [19,22,30]: $\gamma_s = \frac{\sum_{i=1}^{N}(y_m(i)-\zeta_s)^3 * y_m(i)}{\sigma_s^3}$

Spectral Kurtosis determines if the distribution of the spectral magnitude values contains non-Gaussian components [34]: $\beta_s = \frac{\sum_{i=1}^{N}(y_m(i)-\zeta_s)^4 * y_m(i)}{\sigma_s^4 - 3}$

Spectral Crest measures the peakiness of a spectrum and is inversely proportional to the flatness feature [36]: $CR_s = \frac{(Max(y_m(i))|_{i=1\ to\ N}}{\zeta_s}$

Irregularity-K measures the degree of variation of successive peaks in a spectrum. Irregularity-K refers to the definition of Krimphoff et al. [21] where irregularity is the sum of the amplitude minus the mean of the preceding, same and next amplitude: $IK_s = \sum_{i=2}^{N-1}\left|y_m(i) - \frac{y_m(i-1)+y_m(i)+y_m(i+1)}{3}\right|$

Irregularity-J measures the same as irregularity-K, but refers to the definition of Jensen [16] where irregularity is defined as the sum of squaring the differences in amplitude between adjoining partials [35]: $IJ_s = \frac{\sum_{i=1}^{N-1}(y_m(i)-y_m(i+1))^2}{\sum_{i=1}^{N-1}(y_m(i))^2}$

Smoothness measures the degree of differences between adjacent amplitudes [24, 27]: $S_s = \sum_{i=2}^{N-1} |20.log(y_m(i)) - \frac{(20.log(y_m(i-1))+20.log(y_m(i))+20.log(y_m(i+1)))}{3}$

Flatness measures the flatness of a spectrum and is inversely proportional to the spectral crest. The differences between spectral crest and flatness are in the less required computational power for spectral crest, but more accurate results in spectral flatness since not normalized signals have less influence on the result [9]: $F_s = \frac{\sqrt[N]{(\prod_{i=1}^{N} y_m(i))}}{\frac{1}{N}\sum_{i=1}^{N} y_m(i)}$

3.3 Classifier

In the third step of our analysis, we apply five field-tested classification algorithms to all data sets described in Section 3.1. We chose algorithms from different machine learning categories to address the model selection problem. Note that our scenario for device or model recognition does not pose a typical classification problem, as every device and model which needs to be "classified" has been seen during the training phase. This circumstance is more likely related to matching problems. We evaluate the following five classification and ensemble methods in our experiments:

- **k-NN**: The k-Nearest-Neighbor classifier is a basic ML algorithm. We chose $k = 1$ as this correlates with the fact that we want to achieve a matching.
- **SVM**: As Support Vector Machines are designed to handle numeric values, they are naturally suitable for processing our data set.
- **Bagging Tree**: This classifier has been used originally by Dey et al. [7]. In order to evaluate the effectiveness on the basis of real-world data, we must evaluate this classifier as well.
- **Random Forest**: The Random Forest classifier combines the merits of Bagging Tree and a random selection of features. This method remedies tendencies of overfitting.
- **Extra Trees**: Extra Trees is an averaging ensemble method known for a high prediction accuracy. The drawback is that it usually grows bigger than Random Forests, especially on large data like our sensor measurements.

In every test, we split the existing data into a training set and a test set. The training set is used for cross validating the classifiers' parameters before creating a model and testing the test set. In the research of Guyon [12] and Amari et al. [2], the ratio of the test set to the training set is proposed to be inversely proportional to the square root of the number of features if the number of features is greater than one. For the 17 features described above, this means:

$$\frac{1}{\sqrt{\#features}} = \frac{1}{\sqrt{17}} \approx 0.243$$

Hence, we use a split of 75 % of the data for training and 25 % for testing.

Please note that we did not use the same machine learning classification models for recognition of device manufacturing models and for recognition of single devices. We conducted these experiments separately and trained classification models for the specific tasks. To perform a comprehensive analysis, we prepared each data set in three different ways for every experiment and applied the classifiers to the data (i) as-is, (ii) normalized and (iii) scaled.

Finally, we determine the maximum recognition precision of all *raw* classifications and *features* classification for each data set in order to clarify whether classification of models and devices can be performed better on raw data or the introduced features. Every test—from splitting the data set into training set and test set up to classification—has been performed three times and the mean of these repetitions is represented to mitigate "lucky strikes", which may occur when a data set is split randomly.

3.4 Formalization

In the following, we work with the four data sets $R_{ModelID}$, $F_{ModelID}$, $R_{DeviceID}$, and $F_{DeviceID}$ described in Section 3.1. Consequently, R represents all raw sensor measurements and F represents the features calculated on the basis of R. Hence, the feature set is derived as a function from raw sensor events: $F = f(R)$

The function f includes the steps for feature extraction described in Section 3.2 including calculations of Root Sum Square, interpolation and Fast Fourier Transformation. Consequently, F includes all features from time domain *and* frequency domain. Please keep in mind that every data set is split into a training subset and a test subset for subsequent machine learning procedures.

Every data record of these data sets consists of a data vector and a class attribute. The data vector D includes all attributes which are used for recognition by machine learning. For data vectors of the raw sensor measurements data sets, the single values are plain readings of the dimensions x, y and z provided by the sensors directly: $D_R = r_1, r_2, ..., r_n, n \in \mathbb{N}$. Consequently, for data vectors of the examined feature sets, every value represents a feature: $D_F = f_1, f_2, ..., f_n, n \in \mathbb{N}$. The class attribute c is derived from the chosen identifier which is either the *ModelID* or the *DeviceID*.

In order to calculate the recognition precision, we define a *match* as true positive. A match will be achieved if a data vector of a test set is related to a data vector with the same class c of the corresponding training set by the machine learning algorithm. A correct *reject* expresses a true negative, meaning that a non-trained device is not matched accidentally to a trained device. If a device which has been in the training set gets rejected while testing, it is a false negative while a non-trained device which is matched with a device from the training set poses a false positive.

Finally, we are able to define the recognition precision P for specific data sets and feature sets as

$$P_{S,M_{id}} = ML(Set_{Training}, Set_{Test}),$$

where S is a sensor type, $M \in \{R, F\}$, $id \in \{\mathit{ModelID}, \mathit{DeviceID}\}$, ML is the chosen machine learning algorithm and *trainingset* and *testset* are subsets corresponding to S and *id*. For instance, the recognition precision achieved by a Bagging Tree classifier (BT) for the data set of features grouped by *DeviceID* and based on gravity sensor data will be

$$P_{Gravity,F_{DeviceID}} = BT(train_{F_{DeviceID}}, test_{F_{DeviceID}}).$$

4 Evaluation

We conducted recognition experiments for every sensor type with each data set utilizing each classifier seeking for the best precision to use for device authentication. Hence, we present only the results of the best performing classifiers for each experiment. More specifically, for each sensor type and each data set we applied the algorithms described in Section 3.3, but for comparison we take the maximum recognition rate of all classifiers into account. Furthermore, we repeated every test with the data set three times using it (i) as-is, (ii) scaled and (iii) normalized to ensure to have the best preprocessing for every test. Again, we describe only the best results of all preprocessing methods in the following to compare the results of the best performing fingerprinting processes. A comparison of non-best performing classifiers and preprocessing methods would be possible but does not support our aim to find the best method for hardware-based fingerprinting for the purpose of device authentication.

In order to determine the effectiveness of state-of-the-art features over raw data for sensor fingerprinting, we carried out several tests in two phases: First, we ran comparison tests for every single sensor listed in Table 1. The goal of these tests is to compare the recognition precision between utilizing the raw sensor data and the extracted features for fingerprints for each sensor. Second, we combined the data from different sensors to multi-sensor tests and applied the described methods to clarify the recognition precision when taking several sensors into account. We determined five combinations to be of interest due to results from previous experiments:

1. *Accelerometers* including sensors for measuring acceleration and linear acceleration. Recognizing this data precisely has been the explicit purpose of the fingerprint features.
2. *Accelerometers & Gyroscope* extends the combination by gyroscope sensor readings. Usually, if a device embeds accelerometers, a gyroscope is built-in.
3. *All Available Sensors* includes data from all sensors listed in Table 2.
4. *No Accelerometers* takes only sensors into account that do not measure acceleration. This is the "inverse scenario" of 1.
5. *No Accelerometers & Gyroscope* is the "inverse scenario" of 2 and excludes acceleration sensors as well as gyroscope measurements.

In the following, we present the results of the single sensor tests as well as the combination tests.

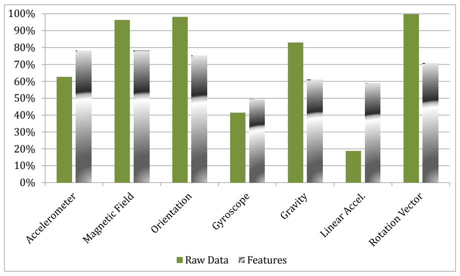

Fig. 2. Recognition precisions per sensor for device recognition

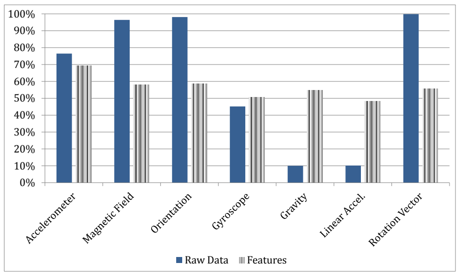

Fig. 3. Recognition precisions per sensor for model recognition

4.1 Single Sensor Tests

Our experiments confirm the overall result by Dey et al. [7]: The presented features provide the best precision for recognizing single devices on the basis of general purpose accelerometer data. Nevertheless, the precision of this case is about 78 %, leaving room for improvement. Our results indicate that for linear accelerometer sensors as well as for gyroscope data, the proposed feature set provides a better recognition rate than using the raw sensor readings. But the corresponding precision rates of about 49 % and 59 % are not suitable for device authentication in practice. Hence, device authentication methods should not rely on accelerometers and gyroscopes only.

For all other sensors, the utilization of features leads to a lower precision compared to the raw sensor measurements. The highest recognition precision could be achieved with plain sensor measurements and more different sensor types lead to a better precision. Thus, device authentication does not need to be based on a mathematical feature set but is feasible using raw sensor data as well. Table 3 summarizes the results of all single-sensor recognition experiments.

It is tempting to suspect a connection between a high device recognition precision and the number of different sensor hardware models shown in Table 2. Nevertheless, we could not show a significance to substantiate this assumption: Linear accelerometers and rotation sensors both have low model diversity and while the first are not suitable for device recognition showing a maximum precision of about 59 %, the second show an outstanding precision for recognizing single devices of almost 100 %. These differences are visualized in Figure 2.

As shown in Figure 3, the precision based on accelerometers, magnetic field sensors, orientation sensors and rotation sensors is significantly lower compared to when raw measurement events are used when it comes to model recognition. Concurrently, using raw data for model recognition fails for data from gyroscopes, gravity sensors and linear accelerometers. Using the feature set performs better for these sensors, but still the recognition precision does not exceed 55 % and cannot be considered for a reliable model recognition in a practical setting. In summary, recognizing device models by only one sensor type does not require the use of features and can be done on the basis of raw sensor data for

Table 3. Recognition precisions per sensor type, identifier and data set of single-sensor tests in percent

Sensor	Identifier	Data	Classifier	Average Precision
Acceleration	**Device**	**F**	**ET**	**78.2300**
	Device	R	k-NN	62.6781
	Model	F	ET	69.3900
	Model	**R**	**BT**	**76.4570**
Magnetic Field	Device	F	ET	78.0100
	Device	**R**	**RF**	**96.3808**
	Model	F	ET	57.9100
	Model	**R**	**ET**	**96.4232**
Orientation	Device	F	ET	75.2400
	Device	**R**	**k-NN**	**98.2033**
	Model	F	ET	58.7400
	Model	**R**	**k-NN**	**98.1090**
Gyroscope	**Device**	**F**	**BT**	**49.4400**
	Device	R	k-NN	41.4460
	Model	**F**	**BT**	**50.5000**
	Model	R	k-NN	45.1595
Gravity	Device	F	ET	60.9500
	Device	**R**	**k-NN**	**82.9912**
	Model	**F**	**ET**	**54.7200**
	Model	R	k-NN	9.9967
Lin. Acceleration	**Device**	**F**	**BT**	**58.9200**
	Device	R	k-NN	18.8124
	Model	**F**	**BT**	**48.3500**
	Model	R	k-NN	10.1388
Rotation Vector	Device	F	ET	70.7200
	Device	**R**	**k-NN**	**99.8063**
	Model	F	ET	55.5700
	Model	**R**	**k-NN**	**99.8216**

R = raw data, F = features
k-NN = k-NearestNeighbor, BT = BaggingTree,
ET = ExtraTrees, RF = RandomForest
showing only best performing classifiers
bold rows show maximum precision rate

four of seven sensor types, while the other three sensor types cannot be used to distinguish between device models at all.

In summary, the state-of-the-art feature set for accelerometer fingerprinting serves its purpose and is a reasonable way for device recognition based on

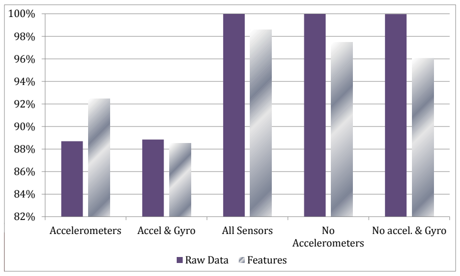

Fig. 4. Recognition precisions per combination for device recognition

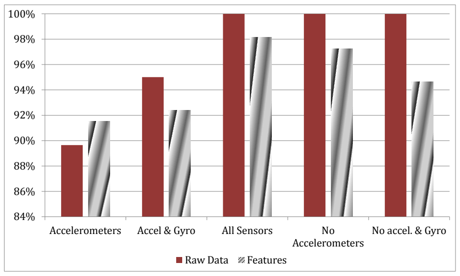

Fig. 5. Recognition precisions per combination for model recognition

accelerometers and gyroscope data. However, it is not suitable to distinguish devices based on data from other sensor types. Furthermore, using raw measurements of these other sensor types enables an even higher recognition precision. The use of accelerometer-based fingerprinting utilizing mathematical features for device authentication is questionable as fingerprinting based on other sensors performs significantly better.

4.2 Multi Sensor Tests

While the use of single sensors does not seem to provide a reliable method for device or model recognition—and thus for device authentication—precision rates increase generally when sensor types are combined.

The first combination includes both types of accelerometers. In this case, using the feature set for device recognition performs well and achieves a precision of about 92 %. For every other case we tested, the utilization of features did not exceed the precision achieved by the use of raw measurements. Especially when accelerometers are left out (cases four and five), the recognition based on raw data is more effective. In total, there is no precision result lower than 88.5 %, while the maximum of 99.99 % can be achieved by using raw data of all sensors except accelerometers. Table 4 shows the results for all combination test.

Consequently, using raw measurements of sensors for magnetic field, orientation, gravity, rotation and gyroscope is most effective for fingerprinting mobile devices. These sensors, which are common in modern devices, improve sensor-based fingerprinting significantly and can be used as a basis for reliable device fingerprinting in practice. Figure 4 shows the achieved maximum precisions of each sensor combination described above.

This finding is also valid for model recognition: Again, for accelerometers the feature set yields the best precision, but in all other combinations, features are not necessary to achieve recognition rates of up to 99.995 %. Figure 5 shows the results of the combination tests per model.

Table 4. Recognition precisions per sensor combination, identifier and data set of combination tests in percent

Sensors	Identifier	Data	Classifier	Average Precision
Accelerometers	**Device**	**F**	**BT**	**92.4782**
	Device	R	BT	88.6941
	Model	**F**	**ET**	**91.5432**
	Model	R	BT	89.6469
Accelerometers & Gyroscope	Device	F	ET	88.5019
	Device	**R**	**BT**	**88.8444**
	Model	F	ET	92.3950
	Model	**R**	**RF**	**95.0076**
All Available Sensors	Device	F	ET	98.6026
	Device	**R**	**ET**	**99.9806**
	Model	F	ET	98.1615
	Model	**R**	**RF**	**99.9950**
No Accelerometers	Device	F	RF	97.2484
	Device	**R**	**ET**	**99.9922**
	Model	F	ET	97.4589
	Model	**R**	**ET**	**99.9821**
No Accelerometers & No Gyroscope	Device	F	RF	94.6407
	Device	**R**	**RF**	**99.9848**
	Model	F	RF	96.0450
	Model	**R**	**ET**	**99.9671**

R = raw data, F = features
k-NN = k-NearestNeighbor, BT = BaggingTree,
ET = ExtraTrees, RF = RandomForest
showing only best performing classifiers
bold rows show maximum precision rate

4.3 Discussion

While we found the feature set to be most precise for device recognition on the basis of accelerometer data, best recognition rates for devices and models can be achieved by sensor combinations without accelerometers applied to raw measurements. Taking common sensors together, recognition precisions of 99.98 % up to 99.995 % can be achieved without needing to consider complex features. Our experiments indicate that combining the data of different sensors leads to a more effective fingerprinting than the application of the proposed features.

The feature set is suitable for the case of recognizing single devices by accelerometer data, but not reliable in any other case. Furthermore, given a large quantity of real-world data, the same results can be achieved without these features using the same or comparable machine learning techniques. For other sensor types, using raw sensor data is more effective, esp. for recognition of single devices. However, both data types yield disadvantages: On the one hand, calculating features needs computational power but also condenses the data. On the other hand, storing all events' raw measurements requires more storage capacities but no mathematical calculations need to be made. Ultimately, single devices as well as device models can be recognized best when combining the measurement data of several sensors.

For the purpose of device authentication, sensor fingerprinting is a valid method: High recognition rates can be achieved under realistic conditions in a real-world data set. While previous research mostly focusses on accelerometers and gyroscopes as these are accessible via the Web, we found other sensor types' hardware imperfections to be more characteristic making them even more important in this context. As sensor-based hardware fingerprinting opens up the possibility to distinguish unique devices at very high precision, it does not seem to be necessary to have a fallback solution like device model recognition at all.

Additional to the adversarial scenarios described in Section 2, it may be possible to randomize sensor measurements in order to prevent a recognition of a specific device. However, tampering sensor readings with random data requires a customization of the device's software like its browser when sensors are queried by websites or even the operating system when apps access the sensors for fingerprinting. Furthermore, tampering sensor readings raises a problem in practice: Sensors are used for specific reasons and adding randomness to their measurements may be helpful to evade fingerprinting their hardware imperfections but may also result in unwanted behavior of functionalities which rely on sensors. For instance, if a websites accesses a device's accelerometers or gyroscope and their data is randomized or tampered by the device first, the website's functionality and hence the user experience may be affected. Ultimately, as the goal is to authenticate a device and randomization is only capable of preventing a recognition, the more relevant attack would be the imitation of a specific device.

For such a mimic attack, an attacker would need to fake her own sensor data and replace it by the target device's sensor data. As described in Section 2.2, an attacker has to solve some challenges to perform this attack while having little chances of success. Although such an attack is difficult to carry out in practice, we will investigate this scenario in future work.

As more and more sensors are embedded in modern mobile devices, examining more sensor types for the purpose of hardware-based device fingerprinting will be the subject of future work. The availability of other sensors may lead to even better recognition results.

5 Related Work

Dey et al. [7] proposed mathematical features based on accelerometer readings for fingerprinting mobile devices. Their work illustrates the possibility to identify devices by conducting a series of training and test set scenarios on 107 different stand-alone chips, smartphones, and tablets under laboratory conditions. While their work focuses on accelerometers only, we also inspected other sensor types like magnetic field or rotation vector sensors. The usefulness of the feature set could be verified for accelerometers on a large real-world dataset of nearly 5,000 devices. We have also shown that fingerprinting mobile devices is more precise when taking other available sensor data into account. Furthermore, our results indicate that machine learning algorithms can be applied on the raw measurement events and specific features used for pre-processing the raw measurements do not yield better results.

Several studies focus on real-world accelerometer data for recognizing movement or behavior. For instance, it has been shown that steps of a walking or running person can be detected clearly with the help of a smartphone's accelerometers [31]. Dargie and Denko studied the behavior of accelerometers during similar movements and placed accelerometers on moving humans and cars [6]. They conclude that the extracted frequency domain features remain generally more robust than time domain features. In our study, we applied sensor readings gathered from both resting and moving devices and included features of time domain as well as frequency domain.

A study by He utilized machine learning techniques to recognize human activities by accelerometer and gyroscope data [14]. Three feature sets were applied including 561, 50 and 20 features to distinguish between six different human activities. While this work aims to detect activities, our experiments do not consider the current movement as artefact, but aim to identify devices (and group devices by model) on the basis of real-world sensor data.

A non sensor-based method for hardware fingerprinting has been introduced by Moon et al. [25] as well as Kohno et al. [20]. The identification of devices is achieved by measuring *clock skews*. While the common idea is the recognition of devices by hardware differences, these studies focus on time differences and do not consider any of a device's sensors.

Bates et al. explored mobile device model recognition and showed that manufacturer models can be distinguished by USB data with an accuracy of 97 % [4]. Notwithstanding, our experiments have shown that an even higher accuracy can be achieved by sensor-based hardware fingerprinting.

6 Conclusion

In this paper, we performed a detailed assessment of the effectiveness of sensor-based fingerprinting. We compared the benefit of using a well-defined feature set including attributes from time domain as well as frequency domain to using raw sensor data as input. We utilized five different machine learning techniques together with three data preparation processes and compared the precision at which a single device or a device model can be recognized on the basis of its hardware. To base our work upon real-world conditions, we gathered sensor data of almost 5,000 mobile devices. As a part of our work, we implemented the signal feature extraction process described by Dey et al. [7].

While we found the proposed feature set suitable for accelerometer-based recognition of single devices we have shown that it lacks precision for other sensor types. For non-accelerometer sensors the use of raw sensor readings as a basis for hardware fingerprinting results in a higher recognition precision. Furthermore, combining different sensor types leads to an even better precision and a higher robustness. We find that accelerometer measurements combined with other sensor data yield real-world recognition precisions of 99.98% up to 99.995%. In general, taking other common sensor types into account for fingerprinting results in a better precision than utilizing the previously proposed feature set.

Given these findings, hardware-based device fingerprinting with sensor data is feasible and a valid method for device authentication. However, device authentication methods should not rely on accelerometers and gyroscopes only but on combinations of different sensor types. For these, the calculation of features means computational effort without improving device recognition. Ultimately, using raw measurements of different sensor types is the most accurate way to instrument sensor-based hardware fingerprinting for device authentication. Implementing such an authentication mechanism may help handling suspicious login attempts, password resets, and even remedy SIM spoofing.

References

1. Acar, G., Juarez, M., Nikiforakis, N., Diaz, C., Gürses, S., Piessens, F., Preneel, B.: FPDetective: Dusting the Web for Fingerprinters. In: ACM Conference on Computer and Communications Security (CCS) (2013)
2. Amari, S.I., Murata, N., Muller, K.R., Finke, M., Yang, H.H.: Asymptotic statistical theory of overtraining and cross-validation. Neural Networks, IEEE Transactions on 8(5), 985–996 (1997)
3. Bader, R.: Nonlinearities and Synchronization in Musical Acoustics and Music Psychology. Current Research in Systematic Musicology, Springer (2013)
4. Bates, A., Leonard, R., Pruse, H., Butler, K., Lowd, D.: Leveraging USB to Establish Host Identity Using Commodity Devices. In: Proceedings of the Network and Distributed System Security Symposium (NDSS) (2014)
5. Bojinov, H., Michalevsky, Y., Nakibly, G., Boneh, D.: Mobile Device Identification via Sensor Fingerprinting. arXiv preprint arXiv:1408.1416 (2014)
6. Dargie, W., Denko, M.K.: Analysis of Error-Agnostic Time-and Frequency-Domain Features Extracted From Measurements of 3-D Accelerometer Sensors. Systems Journal, IEEE 4(1), 26–33 (2010)
7. Dey, S., Roy, N., Xu, W., Choudhury, R.R., Nelakuditi, S.: AccelPrint: Imperfections of Accelerometers Make Smartphones Trackable. In: Proceedings of the Network and Distributed System Security Symposium (NDSS) (2014)
8. Eckersley, P.: How Unique is Your Web Browser? In: International Conference on Privacy Enhancing Technologies (PETS) (2010)
9. Eisenberg, G.: Identifikation und Klassifikation von Musikinstrumentenklängen in monophoner und polyphoner Musik. Cuvillier (2008)
10. Eubank, C., Melara, M., Perez-botero, D., Narayanan, A.: Shining the Floodlights on Mobile Web Tracking – A Privacy Survey. In: Web 2.0 Security & Privacy Conference (W2SP) (2013)

11. Gelfand, S.: Essentials of Audiology. Thieme (2011)
12. Guyon, I.: A Scaling Law for the Validation-Set Training-Set Size Ratio. In: AT & T Bell Laboratories (1997)
13. Hardcastle, W., Laver, J., Gibbon, F.: The Handbook of Phonetic Sciences. Blackwell Handbooks in Linguistics, Wiley (2012)
14. He, H.: Human Activity Recognition on Smartphones Using Various Classifiers (2013)
15. Hupperich, T., Maiorca, D., Kührer, M., Holz, T., Giacinto, G.: On the Robustness of Mobile Device Fingerprinting. In: Annual Computer Security Applications Conference (ACSAC) (2015)
16. Jensen, K.: Timbre models of musical sounds. Ph.D. thesis, Department of Computer Science, University of Copenhagen (1999)
17. Jones, E., Oliphant, T., Peterson, P., et al.: SciPy: Open source scientific tools for Python (2001–), `http://scipy.org,` as of April 26, 2016
18. Kamkar, S.: Evercookie – never forget. (2010), retrieved in June 2015 from `http://samy.pl/evercookie/`
19. Klapuri, A., Davy, M.: Signal Processing Methods for Music Transcription. Springer (2007)
20. Kohno, T., Broido, A., Claffy, K.C.: Remote physical device fingerprinting. Dependable and Secure Computing, IEEE Transactions on 2(2), 93–108 (2005)
21. Krimphoff, J., McAdams, S., Winsberg, S.: Caractérisation du timbre des sons complexes. ii. Analyses acoustiques et quantification psychophysique. Le Journal de Physique IV 4(C5), C5–625 (1994)
22. Lerch, A.: An Introduction to Audio Content Analysis: Applications in Signal Processing and Music Informatics. Wiley (2012)
23. Liang, B., You, W., Liu, L., Shi, W., Heiderich, M.: Scriptless Timing Attacks on Web Browser Privacy. In: Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN) (2014)
24. Mcadams, S.: Perspectives on the contribution of timbre to musical structure. Computer Music Journal 23(3), 85–102 (1999)
25. Moon, S.B., Skelly, P., Towsley, D.: Estimation and removal of clock skew from network delay measurements. In: INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE. vol. 1, pp. 227–234. IEEE (1999)
26. Nikiforakis, N., Kapravelos, A., Joosen, W., Kruegel, C., Piessens, F., Vigna, G.: Cookieless Monster: Exploring the Ecosystem of Web-Based Device Fingerprinting. In: IEEE Symposium on Security and Privacy (2013)
27. Park, T.H.: Salient feature extraction of musical instrument signals. Ph.D. thesis, DARTMOUTH COLLEGE Hanover, New Hampshire (2000)
28. Peeters, G., Giordano, B.L., Susini, P., Misdariis, N., McAdams, S.: The timbre toolbox: Extracting audio descriptors from musical signals. The Journal of the Acoustical Society of America 130(5), 2902–2916 (2011)
29. Sanei, S., Chambers, J.: EEG Signal Processing. Wiley (2013)
30. Satapathy, S., Udgata, S., Biswal, B.: Proceedings of the International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) 2013. Advances in Intelligent Systems and Computing, Springer (2013)
31. Sinofsky, S.: Supporting sensors in Windows 8 (2012), `http://blogs.msdn.com/b/b8/archive/2012/01/24/supporting-sensors-in-windows-8.aspx,` as of April 26, 2016
32. Smith, S.W.: Digital signal processing: a practical guide for engineers and scientists. Newnes (2003)

33. Van Der Walt, S., Colbert, S.C., Varoquaux, G.: The NumPy array: a structure for efficient numerical computation. Computing in Science & Engineering 13(2), 22–30 (2011)
34. Wang, J., Yen, G., Polycarpou, M.: Advances in Neural Networks – ISNN 2012: 9th International Symposium on Neural Networks, ISNN 2012, Shenyang, China, July 11-14, 2012. Proceedings. No. Teil 2 in Lecture Notes in Computer Science, Springer Berlin Heidelberg (2012)
35. Yang, Y., Chen, H.: Music Emotion Recognition. Multimedia Computing, Communication and Intelligence, CRC Press (2011)
36. Zelkowitz, M.: Advances in Computers: Improving the Web. Advances in computers, Elsevier Science (2010)