Hindawi
Security and Communication Networks
Volume 2020, Article ID 3893106, 13 pages
https://doi.org/10.1155/2020/3893106

Research Article

PowerPrint: Identifying Smartphones through Power Consumption of the Battery

Jiong Chen,[1] Kun He,[1,2] Jing Chen,[1] Yingying Fang,[1] and Ruiying Du[1,3]

[1]*Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan, China*
[2]*Science and Technology on Communication Security Laboratory, Chengdu, China*
[3]*Collaborative Innovation Center of Geospatial Technology, Wuhan, China*

Correspondence should be addressed to Kun He; milloglobe@gmail.com

Received 23 February 2020; Revised 16 October 2020; Accepted 28 October 2020; Published 17 November 2020

Academic Editor: Kim-Kwang Raymond Choo

Device fingerprinting technologies are widely employed in smartphones. However, the features used in existing schemes may bring the privacy disclosure problems because of their fixed and invariable nature (such as IMEI and OS version), or the draconian of their experimental conditions may lead to a large reduction in practicality. Finding a new, secure, and effective smartphone fingerprint is, however, a surprisingly challenging task due to the restrictions on technology and mobile phone manufacturers. To tackle this challenge, we propose a battery-based fingerprinting method, named PowerPrint, which captures the feature of power consumption rather than invariable information of the battery. Furthermore, power consumption information can be easily obtained without strict conditions. We design an unsupervised learning-based algorithm to fingerprint the battery, which is stimulated with different power consumption of tasks to improve the performance. We use 15 smartphones to evaluate the performance of PowerPrint in both laboratory and public conditions. The experimental results indicate that battery fingerprint can be efficiently used to identify smartphones with low overhead. At the same time, it will not bring privacy problems, since the power consumption information is changing in real time.

1. Introduction

The rapid development of smartphones has made people's lives more convenient. However, it can also give unlawful users a chance to make some wrongdoing. For example, some illegal users may publish comments, pictures, or videos that violate the law, and some may use multiple devices for fraud. Therefore, we need to locate these problems and protect the civilization of the network space. We can use device fingerprinting technology to track the devices, even to track the users. A device fingerprint is information collected about a remote computing device for the purpose of identification. Fingerprints can be used to fully or partially identify individual users or devices even when cookies are turned off. For example, ad networks and web publishers could collect users' online habits and other information from the smartphones and further use this information to generate device fingerprint to identify devices [1–4].

Existing device fingerprinting methods can be divided into two categories: browser fingerprinting and hardware fingerprinting. Browser fingerprints, such as cookies and browser history, have been widely used in the computer systems to identify the devices [5]. Web analytic services also collect basic web browser configuration information in an effort to accurately measure real human web traffic and discount various forms of click fraud for a long time. In the ideal case, all client devices would have a different fingerprint that would never change. With those assumptions, it is possible to uniquely distinguish all devices on the network without the explicit consent of the users themselves. However, the assumptions of uniqueness and immutability are difficult to guarantee in practice. On the smartphones, the browser fingerprint is even less efficient since the high

system version and plug-in repetitive rate reduce the recognition accuracy significantly [6]. Moreover, the browser fingerprinting may bring the privacy problems due to its fixity. As the improvement of users' privacy awareness, many users have disabled the cookies and history in their browsers, which has influenced the development of the browser fingerprinting.

Hardware fingerprinting approaches identify devices via the characteristics of hardware components [7]. On the smartphones, the most common hardware that are used for fingerprinting are the built-in sensors, such as accelerometers, gyroscopes, microphones, and microspeakers. The discrepancies of those sensors on manufacturing make them having unique characteristics and can be captured in the form of a fingerprint. Most importantly, the sensor data information has always been changing. Thus, it does not bring any privacy problems.

Unfortunately, existing sensor fingerprinting schemes require certain conditions to capture the characteristics. For example, the fingerprinting schemes employing a gyroscope, accelerometer, and microphone, which are widely used in smartphones, require the smartphone is either held in a user's hand or resting on a flat surface. More specifically, measuring the offset and sensitivity of the gyroscope would require subjecting the device to constant angular velocity rotation at different speeds, and the microphone requires users to make certain sound [8]. These methods are difficult to carry out even in the laboratory. We will discuss the limitations of the existing schemes further in Section 6. Therefore, designing an effective hardware fingerprinting is still an urgent challenge.

In this paper, we aim to design a method that can identity the device with a high accuracy without revealing the privacy information. To this end, we use the power consumption of the smartphone battery to generate device fingerprint, since capturing the changing data of the battery does not require any authority and does not disclose any privacy data. We find many differences in the design of different devices and batteries, such as the CPU clock frequency and battery capacity. When integrating these deviations, we can capture the uniqueness of the power consumption rate in performing specific tasks on different devices. Moreover, we prove that the power consumption rate is relatively stable on a smartphone, and the battery durability is different even for the same phone model since the usages are different between users. These differences can be further extracted to unique device fingerprints. However, there is a great challenge in this kind of fingerprinting, that is, the performance of battery is affected by many factors, such as environment, temperature, and service life.

To tackle this challenge, we propose a machine learning-based approach to extract a large number of features from the battery, whose accuracy is improved by performing customized tasks at different times. The proposed method, named PowerPrint, is evaluated from different aspects on a wide variety of smartphone models and operating systems, both the laboratories and nonlaboratories. According to our experiments, generating a unique fingerprint only takes a collecting session lasting for 25 seconds, and recognizing the device with the fingerprint can be carried out within about 3 seconds. Our contributions can be summarized as follows:

(1) We perform the first large-scale study of power consumption rate of the smartphone batteries. We show that the power consumption rate of the battery is one of the most discriminating attributes.

(2) We present a new and efficient device fingerprinting method to identify the devices, which utilizes the power consumption rate of the battery.

(3) We explore a more secure way to identify a device which uses the characteristics in variation to extract the fingerprint features. In this way, the security problems brought by the stationarity of fingerprints can be eliminated.

This work is an extension of our conference version [9]. Compared with [9], we evaluate PowerPrint with a new measure as we take the recall rate into consideration, so we can make a more comprehensive and accurate evaluation. We also analyze the principle of app power statistics. Moreover, we analyze the impact of the user's operations on the final recognition performance when the device is being identified, which has further explained the reason why we has not used apps in the app store as a power collection object. The final results show that the battery fingerprint can well be used to identify smartphones even if we re-evaluate the whole scheme.

Roadmap. The remainder of this paper is organized as follows. We present the application scenarios in Section 2. Section 3 shows how the battery works and why we can fingerprint it successfully. In Section 4, we describe the different temporal and spectral features selected in our experiments and how we fingerprint the battery, along with the classification algorithms. We present our fingerprinting results and evaluate the features we selected in Section 5. We present the related work about device fingerprint in Section 6. Section 7 discusses some limitations of our approach and a controversial problem. Finally, we conclude in Section 8.

2. Application Scenarios

In this section, we introduce several interesting application scenarios about device fingerprinting. There are also many other scenarios where device fingerprinting can be used.

Considering the first situation: many shopping malls have their own applications or official accounts, which are used for popularizing themselves. Obtaining users' consumer habits is very important to increase the turnover for the malls. With a mall getting a user's consumption record, the mall can recommend to the user some discount information and shopping guide information of some stores which are more likely to buy. In most cases, the devices and users are one to one correspondence, which means that identifying devices is identifying users. Our work can help identify a device without user's login or any permissions, which can perform well in this scenario.

Perhaps, an even more relevant scenario is user blocking. Websites and forums need to block some illegal user accounts regularly. However, once the user re-registers with another account, this user cannot be detected easily. Our methods can be used in this situation. The websites and forums only need to collect information about the device used by the illegal user. Once the illegal user re-registers on the websites on the same devices even if he/she employs various anonymity technologies such as IP hiding, the managers of the websites can identify and block the user. Similarly, such fingerprints are useful in the detection and prevention of online identity theft and credit card fraud. In practice, device fingerprints can be used to predict the likelihood of users who may be deceived based on their signal feature, before they have been deceived [10], which can be meaningful for many users. For different application scenarios, PowerPrint can be used as a plug-in for app developers to identify smartphones.

3. Background of Battery Fingerprint

In this section, we briefly take a closer look at the background of battery; this will provide an understanding of how they can be used to uniquely fingerprinting smartphones. In this paper, we collect the power consumption information of a specified app, and the details of the app will be introduced in the next section. We choose to fingerprinting the power on an app rather than the smartphone as that the overall power consumption of smartphones is easily affected by many factors, which will result in the unstable recognition rate of the smartphone. For example, users may play games the whole day in the weekend, which is very power consuming. After that, it may be regular life away from the game for a whole week. There will be many differences in the power consumption information in these two states. If just fingerprinting a specified app (we default that the user's habit of using this app is relatively stable), the power consumption information will be relatively stable, which can reflect the characteristics of smartphone battery and the characteristics of other components in the smartphone.

It has been widely known that the smartphone is consisted of many components, including CPU, WIFI, GPS, and so on. Therefore, the total power consumption of Android app is the sum of the total consumption of components involved in the running process of app. Assuming the operation of app resulted in CPU operation, if t and w represent time and power consumption per unit time, then power consumption for CPU in app (W) can be computed as, $W = w * t$. In physics, it is universal to use voltage value (U) and current value (I) to calculate the power consumption information, $W = U * I * t$. The voltage value is constant in the smartphone, so the power consumption can be computed through electric capacity (Q, unit: mAh), $Q = I * t$. We use electric capacity to represent the power consumption information in this paper.

The next brief introduction of Android is how to store and read app power consumption information, which is explained in detail in the official documents [11]. First, the class PowerUsageSummary.java gets the file (named as betterystats.bin) through the system service (batteryinfo). Android engineers have already declared that the file is used to maintain low-level data about the kinds of operations the device and apps are performing between battery changes. It is used to compute the battery usage shown in the "Battery use" UI in settings. Moreover, the file has no impact on the battery life and current battery level shown to the user. Android component current information is stored in power-profile.xml. Meanwhile, the components' power consumption should be related to the specific hardware so that every OEM vendor should have its own power-profile.xml. Smartphones are equipped with hardwares of different vendors, as we have mentioned before, and the difference will make a difference of smartphones' power consumption.

In Android 5 and earlier versions, the object of power statistics is not the app, but UID. The relationship between UID and an app is that, if 2 app signatures are the same as sharedUserId, they have the same UID at run time. That is to say, what processAppUsage has counted may be the power consumption information of multiple apps. For ordinary apps, most of them have their own exclusive UID, but two Android system applications may have the same UID. Therefore, the power consumption of app is equal to the sum of power consumption of every Process of UID, wake lock power consumption, data traffic consumption, WIFI power consumption, and other sensor power consumption. We can see that the steps are relatively cumbersome. However, there is good news that the total power consumption could be obtained easily by adding all the UID power consumption under the app after Android 5. In addition to the effects of these components on power consumption, it is also affected by environmental factors, the aging of the battery, the usage habits of the users, and the list of installation and application (Section 5 will show how we deal with these environmental noises). All of these once again confirm the thesis we have mentioned before, even the same type of smartphones may show different battery features, which finally can be used to identify smartphones uniquely.

4. Features Selection and Classification Algorithms

In this section, we describe the design of PowerPrint, which is based on the power consumption rate of smartphones' batteries. In PowerPrint, we have designed four fundamental steps: (1) data collection; (2) data processing; (3) feature extraction and selection; and (4) fingerprints matching. Figure 1 introduces these steps of our design on how to identify the devices briefly, which has shown us that data collected from Jack's phone in different time can be extracted to fingerprints and be matched each other in the servers successfully.

4.1. Data Collection. To collect more features and identify a smartphone accurately, we design a number of tasks of different power consumption rates and characteristic, such as big number computing, large file reading, large file writing, and broadcast transceiver. These tasks are easy to implement, and the power consumption rates are small but

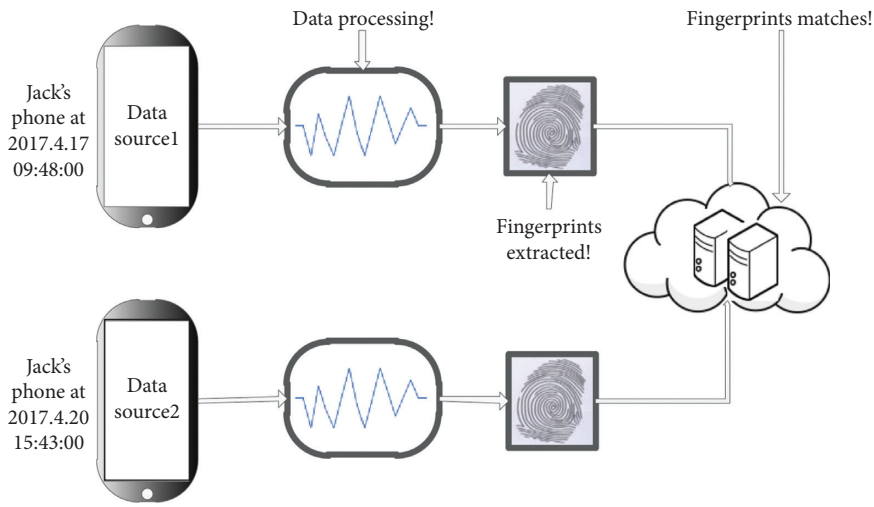

Figure 1: The project design of data processing, fingerprints extracting, and fingerprints matching. The data sources refer to smartphones' batteries.

quite different. The power consumption rates' order of these tasks from small to large is broadcast transceiver, large file reading, large file writing, and big number computing. As different tasks may exhibit different features, it is necessary to combine different tasks in a round. When we implement the tasks in the terminal, we manually assign the tasks by the power consumption rate. In our design, the task which consumes less energy is performed first followed by the task which consumes more energy. The different tasks are performed by turn, and the cycle is about 5 seconds. When the tasks are performed, we collect the initial power consumption trend of the smartphones.

Moreover, we design two tools to collect data in smartphones. As our work follows the work of Zhang et al. [12], we use an altered PowerTutor (named APower) as the tool to collect the data of power consumption. PowerTutor is an application for Google phones that displays the power consumption by major system components such as the CPU, network interface, display, GPS receiver, and different applications. It allows software developers to observe the impact of design changes on the power efficiency. Users can also use this application to determine how their actions impact battery life [13]. Besides, our experiment has confirmed that the tool works well and the result is accurate even in the other models of smartphones after we improve it. Moreover, we design an application ATest to perform the different tasks we design in this section. We will explain the reason why we choose ATest rather than the apps in the app store as the power collection object in our experiments. ATest afpownd APower can work well on Android 4.1–6.0. Therefore, APower collects the data generated from ATest; Figure 2 shows how they work. All subsequent experiments are conducted on this basis.

4.2. Data Processing. We need to process the data with smoothing to eliminate some noise interferences after we collect the original data. At the same time, it is also necessary to preserve the characteristics of the original data. In this paper, we choose the smooth function in MATLAB. There are several different smoothing methods, such as the moving average method, the Lowess method, the Loess method, and the Savitzky–Golay method. After comparing the smoothness and the distortion among these methods, we finally choose the Lowess method (local regression using weighted linear least squares and a 2nd degree polynomial model) to smooth the data curves. The method we chose is very good to remove the noise and to retain the data features to make the necessary preparation for the next step of feature extraction.

4.3. Feature Extraction and Selection. There are several possible strategies to extract relevant features from collected data. We design a simple tool based on mirtoolbox (a popular feature extraction library) and use it to extract features of the battery data stream [14]. We have extracted 25 features in both time and frequency domains finally.

One may think that using all features to identify the device is the optimal strategy. However, including too many features may worsen the performance in practice, due to their varying accuracies. Therefore, we explore all the features and select a subset of the features to optimize our fingerprinting accuracy. We use the FEAST toolbox [15] and utilized the joint mutual information criterion (known for providing the best balance around accuracy, stability, and flexibility with small data samples [16]) for ranking the features. We find that the best device identification effect can be achieved when the first 12 of all the features given by the tool are selected. As a result, we select the top 7 time domain features (see in Table 1) and top 5 frequency domain features (see in Table 2) to construct the fingerprints. We present the performance of the features in Section 5.3.

To show the validity of the features we selected, we have selected five devices (abbreviated as *A*, *B*, *C*, *D*, and *E*) to show the distinction of the spectral kurtosis, sum, standdeviation, and maximum between them. The results are presented in Figure 3. We can see that these features are much differentiated, as the performance of different features on different devices is different. The device *A* and device *B*

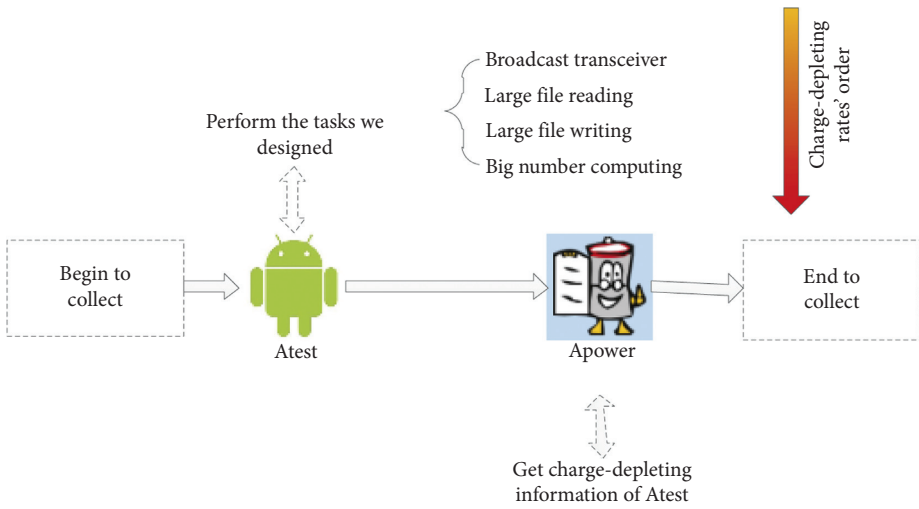

Figure 2: The workflow of the two applications.

Table 1: List of time domain features.

Feature name	Description
Sum	The whole power consumption the device at certain timestamps
Mean	The arithmetic mean of the power consumption at different timestamps
Max	Maximum power consumption
StandDeviation	Standard deviation of the power consumption
Kurtosisl	Measure of the spikiness of a distribution
Valley1	Measure of the flatness of a distribution
Valley2	Measure of the flatness of a distribution

Table 2: List of frequency domain features.

Feature name	Description
Spec. Kurtosis	Measure of the flatness or spikiness of a distribution relative to a normal distribution
Spec. Dev	Standard deviation of the signal strength of a spectrum
Spec. RMS	Spectral root of the arithmetic mean of the squares at various frequencies
Spec. Centroid	Represents the center of mass of a spectral power distribution
Flatness	Measures how energy is spread across the spectrum

are two devices which are similar in models, the system configuration, and installation software list, so the discrimination is lower than other devices. But, there is a certain degree of discrimination between the two devices, especially when we consider in combination with other characteristics.

4.4. Fingerprint Matching. Once the features have been extracted, we use unsupervised learning to classify smartphone battery. As in other supervised learning methods, our classifier consists of a training set and testing set. The training set is derived from the set of the specified device, and the testing set is derived from different devices. In the training set, in order to create a classifier, the learning algorithm may learn the sample dataset by matching some parameters. Moreover, the classifier predicts the most probable class for a given feature vector in the testing set. There are different classifiers as follows: Support Vector Machine (SVM), Naive–Bayes classifier, Multiclass Decision Tree, k-Nearest Neighbor (k-NN), Quadratic Discriminant Analysis classifier, and Bagged Decision Trees. We observe that SimpleKMeans outperforms the other classifiers after a comprehensive comparison of all the methods.

5. Performance Evaluation

In this section, we first describe our performance metrics (Section 5.1) and experimental preparation (Section 5.2). We, then, explore features to determine the minimal subset of features required to obtain high classification accuracy and evaluate the impact of some factors on the fingerprint accuracy at the remaining part of this section.

The key questions we have investigated and the corresponding findings are summarized as follows:

(1) How long will it take to generate a fingerprint? We find that 25 seconds of task execution is sufficient to model a device's fingerprint, which is shown in Figure 4.

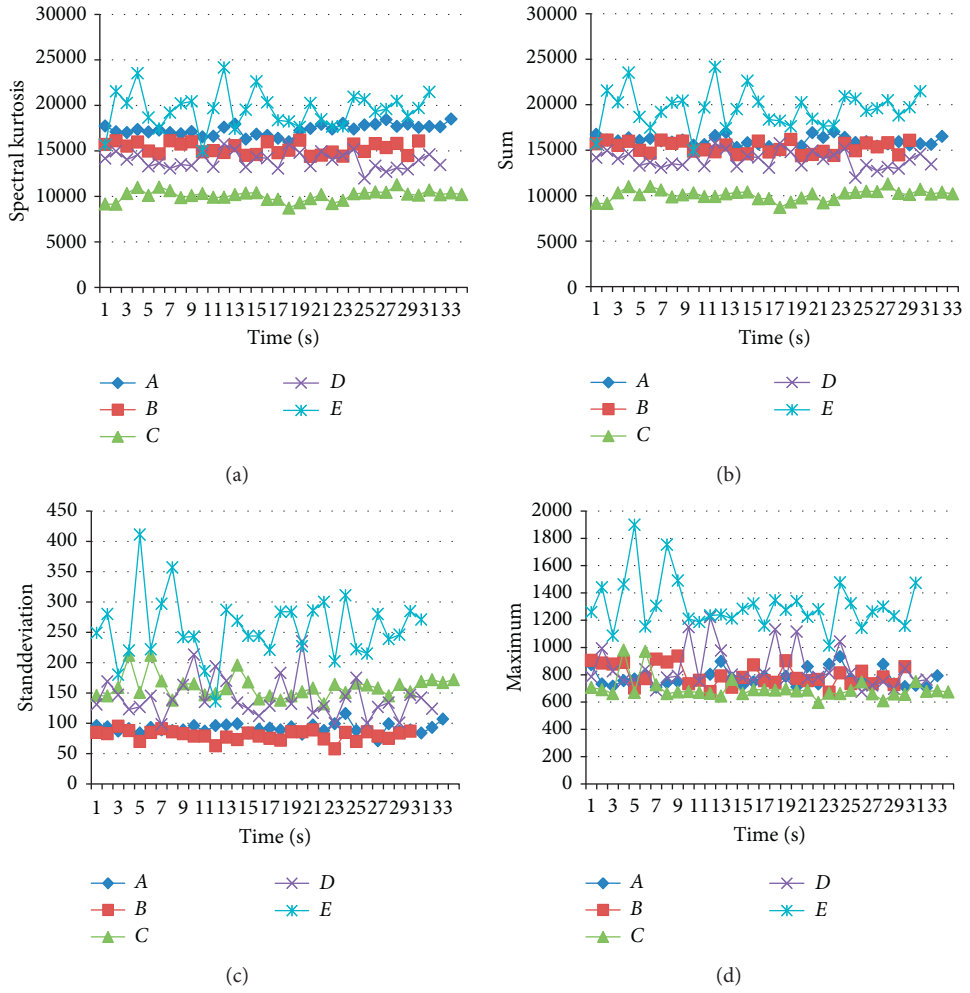

Figure 3: Battery responses of 5 devices for the same environment. (a) Spectral kurtosis. (b) Sum. (c) Standdeviation. (d) Maximum.

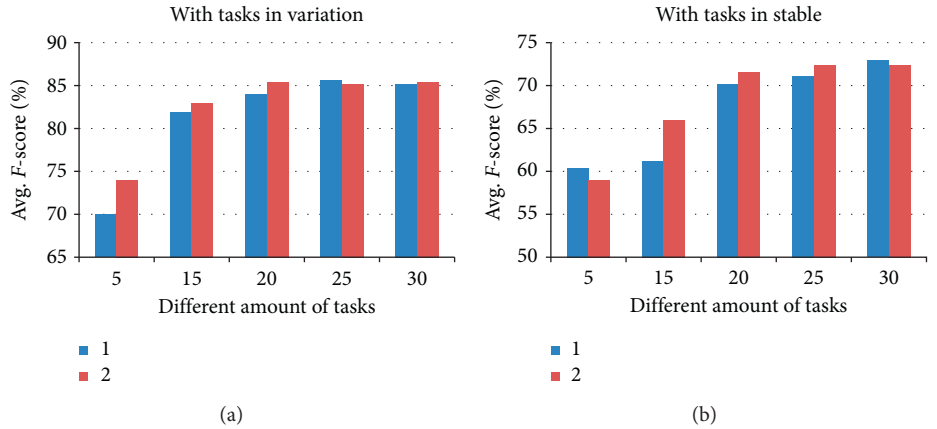

Figure 4: Significance of stimulation. (a) With stimulation. (b) Without stimulation.

(2) Are the fingerprints effective only at the fastest sampling rate? No, even we collect the data at a lower sampling rate, the device can also show a certain degree of distinction. Table 3 will prove the conclusion.

(3) Do the users need to be aware of the power consumption rate when they use the devices? No, our fingerprinting program does not seriously affect the smartphone battery life, which can be seen in the end of this section. Moreover, the program is not always

carried out, so the impact on the user experience can be ignored.

(4) Should we always perform our tasks in the background? Not needed. Collecting data in a short time can extract the eigenvalues to perform identification accurately.

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

5.1. Performance Metrics. In order to measure the performance of our fingerprinting techniques throughout this work, we calculate the standard classification metrics of precision, recall, and F-score in our evaluation [17]. We first compute the true positive (TP) and the true negative (TN) rate, which both mean the number of traces from the class that are classified correctly. Similarly, we compute the false positive (FP) and false negative (FN) rate as the number of wrongly accepted and wrongly rejected traces. Assuming there are n classes, for each class i ($1 \le i \le n$), we compute TP_i, TN_i, FP_i, and FN_i, respectively. To evaluate the overall performance of the multiclass classification in the classification in the presence of untrained devices, we compute the precision Pr_i, recall Re_i, and the F-score F_i for each class using the following equations:

$$\mathrm{Pr}_i = \frac{\mathrm{TP}_i}{\mathrm{TP}_i + \mathrm{FP}_i},$$

$$\mathrm{Re}_i = \frac{\mathrm{TP}_i}{\mathrm{TP}_i + \mathrm{FN}_i}, \tag{1}$$

$$F_i = \frac{2 \times \mathrm{Pr}_i \times \mathrm{Re}_i}{\mathrm{Pr}_i + \mathrm{Re}_i}.$$

The F-score is the harmonic mean of precision and recall, and it provides a good measure of the overall classification performance, since precision and recall represent a trade-off: a more conservative classifier that rejects more instances will have higher precision but lower recall, and vice versa. To obtain the overall performance of the system, we use the classification metrics of average precision AvgPr, average recall AvgRe, and average F-score Avg$F$:

$$\mathrm{AvgPr} = \frac{\sum_{i=1}^{n} \mathrm{Pr}_i}{n},$$

$$\mathrm{AvgRe} = \frac{\sum_{i=1}^{n} \mathrm{Re}_i}{n}, \tag{2}$$

$$\mathrm{Avg}F = \frac{2 \times \mathrm{AvgPr} \times \mathrm{AvgRe}}{\mathrm{AvgPr} + \mathrm{AvgRe}}.$$

5.2. Experimental Preparation. Considering that smartphones take up for a half of all global web pages served in 2017, especially in Asia, smartphones have occupied 60% of the total web traffic [18]. We perform our experiments on 10 phones of 9 different models in different periods. The different phones and the system versions are shown in Table 4.

Table 3: The impact of the experiment on the power consumption and users' experience of different devices.

Models	Capacity	Sampling frequency	Power consumed	Proportion (%)
Galaxy ON5	9360000	1	13110	0.14001
		2	13400	0.14320
Rongyao 5C	10800000	1	10223	0.09470
		2	12000	0.11110
Xperia Z2	11520000	1	11603	0.10070
		2	11350	0.09850
Lenovo K3 Note	10800000	1	9471	0.08770
		2	10412	0.09640
Samsung GT-P3110	4400000	1	14111	0.09800
		2	16011	0.11120

Table 4: Types of phones used.

Model	Versions	Quantity
HUAWEI RONGYAO 5C	Android 6.0	2
Samsung galaxy ON5	Android 5.1	1
SONY Xperia T2 UItra	Android 4.3	1
SONY Xperia Z2	Android 4.4.2	1
Lenovo K3 Note	Android 5.0	1
Samsung GT-P3110	Android 4.4.4	2
Xiaomi2	MIUI V4	1
Xiaomi4	MIUI 6	1

The models of these smartphones vary from Android 4.1 to Android 6.0. For training and testing the classifiers, we randomly split the dataset in the way that 95% of the data collected from each device goes to the training set and the remaining 5% goes to the test set. To prevent any bias in the selection of the training and testing set, we randomize the training and testing set several times and report the average F-score. For analyzing and matching fingerprints, we use a desktop machine with an Intel i3-4160 3.6 GHz processor with 8 GB RAM. The results show that the average time required to match a new fingerprint and identify a device is within 3 seconds.

Next, we consider several factors that may affect our ability to model fingerprints and classify devices. First, when the battery is low, the power consumption rate would be significantly higher. In order to maintain accuracy, we do not perform any recognition algorithms when the battery is less than 30%, and there is a detailed description about why we choose the level of 30% in Section 5.5. Second, when the device is charging, the battery is unstable, and we also do not perform recognition algorithms as the result may be inaccurate. Finally, our experiments are carried out at a temperature of 21 degrees in a laboratory environment, so the effect of environmental temperature on the accuracy of the results is insignificant.

Moreover, considering the other influence factors, the power consumption rate of the old batteries is different from new batteries. As the service time of the battery can be a characteristic to identify different devices, we do not consider the impact of battery loss. Certainly, this means that the battery can be used normally. Finally, as the noises come from other applications, we take measures similar to the

battery loss. Because even for the same device, if the application runs in different time, the power consumption is different. Besides, what we collect is the rate of power consumption, not merely the overall power consumption. We can think of it in this way: the power consumption of other applications is also one of the fingerprint elements for each device, so it seems that there is no need to distinguish these noise data.

5.3. Feature Evaluation. In the first phase of the experiment, we conducted a cluster analysis of our 10 smartphones in the laboratory to show the validity of the features we selected in Section 4.3. The classification results are shown in Figure 5, which shows the relationship among the accuracy of the classification, the feature quantity, and the sample size we selected.

We can see that, with the increase of the number of extracted features, the average F-score of the device identification changes at the same time. When the number of features increases to 12, the balance between the best rate of precision and the recall rate is reached, which shows that different devices can be distinguished at a high accuracy rate. When we increase the number of features, the curve no longer keeps growth. On the contrary, there may be a certain degree of decline for the curve. Therefore, we select the top 12 features to generate the fingerprints in order to get the best balance of the precision and recall. Moreover, we use this subset of features in all our later evaluations. In addition, we can see that the larger the sample size, the higher the accuracy of device identification in Figure 5, so we increase our sample set as much as possible. The relationship between the sample size and the accuracy of the device identification also prove the effectiveness of our program at a certain extend.

5.4. Precision. In this section, we conduct experiments with 10 devices mentioned in our experimental preparation. In order to ensure the stability of the training values and the effectiveness of the fingerprint, we only collect data when the battery capacity is above 30%; there is a detail description about why we choose the level of 30% in Section 5.5. The final average classification results are shown in Figure 6.

We test each device with 12 samples after the training process. The classification result for each device is shown in Figure 6. If points of the same color fall on the horizontal line that corresponds to them, it means that the smartphones that correspond to them are correctly classified. We can see that most points fall on the same horizontal line they correspond to, which means that most devices can be classified accurately. One of the Samsung GT-P3110 has been classified correctly for 8 times, and another Samsung GT-P3110 has been classified correctly for 10 times. Since the two Samsung tablets are totally the same in models, software configurations, and the service life, the classification result is not as good as that of other devices. Meanwhile, there are also two HUAWEI RONGYAO 5C, but we can see that the two devices' performances are not bad in the result. There are many factors which can impact the final performance of

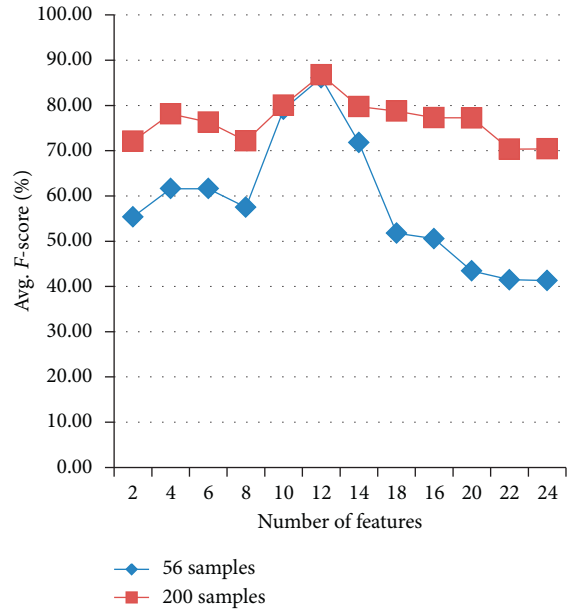

Figure 5: Exploring the relationship among the accuracy of cluster analysis, the number of features, and the number of the samples.

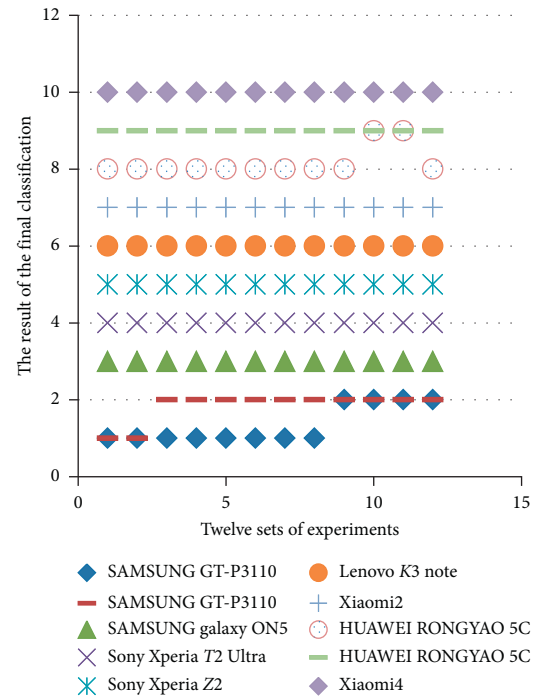

Figure 6: Exploring the overall performance of the classification.

the four devices, just as we have mentioned in Section 1; the service life of the devices and the number of kinds of software installed also have an impact on the results. So, even for the same type of smartphones, the recognition effect is not bad.

5.5. Impact of Batteries' Current Power. We will give an answer at what have been mentioned earlier to make sure that the current power is not less than 30% in this section.

We explore the impact of smartphone current power on fingerprinting effectiveness, which is inseparable with the choice of the right moment to start data collecting. Data acquisition is performed at different power levels (10%, 20%, 30%, 40%, 50%, 60%, 70%, and 80%, respectively), in order to test the effect of different power levels on the accuracy of the test values. Meanwhile, both our training set and test set are selected under the same power levels.

We show the relationship between batteries' current power and the accuracy of the test values in Figure 7. The result shows that the accuracy can reach more than 86% even when the power level is changed. We can also notice that as the current power increased, the accuracy changed from 75% to more than 86%, and when the current power is greater than 30%, the accuracy remained stably around 86%. Thus, we can draw a conclusion that the batteries' current power does have an impact on the accuracy of the classification performance. Besides, when the current power consumption is greater than 30% while carrying out an experiment, the impact of the current power on the classification is almost negligible. That is why we need to ensure the current power to be greater than 30% when we carry out all the experiments.

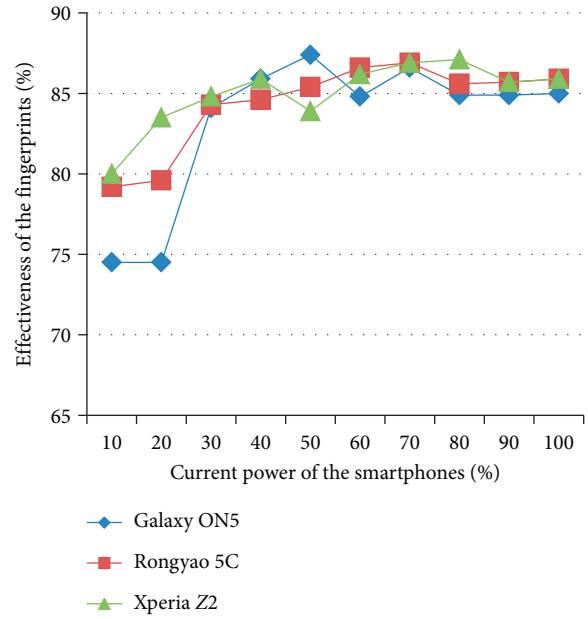

Figure 7: The impact of smartphone current power on fingerprinting effectiveness.

5.6. Scalability. The accessible amount of data in a laboratory environment are limited, and it is difficult to test a system with a very large set of devices. Thus, to verify the effectiveness of the fingerprints, we conduct an experiment where we increase the number of devices outside the laboratory environment. For this purpose, we called the other five users' devices to carry out our experiment. We have not called for more devices because the process of collecting data is a little time consuming. With their assistance, we collect the other more than 150 data sets from the five devices. However, some of the volunteers have not followed the steps we request, so we only keep those following the steps, and finally, we get 100 data sets.

The results in Figure 8 show that the fingerprints collected from these devices are valid, and these devices can be identified correctly at most times. We can see that the 12 tests have been classified quite precisely among all the 5 smartphones. This figure shows that the system performance does not change much for a larger set. Besides, these results provide encouraging signs that our features selection and classification algorithm is likely scalable to a large number of devices.

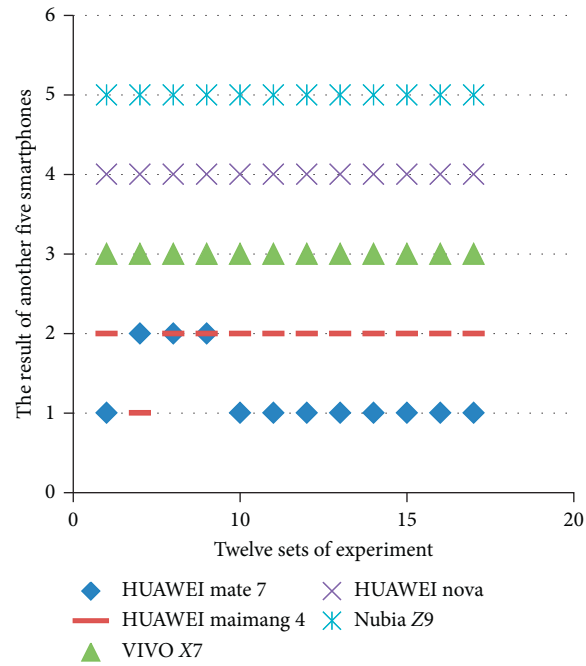

Figure 8: The scalability of our features selection and classification algorithms.

5.7. Impact of the Amount of Tasks. In this section, we conduct an experiment to study how the amount of the tasks impacted on the classification performance. We designed our experiment as we change the tasks' power consumption gradually, which is depending on the amount of the tasks. We compare the performance of different tasks in different devices; then, it is natural to raise the following three questions while displaying our experiment:

(1) Can the power consumption be the same when certain task is carried out in different devices?

(2) Does the power consumption change obviously when the same task is carried out in different devices?

(3) How about fingerprinting a device without any variation about the amount of tasks?

Next, we do a research about the impact of the amount of tasks and try to answer these questions. First, we perform the task which is in variation on the devices, and this experiment

confirms the power consumption is significantly different when the different task is conducted in different devices. Moreover, there is slight difference even for the same task in two devices which is of the same model. Second, we perform experiments keeping devices performing the same tasks, and the results show that they can be discriminated to some extent without any stimulation.

5.7.1. Fingerprinting with Tasks in Variation. In this setup, we design four different tasks in 20 s, such as big number computing, large file reading, large file writing, broadcast transceiver, and other tasks of different power consumption. Each task is executed for 5 s. Moreover, we fingerprint the devices to research the result with stimulation. The accuracy of this setup is shown in Figure 4(a).

Figure 4 shows that the relationship around the accuracy, the sampling time, and the sampling rate. As shown in the figure, we give the result of two sampling rates, once or twice in a second. We can see the sampling rate has little effect on the accuracy, as the height of the red and blue cylinders remains nearly flat. We can see clearly that, for the fingerprints with stimulation, the average F-score can be stable at 85% after the sampling time reaches 25 s. The most important reason is that, the number of eigenvalues extracted when the amount of tasks in variation is more than which are extracted when the amount of tasks is stable, which can directly affect the performance of the recognition. Besides, it is enough for the sampling time to collect the data and generate discriminating fingerprints within 25 seconds.

5.7.2. Fingerprinting with Tasks in Stable. To understand if the fingerprints can be extracted without any stimulation, we conduct an experiment where we keep devices performing the same tasks. The accuracy of this setup is shown in Figure 4(b). We can see that when the amount of the task remains stable, the accuracy of recognition is much lower. After the sampling time reaches 20 s, the average F-score just remains at around 72%, which is because the number of eigenvalues extracted is less than the normal quantity when the devices perform the same tasks all the time. When the sampling time increases gradually, the accuracy rate keeps increasing. From the diagram, we can see that it can maintain at around 74% ultimately, which is discriminated to some extent.

5.8. Impact on Users' Experience. It is inevitable for the programs which is relevant to the user to consider the impact of the program on users' experience. In this section, we use five devices to conduct the impact on the power consumption or users' experience, namely, Samsung Galaxy ON5, Huawei Rongyao 5C, Sony Xperia Z2, Lenovo K3 Note, and Samsung GT-P3110. This experiment is carried out in the laboratory. Besides, we assume that the phone is fully charged. We compute the total power consumption before and after each experiment (we sampled 10 times and calculated the average power consumption of the 10 time tests) and, then, compare the effects of different sampling rates on the power consumption.

Table 3 shows us the impact on the different devices' power consumption of this round. We can see the power consumption of the different smartphones in different sampling rates, as well as the proportion of the power consumption by our experiment and the total capacity detailed. The amount of power consumption during a sampling process is no more than 0.2% of the total power in all the five devices. Different sampling frequencies also have a small impact on the power consumption amount. So, we can draw such a conclusion that the impact of our experiments on the devices itself and users' experience is negligible.

5.9. Impact of Users' Behavior. Previous experiments focus on the fingerprinting performance in daily usage. That is, the fingerprint is computed once Apower is run without closing background apps. In this section, we discuss how the users' behavior (i.e., the foreground app) impacted on the identification performance of PowerPrint. We carry out the device identification with different behaviors. Among all the users' behaviors, we only consider a few simple and common users' behaviors, such as playing games, web browsing, making phone calls, and SMS. Different behaviors may have different impacts on power consumption information, so the behaviors may influence on the final identification accuracy. Before we begin this experiment, we clean up the background process in advance to eliminate the interference factors, and we assume that the subsequent tasks running in the background have little impact on power consumption in practice.

Figure 9 has showed us the impact of users' behavior on the final performance. We can see that the final performance is obviously influenced by the user's behavior. The final performance varies greatly from 45% to 86% when identifying the devices with different users' behavior. The identification accuracy is the worst when playing games and the best when sending messages. Furthermore, the performance is unstable when playing games and is relatively stable when other behaviors are performed.

In this experiment, we also consider the impact of the network; therefore, we carried out two groups of experiments. SMS and phone calls do not require the support of the network environment, but playing games and browsing the web page must be carried out in a network environment. In the picture, the two curves of the same color are the performance of the same user's behavior under different network environments (the network and no-network environment). We can see that, for the same user behavior, the device identification accuracy is better in the absence of the network environment. Therefore, we conclude that some behavior can result in a significant reduction in the final performance, and the effect on the performance is not negligible. Moreover, the network environment can also not be negligible for the effect of device recognition. This is why we used the app designed by ourselves as a source of data, which do not require the network environment.

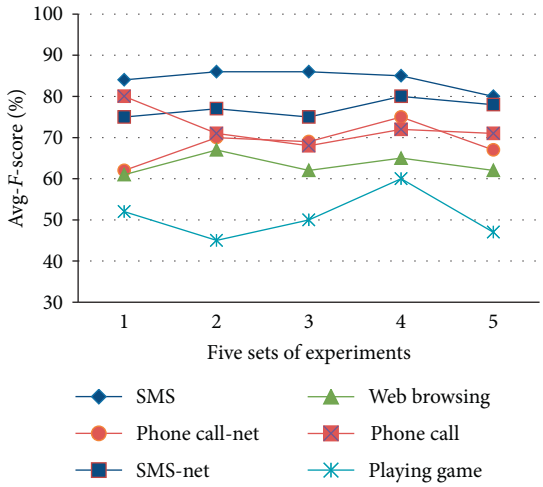

Figure 9: The impact of users' behavior on the identification performance.

To summarize, our evaluation using 15 smartphones and tablets shows that they can be identified robustly leveraging the fingerprints of the batteries. The features we have chosen have a good performance, while others such as current power and the amount of task have little effect on the accuracy. Moreover, while a further study is needed to increase the number of devices to confirm the scalability of our findings, to the best of our knowledge, this can be used to identify the device effectively.

6. Related Work

As everyone knows that, human fingerprints can be used to certify a person well through their uniqueness. Just as the human fingerprints, many things have features that can identify themselves, such as the devices, OS, and commodities. The features which can identify themselves can be called as fingerprints (device fingerprint, OS fingerprint, and commodity fingerprint). We mainly introduce two effective and widely used technologies in identifying mobile and computer technologies: browser fingerprinting and hardware fingerprinting.

6.1. Browser Fingerprinting. Acar has identified users successfully by some other browser techniques in 2013, such as JavaScript-based, extension-based, and header-based. Moreover, it is the first study about the adoption of browser fingerprinting on the web [19]. Wang also identified a user's web activity by leveraging packet sequence information [20]. Similarly, Goethem used the cross-site resource to infer the size of an external package based on the loading time of the web page on the browser side [21, 22]. OS fingerprint is one of the most widely studied fields in browser fingerprints [23].

Browser fingerprints include not only these but also different protocols used in different levels of the OSI model which may also be used as fingerprints to identify operating systems and devices. They rely upon precise classification of such factors as the client's TCP/IP configuration, OS fingerprint, and wireless setting. In normal operation, various network protocols transmit or broadcast packets or headers from which one may infer client configuration parameters. For example, Franklin developed a technique that accurately and efficiently identifies a wireless device that employs IEEE 802.11 networking [24], and Zuo proposed a fingerprinting method based on the communication protocol between a Bluetooth low-energy device and its companion mobile app [25].

However, collection of device fingerprints from web clients relies on the availability of JavaScript or similar client-side scripting language for the harvesting of a suitably large number of parameters. The client-side scripting will be limited, if users use privacy software and browser extensions which block ads and trackers. Moreover, browser fingerprinting was limited to single browsers. If a user switches browsers in a device regularly, browser fingerprint cannot be used to link the user to these browsers. As each distinct client and OS has distinct internal parameters, one may change the device fingerprint by simply running a different browser on the same device. In 2017, Cao put forward a cross browser fingerprinting method, which allows tracking of a user across multiple browsers on the same device [26]. Nonetheless, these techniques for identifying smartphones are not enough and do not work well in the smartphone because the plug-ins in browsers of the smartphones are removed mostly. Furthermore, the fonts are always the same, so the accuracy to identify a smartphone based on these techniques is quite low [6].

6.2. Hardware Fingerprinting. Eventually, hardware fingerprinting appears as an efficient method to identify devices, which can be combined with webGL and canvas element to perform a higher recognition rate. Sensors are an important part of smartphone hardware, and smartphones are equipped with many motion sensors. Naturally, researchers found that many sensors can be used to extract fingerprints to identify the device. Dey considers using accelerometers to generate fingerprints to identify the devices [27]. He has confirmed that the fingerprints arising from hardware imperfections during the sensor manufacturing process cause every sensor chip to respond differently to the same motion stimulus. These fingerprints can be exploited for identifying devices. Besides, mobile sensors such as a gyroscope, accelerometer, and magnetometer are also useful for inferring users' routes and locations [1]. Narain collects information from these sensors, such as angles, curvatures, heading, accelerations, and timestamps. Then, he infers a rough route and compares with the public map to get the user's motion trail. Zhang designed a fingerprinting method based on sensor calibration, which requires only the access to the sensor output [28].

In addition to the common sensors, there are some other hardware components in smartphones which can be used to identify devices. Das and Zhou have confirmed that smartphones' microphones and speakers can be used to identify a device successfully [29, 30]. Kohno introduced the technology that uses clock skews to fingerprinting a physical device [31]. Then, Polcak validated Kohno's technology

again in 2014, which shows that the method is suitable for the computer identification [32]. Identifying a device through these hardware does have a good performance. However, as we have mentioned in Section 1, their implementation process has many restrictions. Users need to do some specific operations to ensure the final accuracy. To avoid user operations, Sanchez-Rola designed a time-based approach, which distinguishes devices via observing the execution time of specific functions [33]. Cheng employed the magnetic induction signals emitted from the CPU module to fingerprint devices, which needs dedicated magnetic sensor [34].

In all the existing work, Olejnik's study is the most similar to ours. Olejnik et al. generated the battery fingerprinting based on the HTML5 Battery Status API to identify users [35]. They compare the power consumption information during the user's twice visiting in a short time and, then, use the information to identify users. However, his research requires quite harsh conditions. First, the identify progress works well only on Firefox browser, and the accuracy of the identifying is lower in other browsers. Second, the methods do not work well after the standardization of the Firefox browser. Moreover, it also requires users to log in a website repeatedly after a few seconds; only in this way it can perform a high accuracy, which is difficult to achieve outside the laboratories.

We also identify the devices based on the smartphone battery. Most importantly, our experiment do not require any conditions except for that the current power need to be higher than 30%, in order to ensure a higher accuracy, our methods is easier to implement, and the accuracy of our identification is also quite high.

7. Limitations and Discussion

Our work is progressed mainly on Android 4.0–6.0. We have not considered other Android versions for the following three reasons. First, the total number of devices in the laboratory is very limited. Second, the permission is restricted after Android 6.0. Finally, Google's market share report on Android version in December 2017 showed the user group of Android 4.0–6.0 accounting for 54.15% in all the Android users [36]. Therefore, our experimental data are convincing because that the user group we faced is huge.

In the future work, we will design an application plug-in to identify the device based on the battery information and use this method to identify the users rather than just the devices. Another direction of our work is to explore the relationship between different apps and power consumption rate.

Let us consider a long and controversial problem, how we define privacy and how to balance the relationship between privacy and convenience. As consumers and their advocacy groups may consider concealed tracking of users to be a violation of user privacy, while computer security experts may consider the ease of bulk parameter extraction to be a browser security hole, and web publisher may consider the collection of users' Internet habits information as a convenience and better user experience for users. Therefore, it is important to find a proper balance between privacy and convenience. In this paper, we are committed to finding this balance, which uses the power consumption information to protect user privacy and guarantee the accuracy of device identification to provide better user services.

8. Conclusions

We have presented a new approach for mobile device identification which allows devices to be recognized without any login operation and authorization. Our approach extracts features from smartphones' power consumption to identify the devices. When selecting our power consumption research objects, we abandoned the various applications in the application store, using our own simple and lightweight application as to remove some interference. At the same time, we use different power consumption tasks to collect more features. Then, we select several kinds of characteristics in variation in both time domain and frequency domain by feature selection algorithm. Moreover, we employ the machine learning algorithm to classify the devices. The results on 10 mobile devices, including 2 tablets, offer evidences that such fingerprints exist, and they are well equipped for device identification. While a conclusive result is needed to be confirmed with plentiful experiment outside the laboratories, we believe that our findings are still an important step for sensor fingerprinting.

Data Availability

The power consumption data used to support the findings of this study are included within the supplementary information file.

Disclosure

A preliminary version of this work appeared in the 23rd IEEE International Conference on Parallel and Distributed Systems (ICPADS) [9].

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant nos. 61772383, U1836202, 61572380, and 61702379, the Joint Fund of Ministry of Education of China for Equipment Preresearch under Grant no. 6141A02033341, and the Science, Technology and Innovation Commission of Shenzhen Municipality under Grant no. JCYJ20170303170108208.

References

[1] S. Narain, T. D. Vo-Huu, K. Block, and G. Noubir, "Inferring user routes and locations using zero-permission mobile sensors," in *Proceedings of Symposium on Security and Privacy*, pp. 397–413, San Francisco, CA, USA, 2016.

[2] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna, "Cookieless monster: exploring the ecosystem of web-based device fingerprinting," in *Proceedings of Symposium on Security and Privacy*, pp. 541–555, San Francisco, CA, USA, 2013.

[3] J. Chen, C. Wang, K. He et al., "Semantics-aware privacy risk assessment using self-learning weight assignment for mobile apps," *IEEE Transactions on Dependable and Secure Computing*, p. 1, 2018.

[4] J. Chen, C. Wang, Z. Zhao, K. Chen, R. Du, and G.-J. Ahn, "Uncovering the face of android ransomware: characterization and real-time detection," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1286–1300, 2018.

[5] S. Englehardt, D. Reisman, C. Eubank et al., "Cookies that give you away: the surveillance implications of web tracking," in *Proceedings of WWW*, pp. 289–299, Florence, Italy, 2015.

[6] P. Laperdrix, W. Rudametkin, and B. Baudry, "Beauty and the beast: diverting modern web browsers to build unique browser fingerprints," in *Proceedings of Symposium on Security and Privacy*, pp. 878–894, San Francisco, CA, USA, 2016.

[7] J. Hua, H. Sun, Z. Shen, Z. Qian, and S. Zhong, "Accurate and efficient wireless device fingerprinting using channel state information," in *Proceedings of INFOCOM*, pp. 1700–1708, Honolulu, HI, USA, 2018.

[8] A. Das, N. Borisov, and M. Caesar, "Tracking mobile web users through motion sensors: attacks and defenses," in *Proceedings of NDSS*, San Francisco, CA, USA, 2016.

[9] J. Chen, Y. Fang, K. He, and R. Du, "Charge-depleting of the batteries makes smartphones recognizable," in *Proceedings of ICPADS*, Tokyo, Japan, 2017.

[10] R. Pangam, "7 leading fraud indicators: from fresh cookies to null values," https://simility.com/blog/data-fraud/device-recon-results/.

[11] Official Documents of Google Android, http://developers.google.com/android/sdk/docs/index.html.

[12] L. Zhang, B. Tiwana, Z. Qian et al., "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in *Proceedings of CODES/ISSS*, pp. 105–114, Scottsdale, AZ, USA, 2010.

[13] M. Gordon, L. Zhang, B. Tiwana, R. Dick, and Z. M. Mao, "An application that displays the power consumed by major system components," http://ziyang.eecs.umich.edu/projects/powertutor/.

[14] O. Lartillot, V. Alluri, and P. Toiviainen, "Mirtoolbox," http://cn.mathworks.com/matlabcentral/fileexchange/24583-mirtoolbox.

[15] A. Pocock and G. Brown, "Feast," http://www.mloss.org/software/view/386/.

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

[17] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing & Management*, vol. 45, no. 4, pp. 427–437, 2009.

[18] Percentage of All Global Web Pages Served to Mobile Phones from 2009 to 2017, https://www.statista.com/statistics/241462/global-mobile-phone-website-traffic-share/.

[19] G. Acar, M. Juárez, N. Nikiforakis et al., "Fpdetective: dusting the web for fingerprinters," in *Proceedings of CCS*, pp. 1129–1140, Berlin, Germany, 2013.

[20] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting," in *Proceedings of USENIX Security*, pp. 143–157, San Diego, CA, USA, 2014.

[21] T. van Goethem, W. Joosen, and N. Nikiforakis, "The clock is still ticking: timing attacks in the modern web," in *Proceedings of CCS*, pp. 1382–1393, London, UK, 2015.

[22] T. van Goethem, M. Vanhoef, F. Piessens, and W. Joosen, "Request and conquer: exposing cross-origin resource size," in *Proceedings of USENIX Security*, pp. 447–462, San Diego, CA, USA, 2016.

[23] Z. Shamsi, D. B. H. Cline, and D. Loguinov, "Faulds: a non-parametric iterative classifier for internet-wide OS fingerprinting," in *Proceedings of CCS*, pp. 971–982, Dallas, TX, USA, 2017.

[24] J. Franklin and D. McCoy, "Passive data link layer 802.11 wireless device driver fingerprinting," in *Proceedings of USENIX Security*, San Diego, CA, USA, 2006.

[25] C. Zuo, H. Wen, Z. Lin, and Y. Zhang, "Automatic fingerprinting of vulnerable ble iot devices with static uuids from mobile apps," in *Proceedings of CCS*, pp. 1469–1483, London, UK, 2019.

[26] Y. Cao, S. Li, and E. Wijmans, "(Cross-)browser fingerprinting via OS and hardware level features," in *Proceedings of NDSS*, San Diego, CA, USA, 2017.

[27] S. Dey, N. Roy, W. Xu, R. R. Choudhury, and S. Nelakuditi, "Accelprint: imperfections of accelerometers make smartphones trackable," in *Proceedings of NDSS*, San Diego, CA, USA, 2014.

[28] J. Zhang, A. R. Beresford, and I. Sheret, "Sensorid: sensor calibration fingerprinting for smartphones," in *Proceedings of Symposium on Security and Privacy*, pp. 638–655, San Diego, CA, USA, 2019.

[29] A. Das, N. Borisov, and M. Caesar, "Do you hear what I hear?: fingerprinting smart devices through embedded acoustic components," in *Proceedings of CCS*, pp. 441–452, Scottsdale, AZ, USA, 2014.

[30] Z. Zhou, W. Diao, X. Liu, and K. Zhang, "Acoustic fingerprinting revisited: generate stable device ID stealthily with inaudible sound," in *Proceedings of CCS*, pp. 429–440, Scottsdale, AZ, USA, 2014.

[31] T. Kohno, A. Broido, and K. C. Claffy, "Remote physical device fingerprinting," in *Proceedings of Symposium on Security and Privacy*, pp. 211–225, Oakland, CA, USA, 2005.

[32] L. Polcak, J. Jirasek, and P. Matousek, "Comment on "remote physical device fingerprinting"," *IEEE Transactions on Dependable and Secure Computing*, vol. 11, no. 5, pp. 494–496, 2014.

[33] I. Sanchez-Rola, I. Santos, and D. Balzarotti, "Clock around the clock: time-based device fingerprinting," in *Proceedings of CCS*, pp. 1502–1514, Toronto, ON, Canada, 2018.

[34] Y. Cheng, X. Ji, J. Zhang, W. Xu, and Y.-C. Chen, "Demicpu: device fingerprinting with magnetic signals radiated by cpu," in *Proceedings of CCS*, pp. 1149–1170, London, UK, 2019.

[35] L. Olejnik, G. Acar, C. Castelluccia, and C. Díaz, *The Leaking Battery: A Privacy Analysis of the HTML5 Battery Status API*, IACR Cryptology ePrint Archive, Las Vegas, NV, USA, 2015.

[36] Operating system share by mobile version, https://www.netmarketshare.com/operating-system-market-share.aspx?id=platformsMobileVersions.