Continuous Authentication on Mobile Devices Using Power Consumption, Touch Gestures and Physical Movement of Users

Rahul Murmuria, Angelos Stavrou, Daniel Barbara, and Dan Fleck

Kryptowire LLC, Fairfax VA 22030, USA
http://www.kryptowire.com

Abstract. Handheld devices today do not continuously verify the identity of the user while sensitive activities are performed. This enables attackers, who can either compromise the initial password or grab the device after login, full access to sensitive data and applications on the device. To mitigate this risk, we propose continuous user monitoring using a machine learning based approach comprising of an ensemble of three distinct modalities: power consumption, touch gestures, and physical movement. Users perform different activities on different applications: we consider application context when we model user behavior. We employ anomaly detection algorithms for each modality and place a bound on the fraction of anomalous events that can be considered "normal" for any given user. We evaluated our system using data collected from 73 volunteer participants. We were able to verify that our system is functional in real-time while the end-user was utilizing popular mobile applications.

Keywords: security, anomaly detection, noise-aware data mining, continuous authentication, behavioral models

1 Introduction

The amount of sensitive data stored on or processed by handheld devices has been on the rise. This is primarily due to a wealth of services that were made available over the last few years including access to emails, social media, banking, personal calendars, navigation and documents. Most commercially available devices employ the use of authentication techniques only at the "entry-point". They require the user to explicitly authenticate before every handheld device interaction but not necessarily when sensitive operations are performed. Thus, although users might be required to use their password often, sensitive data can be misused when an attacker gains physical access to a device immediately after authentication is completed.

There is a plethora of recent work that indicates that password authentication is not appropriate for mobile devices. For instance, Aviv et al. [1] demonstrated the feasibility of smudge attacks using residue oils on touch screen devices. Using this technique, the attackers could extract sensitive information about recent user input, which may include the legitimate user's successful authentication attempt. While intentional misuse of data is a concern, Muslukhov et al. [2] showed that users are also concerned about sharing mobile phones with guest users. Moreover, Karlson et al. [3] conducted interviews of smartphone users and concluded that the entry-point authentication model is too coarse-grained and the type of data that can be considered sensitive varies significantly depending upon the owner's relationship to the guest user. For example, the information that is considered sensitive in the presence of colleagues is different in nature from what is considered sensitive among business clients or competitors. However, protecting every piece of data with additional security mechanisms poses a usability hindrance.

In order to address the shortcomings of the entry-point authentication model, one of the approaches proposed in literature is called *continuous authentication* [4]. This is a process of verifying the identity of the user repeatedly while the handheld device is in use. Generally, continuous authentication methods assume that the process of authentication is unobtrusive. This is necessary as it is impractical to require users to explicitly authenticate themselves at recurring intervals.

In this paper, we propose a technique to authenticate users on handheld devices based on a diverse set of behavioral biometric modalities comprised of power consumption, touch gestures, and physical movement. We are one of the first research groups to propose the use of power measurements as a potential form of authentication for modern (Android) mobile devices. In addition to power behavior, we have implemented touch screen gestures and physical movement as modalities (both are independent behavioral traits in accordance with the survey paper on behavioral biometrics by Yampolskiy et al. [9]). These modalities use measurements from the touch input driver, and from a combination of accelerometer and gyroscope measurements respectively. In this paper, we show that the fusion of these three modalities can be used to corroborate the presence of a legitimate user while capturing long-term characteristics (power modality), short-term physical movement which includes hand vibrations (movement modality), as well as direct device interaction (touch modality).

The proposed approach includes a decision support process (see Fig. 1) where we build models based only on a set of measurements (system readings) for the legitimate user. To detect unauthorized access, we rely on those user-tailored models to provide us with evidence of deviation from the generated user envelope. The decision support process institutes a "test" that requires that no more than n readings within a window of events or time be anomalous, before the user's capabilities are diminished on the device. This threshold can be adjusted to obtain the desired False Reject Rate (FRR) and False Acceptance Rate (FAR), a trade-off that we explored in this paper. We show that every user is prone to infrequent anomalous behavior that is dispersed throughout the user's interaction with the mobile device and the rate at which these anomalies are expected varies for each user. As a result, the number of n readings that are allowed to be

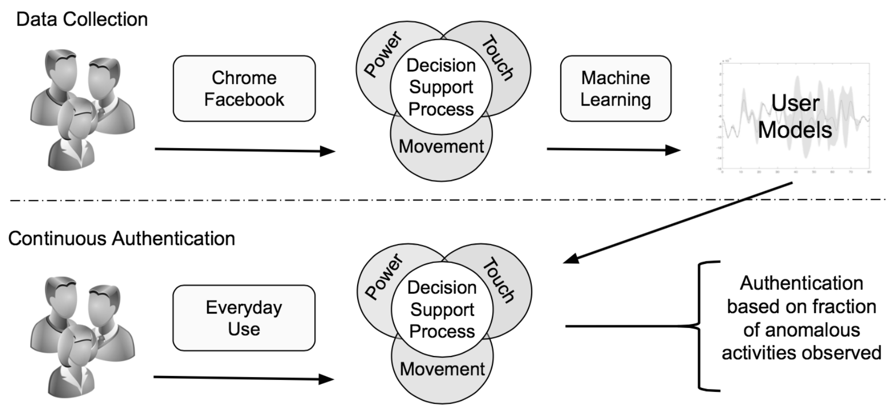


Fig. 1: General workflow

anomalous is part of a user's profile and we show that by using individualized thresholds, we improved performance of our authentication system.

We also show that authentication accuracy is affected by application context; any user's behavior differs from application to application. When a user is playing a game, the digital footprint that the user leaves behind in terms of power consumption, touch screen gestures or physical movement is expected to be significantly different from when the user is sending text messages. We present the performance of our system on two popular mobile applications – Google Chrome and Facebook – and show that by ignoring the application context, there is a clear degradation in identification accuracy.

The rest of this paper is organized as follows. Section 2 is a brief review of research publications that are related to the work done in this paper. Section 3 discusses the implementation details of our data collection architecture. Section 4 provides the experimental design and protocol used while collecting data from volunteers. Section 5 describes how the data was prepared for analysis. Section 6 identifies the algorithms employed for the task of continuous authentication of users in mobile devices. Section 7 presents a performance evaluation of the algorithms on the data collected. In Sect. 9 and Sect. 10, we suggest further research directions and conclude this paper.

2 Related Work

Riva et al. [6] presented an architecture that grants users access to any content on the device only when the authentication system evaluates the device operator's level of authenticity to be higher than what is required to access that content. Their system utilized face and voice recognition, location familiarity, and determining possession by sensing nearby electronic objects as signals to establish the legitimate user's level of authenticity. They motivated their work with a user study that explored models where there are at-least 3 levels of security:

public, private, and confidential. With this framework, they tested nine users, and were able to reduce the number of explicit authentications by 42%. While the use of environmental signals in the authentication system help in reacting to device theft, it does not solve the problem of data misuse while in familiar surroundings.

Shi et al. [5] presented an approach that built on the concept that most users are habitual in nature and are prone to performing similar tasks at a certain time of the day. The researchers collected a wide range of behavioral information such as location, communication, and usage of applications, in order to create a user profile. Their method is based on identification of positive events and boosting the authentication score when a "good" or habitual event is observed. The passage of time is treated as a negative event in that scores gradually degrade. One of the main caveats with this work is that it is trying to model what good geographic locations, phone calls, text messages, and website urls are. The data collected is highly intrusive in terms of privacy. They further model all good events as ones that are expected to be performed at a certain time of day, which is an assumption of habit that is not proven in the literature. However, if this works well, it can also be incorporated as another voter in our approach.

Kwapisz et al. [10] published a system to identify and authenticate users based on accelerometer data. They used a dataset of 36 users, labeled according to activities such as walking, jogging, and climbing stairs. These labels were used as context and the authors presented analysis with and without these labels. For feature extraction, the authors divided the 3 axes readings of the accelerometer into windows of 10-seconds, and for each window they extracted features such as mean, standard deviation, resultant, and binned distribution. For identification, the authors performed a 36-class classification, whereas for the task of authentication, the authors reduced the problem to a 2-class problem. They achieved a classification accuracy of 72.2% for 10-second windows. While they concluded based on their results that it is not critical to know what activity the user is performing, their dataset was generated by users repeating a limited set of predefined activities. In contrast, instead of using real-world activities as context which requires manual labeling of the data, we consider the applications being used by the users as context which can be automated. We were able to show that user behavior is indeed subject to application context.

Frank et al. [7] performed a study where they collected touch screen behavior of 41 users and designed a proof-of-concept classification framework to examine the applicability of screen touches as a behavioral biometric. For their data collection, they designed a custom application that allowed users to swipe vertically and horizontally. Using this dataset as baseline, they matched users based on how they perform the same task in testing phase. For analyzing this data, they separated each type of stroke and matched 30 different features (such as mid-stroke area covered, 20%-perc. pairwise velocity, and mid-stroke pressure) extracted for each stroke. Their study resulted in mis-classification error rates in the range of 0% to 4%. Although the researchers were able to demonstrate good performance while matching gestures in a controlled environment, they limited their analysis to vertical and horizontal swipes on their own application. The analysis depended on the concept that users are performing repetitive predefined tasks. In the real world, different users can perform a large variety of tasks that cannot be modelled individually. This technique clearly has a problem of scale that was not addressed in the paper.

Both Kwapisz et al. and Frank et al. created a two-class problem where the adversarial class had data points from other users. Contrastingly for traditional computing devices, Killourhy et al. [11] and Shen et al. [12] published a comparison of various anomaly-detection algorithms for keystroke dynamics and mouse dynamics respectively, limiting the discussion to 1-class verification due to lack of availability of imposter data in the real-world. In truth, the number of classes representing adversaries is unbounded. Modeling adversaries into a fixed number of classes leads to overfitting and lack of generalization, which results in poor performance of the deployed system.

Bo et al. [8] attempted to create a model to specifically identify transitions or change of hands between the device owner and a guest who may or may not be a known entity. The researchers model a user by leveraging her/his touch screen interactions as well as device feedback in form of vibrations into one single model. Though the device initially has only the owner's behavior data, and a one-class SVM model is trained to provide a judgment whether a new action belongs to the owner or not, the researchers quickly evolve this into a two-class SVM model by collecting guest user's data into a second class. They assign a confidence to this judgment, and the conclusion confidence increases with a continuous sequence of consistent judgments. When a change of user is detected, the sequence of consistent judgment is dissolved. While the authors demonstrate 100% identification accuracy within a sequence of 5 to 10 observations, the analysis fails to show tolerance with anomalies, or in other words, the inherent noise. There is no detailed discussion about finding a new guest user verses anomalies committed by the device owner. Further, their model also does not consider different user behavior in different usage scenarios, such as the application context considered in our work.

Although we know of no other biometric systems based on power consumption as an identifier, there is widespread research in the area of power model generation on electronic devices. Zhang et al. [13] presented an automated power model construction technique that uses built-in battery voltage sensors and knowledge of battery discharge behavior to monitor power consumption of each application on an electronic device. They achieved an absolute average error rate of less than 10%. Murmuria et al. [14] demonstrated that the power consumption by individual device drivers on a smartphone varies by state of operation of that particular device driver. Shye et al. [15] presented a power estimation model by leveraging real user behavior. They presented evidence that system power consumption patterns are highly correlated with user behavior patterns, but stopped short of trying to profile users on this basis.

3 Data Collection Architecture

The various hardware components available on a smartphone include: touchscreen, accelerometer, gyroscope, voltage sensor, current sensor, and battery. Each of the components has device drivers, which report sensory statistics to the kernel. The nature and frequency of this data depends on the individual hardware component. Some components require registering event listeners with the Android API, whereas other components require polling the system for data.

For the application context, the name of the application in focus was recorded using the Activity Manager API while users interacted with the mobile device. This is used to determine the context of the model.

For the power modality, in order to determine the power consumption resulting from the activities performed by the user, we used the built-in voltage and current sensors available as part of the battery driver on smartphones. The power drained from battery is proportional to load. While the batteries decay nonlinearly (reflected directly in the voltage readings), the current readings offset this effect in order to deliver the required power. Therefore, in order to model power consumption, it is sufficient to capture voltage and current. The sensors report the voltage and current to the operating system's kernel in units micro-volts (uV) and micro-Ampere (uA) respectively. While the voltage reading depends on the battery charge and changes gradually between 4.35 Volts to 3.2 Volts, the current reading directly depends on the amount drawn by the Android Operating System depending upon what activities are being performed. As a result, while we poll the voltage every 5 seconds, the current reading is polled every 1 second and we take an average of the recorded values every 5 seconds. Using these readings, we can calculate the average power consumption every 5 seconds.

For the movement modality, readings were recorded using the SensorEvent API, which is a part of the standard Android SDK. Depending upon which hardware sensors are present on the Android device, the API has the capability to report values from the following sensors: accelerometer, gyroscope, magnetic field, light, pressure, and proximity. For our analysis, we gathered movement readings from both accelerometer and gyroscope sensors. The accelerometer sensor measures acceleration in SI units (m/s^2) along the device's local [X, Y, Z] axes and the gyroscope sensor measures rate of rotation in SI units (rad/s) around the device's local axes.

For the touch modality, the user-level touchscreen gestures of key-press, pinch and zoom, swipe, and other gestures are all reported as multiple events to an input driver. The touch event interface exists as a character device under `/dev/input` and can be read by any program that has permissions to read it. For security reasons, this input driver is a protected interface. Only vendor programs are given this permission on any unmodified commercially available Android device. The device driver reports the following information: [X, Y] coordinates, number of fingers touching the screen, pressure of each finger, and touch area of each finger. We capture the events along with precise timing information directly from these low-level event streams and reconstruct it back to user-level gestures.

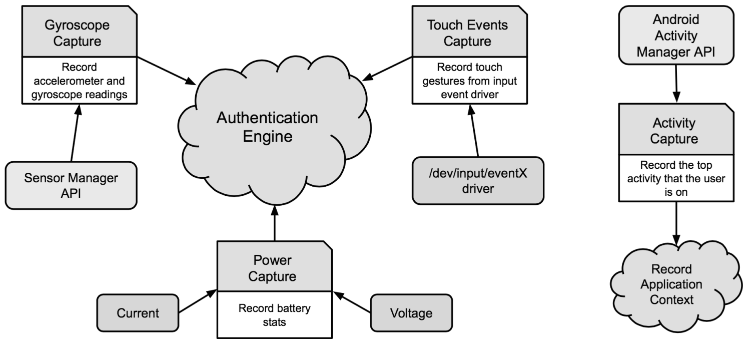

Gyroscope Capture
Record accelerometer and gyroscope readings
Sensor Manager API
Authentication Engine
Touch Events Capture
Record touch gestures from input event driver
/dev/input/eventX driver
Power Capture
Record battery stats
Current
Voltage
Android Activity Manager API
Activity Capture
Record the top activity that the user is on
Record Application Context

Fig. 2: Smartphone sensor data collection framework

Data Collection Tool. Figure 2 shows the smartphone sensor data collection architecture. There are 4 services running in our data collection application: PowerLogger, TouchLogger, GyroLogger, and ActivityLogger.

– Service 1 (PowerLogger): It collects the Voltage, Current and Battery Charge from the battery driver (via *sys* filesystem).
– Service 2 (TouchLogger): This service reads the input events driver to collect system-wide touchscreen events. The touch driver is protected by a system user group "input".
– Service 3 (GyroLogger): This uses the Android API to collect both gyroscope and accelerometer sensor data using a SensorEventListener.
– Service 4 (ActivityLogger): This service uses an Android API to record the user activity on the device. Specifically, we record the top running application, incoming and outgoing calls, and screen-off and screen-on events.

All these services are active during both training and testing, and the overall system power consumption is guided by user's behavior plus a constant from these services. Therefore, our measurement tool does not adversely impact the power profile we generate for a user. We took measures to make our services robust, such that we keep this constant noise in the power consumption readings small, regardless.

4 Experimental Design

When performing a study with volunteer participants, the results obtained depend strongly on the quality of the data collected. It is vital to understand any sources that can cause potential variance in the data for a specific user and to retain data in a uniform format using uniform devices. While our profile generation algorithms do not require such precautions, this step is needed in order to compare the datasets and evaluate the performance fairly.

To achieve uniformity of measurements, we used the same device (Google Nexus 5 Model:LG-D820) for all users who volunteered for this study. Further, all data collections were performed on Android version 4.4.4 (Build number KTU84P). Studying the effects of collecting data across different smartphone models or software versions was not attempted. We also did not use any tablet devices.

In total, 73 users volunteered for this experiment. The experiments were designed to collect data from each volunteer participant for two 45 minute sessions. We assumed that a user's behavior varies while using different applications on the smartphone. All volunteer participants were allowed to use Chrome and Facebook, which are standard applications available on Android phones. They were not restricted in terms of what tasks they can perform using those applications. The application currently in use was recorded and user profiles were generated keeping separate data for each context.

We did not want environmental interference within our data and therefore, the user was restricted to remain within a room. Each user was asked to use the two pre-chosen applications for 20 minutes each with a break of 5 minutes for instructions. This session was repeated on two different days in order to capture the user's behavior effectively. This would total up to 80 minutes of actual smartphone usage data for each user. All tasks were performed while sitting down. Although no user walked or performed any other physical activity, the smartphones were subject to significant movement due to typical usage of the device.

Our experimental setup does not emulate real-world use of the chosen applications. However, related research on mobile authentication techniques relies solely on evaluating touch or movement patterns on custom designed applications or in much more restricted environments where users are asked to perform specific actions repeatedly (swiping or moving in a direction). One of the contributions of our work is the verification of the idea that the application itself plays a significant role and alters the user behavioral patterns. Our results indicate that previous results on active authentication are not applicable in real-world scenarios.

As part of our experimental protocol, we instructed the volunteers to login to Facebook first. No touch or other sensory data was collected during this first step. All other activities the volunteers performed on the smartphones did not involve entering a password of any nature. Each user was assigned a pseudonym with the convention `Sxx`, where the `xx` is a digit between 1 and 100. The real names of the users were not retained. We also did not record any user-generated content outside of the sensory data. No web traffic or URLs were recorded. No attempt was made to capture the content that a user saw on the screen. We recorded data from all the sensors concerned into files for each modality. These files were stored in the external storage directory of each smartphone. Upon completion of a user's session, we extracted that data out from the smartphone into our data store where we analyzed the data.

All our volunteer participants were aged between lower 20s and upper 40s, covering a variety of ethnicities and nationalities. Some of our participants were not regular smartphone users. We did not attempt to discriminate who volunteered, beyond requiring them to have an active Facebook account. Our research required behavioral data of human subjects and necessary approvals were acquired from the Institutional Review Board (IRB).

5 Data Preparation

5.1 Feature Engineering

After collecting the raw data, we performed feature extraction on the data from each modality. Currently, there are no universally accepted set of features that represent individual events for each of the modalities. For the purposes of this research, we selected our feature set based on our own experience with the data.

For the power modality, the activities performed by the user were represented in milliwatts (mW) using the voltage and current readings. These power consumption readings were used in our algorithm as a time-series.

For the movement modality, the recorded events were divided into small windows of time where we can measure properties related to the group of events. Let the size of this window of time be w units, then we employed the use of a sliding window technique that moved $w/2$ units in direction of increasing time for each subsequent record in our prepared movement dataset. As a result, every event in the raw data contributed to 2 windows in the movement dataset. We made this choice because it is difficult to determine the start and end of any particular movement gesture, and using non-overlapping windows would result in loss of precision. For the purposes of our analysis, the data associated within each window frame can be referred to as one movement gesture. Each movement gesture was encoded as a sequence of events; each event is a vector of sensory signals as described in Sect. 3. Fourteen features were extracted from each movement gesture. These features include mean and standard deviation along each axes and resultant magnitude of all axes, for both accelerometer and gyroscope readings.

For the touch modality, the recorded events were aggregated into touch gestures. Each gesture is a sequence of touch events that begins with touch down of the finger and ends with lifting the finger. Five features were extracted from each touch gesture. These include: duration, end-to-end distance, end-to-end direction, average pressure, and average touch area.

5.2 Data Cleaning and User Selection

Since each of the features we collected for touch and movement modalities had different units, we standardized the dataset using the mean and standard deviation of each feature over the entire dataset of all users.

After extracting features, the data was divided according to *application context*. The ActivityLogger in our data collection tool inserted place-markers in the data whenever the user switched from one application to another. As part of pre-processing the data, only those events were extracted, that were generated while using the application for which the user profiles are being created. As a result, multiple datasets were created, one for every combination of the users, applications, and modalities.

We then analyzed if a similar amount of data was collected for every user. As we mentioned in Sect. 4, every user was given a fixed amount of time to use the device. Users who generated very small datasets did not perform enough actions on the device for us to model. Further, users who generated too much data expectantly did not follow a normal use-case and would not match themselves under different circumstances. Therefore, any user who generated data of abnormally large or small sizes was discarded. In order to compute this, we first merged the data collected for each of the 73 users over the two days of experiments. The number of records was tabulated for each of the 6 datasets (2 applications and 3 modalities) for every user, and the means and standard deviations were computed. We then removed those users who had any dataset with sizes more than or less than 2 standard deviations from the corresponding mean. With this method, 59 users were selected who had comparable sizes of data. In order to prepare the baseline, 60% of each user's dataset was used. The algorithms we used to train a model using this data are described in Sect. 6. As a result, users' profiles were created. The remaining 40% of datasets for each user were used to test this model.

6 Analysis to Compute Authentication

We view the authentication task as one of determining whether the current stream of measurements of a given kind follows the same distribution as those obtained in a baseline session for a given user. As such, we employ algorithms that are capable of detecting outliers with respect to the baseline distribution and place a bound on how many outliers we can allow if we assume the test data follows the same distribution of the baseline. Exceeding this bound is an indication of the user being an impostor.

We separate the analysis techniques in two groups. The first, utilized for multivariate data (e.g., the data collected from touch and movement modalities), is an adaptation of an outlier detection algorithm first published by Barbara et al. [16] and described in Sect. 6.1. The second, utilized for univariate time-series data (e.g., power measurements) is based on a technique reported by Keogh et al. [17] and is explained in Sect. 6.2.

6.1 Strangeness-Based Outlier Detection

Strangeness-based Outlier Detection (StrOUD) algorithm, utilizing a machine learning technique called transduction, was devised by Barbara et al. [16] to detect outliers in datasets. Transduction is a machine learning technique based in the process of reasoning from specific (baseline) cases to specific (testing) cases. This is in contrast to induction which reasons from specific cases to rules that can be applied to test other cases. The method was invented by Vapnik

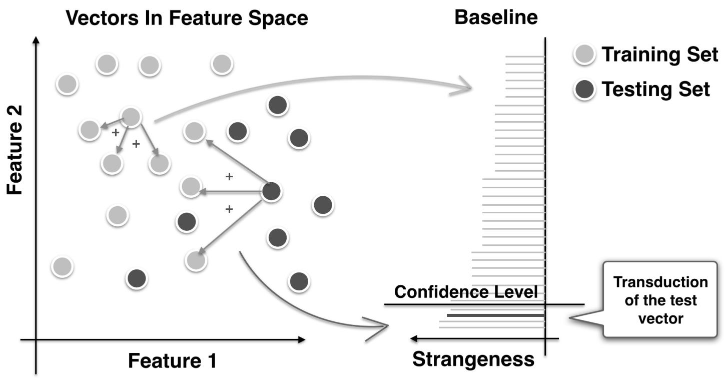


Fig. 3: Strangeness-based outlier detection

et al [18], motivated by his view that induction requires solving a more general problem, while transduction requires solving a more specific problem, which is easier, and, in many cases, more accurate.

Transduction is carried out by placing a point in a known sample distribution of data and using hypothesis testing to determine whether it is a good fit or not. To that end, a measure of uniqueness, or strangeness is used for every point in the distribution, including the one we are trying to fit. Strangeness is defined by a function that measures the uniqueness of that point. Vapnik et al. utilized transduction in the context of classification, or supervised learning, to properly place new points in their rightful class. The technique is called Transductive Classification Machines or TCM. The transduction methodology does not build general models. The 'models' are captured in the distribution of uniqueness values for each class.

Statistical hypothesis testing which aims to prove or disprove one of the following hypotheses: the null hypothesis that says the test point is a good fit in the distribution (and in the case of TCMs, whether the point belongs to the class represented by that distribution), and the alternative hypothesis that says the point is not a good fit. The test is performed by computing a p-value (measure of randomness) as the fraction of the points in the sample distribution whose strangeness is greater or equal to that of the test point. If this p-value is less than the complement of the confidence level desired for the diagnosis, the alternative hypothesis is accepted.

StrOUD borrows the idea of TCM with an important change: nature of the strangeness function utilized. The goal in StrOUD is to find anomalies (not to classify points), so, the strangeness function should be a measure of how anomalous a point is within a distribution. Given a sample distribution, or baseline of observations, the strangeness of the j^{th} point x_j can be computed as the sum of the distances to the k nearest points in the baseline data. Figure 3 presents an illustration of two sets of data (yellow and blue) in feature space. Strangeness calculation has been demonstrated in the figure for a point each from the training

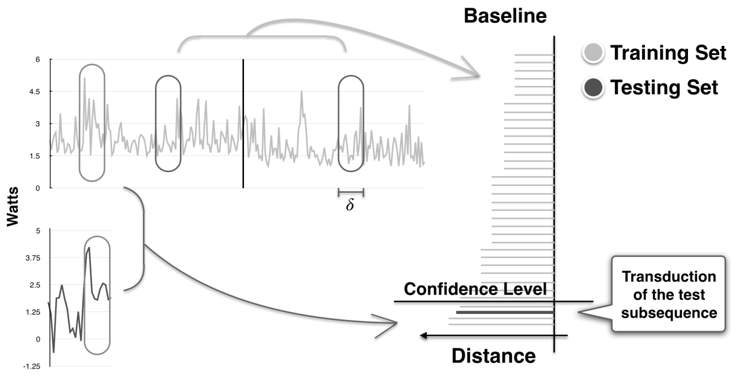


Fig. 4: Discord-based outlier detection

set and the testing set. Equation 1 shows the definition of strangeness utilized by Barbara et al. [16].

$$s_j = \sum_{i \in k} d(x_j, x_i) \ . \tag{1}$$

Calculating all the strangeness measures for points in the baseline and sorting them in ascending order returns a sample distribution of strangeness (shown in the Fig. 3 with the width of the bars representing the strangeness values of each point). For a given new point, its strangeness needs to be computed and its place on that distribution measured, as the fraction of points (including itself) that have strangeness equal or greater than its own. As stated before, that fraction is a measurement of randomness in the form of a p-value, which serves as the basis for hypothesis testing.

6.2 The Discord Algorithm

The power measurements are viewed as a time-series and for this modality an algorithm designed by Keogh et al. [17] that allows the discovery of *discords* on that kind of data was employed. A discord in a time series is a subsequence of the series whose distance to the closest subsequence (of the same size) is maximal. A discord is a particularly desirable indicator of anomaly, because it only requires a very intuitive parameter for its discovery: the size of the subsequence.

The discord idea is used with the power modality in two phases. In the first, the goal is to obtain a distribution of measures that represent the uniqueness of a time series, as a baseline distribution. To that end, the power baseline data collected for a user is divided in two parts. The first, of size `m` is used as a basis to find discords in chunks of the second part. In the Fig. 4, this step has been illustrated by using the training set time-series partitioned using the vertical line. Given a fixed size of the subsequence δ, we compare a subsequence from the second part with all subsequences in the first part, and the distance to its closest neighbor is returned. Doing this over the entire second part of the dataset results in a distribution of distances that can be sorted in ascending order (shown on the right side in the figure). This distribution is viewed as playing the role of the strangeness distribution.

When analyzing test data, after receiving δ observations, the algorithm computes the distance of that set of δ observations to the first part of the baseline time series (of size `m`). Doing so, the algorithm obtains a new distance to the test data's closest neighbor and proceeds to transduce that distance into the strangeness distribution, to analyze whether that subsequence is an anomaly or not. This is repeated for every new observation in the test data (always considering a window of size δ that spans the current observation).

6.3 User Diagnosis

After calculating the sorted distribution of strangeness any future incoming point is diagnosed using this distribution, which represents a user's profile. In an experimental setting, many datasets were tested against each user's baseline; some of these datasets came from the same user and some of them from users other than the one that generated the baseline. Such setting produces a matrix where each column and row represents one user. Every entry represents the probability of committing an anomaly for the corresponding pair of baseline/test user data set. This matrix is called the confusion matrix. The smaller the score, the better the testing data matched the baseline. Examples for this matrix are shown in Tables 3, 4, and 5 under Sect. 7.

The mere presence of an anomaly does not conclude presence of an imposter. The fraction of anomalies in the test dataset is an indication of whether the set belongs to the original distribution or not. Setting a threshold on the maximum probability that can be observed and still consider the data as coming from the same distribution of the baseline gives a way to diagnose a user as an impostor or not.

From this point onward, there are two ways to proceed while choosing thresholds. The first is to select a general threshold and diagnose as reject every matrix entry whose value is bigger than the threshold. If the reject occurs in a case for which the row and column are from the same user, it is a false reject (the model is saying the user is not who they say they are, while the truth says otherwise). If the reject occurs elsewhere, it is a true reject (the model is correctly saying this user is different than that of the baseline). After computing the rates at which these two events occur, False Reject Rate (FRR) and False Accept Rate (FAR) can be calculated as shown in 2.

$$\begin{aligned} FAR &= 1 - \text{ True Reject / Total Reject Cases} \\ FRR &= \text{ False Reject / Total Accept Cases} \quad . \end{aligned} \tag{2}$$

Varying the threshold for fraction of anomalies allows computing pairs of values for FRR and FAR for each threshold, and plotting the Receiving Operating Characteristic (ROC) curve. If a single column from the matrix is used

Table 1: Parameters selected for our algorithms

Parameter	k	δ	m	$conf$
value	3	12	60%	90%

to calculate the FRR and FAR, the ROC curve represents performance of the corresponding user's model. This requires having more than one test set that comes from the same user represented in the column (otherwise the computation of FRR is trivialized). If the entire matrix is used, the ROC curve represents overall performance of all the models for every user.

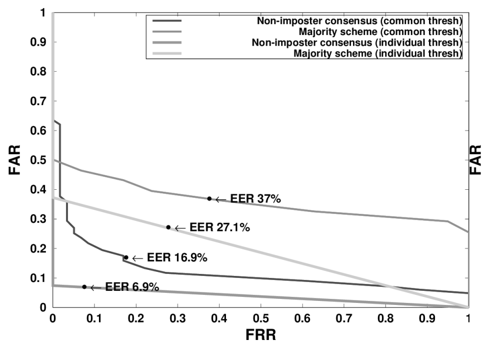
The second alternative is to utilize an individual threshold for each user. These thresholds are calculated using the fraction of anomalies of each user (which represent the rate of anomalies that are "normal" to every user). In this case, from the confusion matrix, the overall FRR and FAR are computed by varying threshold values per column (i.e., per user) and the ROC is reported with each FRR/FAR pair resulting from a vector of threshold values. Experiments show that selecting individual thresholds result in much improved ROC plots, and thus, better models.

Each modality produces its own confusion matrix. To calculate the overall result, we used two schemes to calculate the ensemble: *majority scheme* and *non-imposter consensus*. The majority scheme requires at least 2 out of the 3 modalities to vote for having found an imposter. The non-imposter consensus requires 3 out of the 3 modalities to vote for having found the same user and any other vote results in a declaration of imposter.

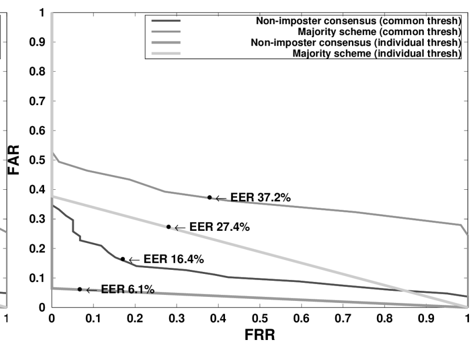
7 Results

We measured our system's performance in terms of the commonly used metrics: False Acceptance Rate (FAR), False Rejection Rate (FRR) and Receiver Operating Characteristic (ROC) curve. We defined these terms in Sect. 6.3 in the context of our analysis. Additionally, we make use of another metric called Equal Error Rate (EER), which is the rate at which the FAR and FRR are equal. EER is a widely used metric to determine the overall performance of an authentication system regardless of the choice of parameters.

The parameters we selected for the two algorithms discussed in Sect. 6 are shown in Table 1. The value of k in the StrOUD algorithm was selected to be 3. Other selections in the neighborhood of 3 did not result in a significant difference in the overall performance. The δ in the power consumption data was selected to be 12, which corresponds to time-series window of 1 minute. Other window sizes of 30 seconds and 5 minutes performed poorly compared to the 1 minute window. The parameter m splits the power data in two chunks during baseline generation phase of the discord algorithm. A split of 60-40 was considered appropriate, but other combinations can be explored in future. The confidence level determines how strictly the user diagnosis phase marks records as imposter. Different values between 85% and 99% were tested, and the best performing level was chosen.



(a) Using Chrome App



(b) Using Facebook App

Fig. 5: ROC from ensemble model on all users

Table 2: Distribution of FAR of users at ≤1% FRR

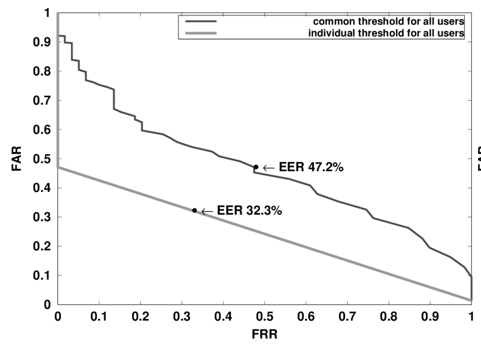
Range	0–2.5%	2.5%–7.5%	7.5%–12.5%	12.5%–50%
Users (Chrome)	38	10	4	7
Users (Facebook)	45	7	0	7

Our analysis of the experimental results demonstrate that our approach using the ensemble of modalities allow us to identify imposters with an Equal Error Rate between 6.1% and 6.9% for training times that vary between 20 and 60 minutes. The first response time to authenticate is as low as 2 seconds for the gyroscope modality, 1 gesture for the touch screen modality and 60 seconds for the power modality. Subsequently, each user action will produce a new authentication score in real-time.

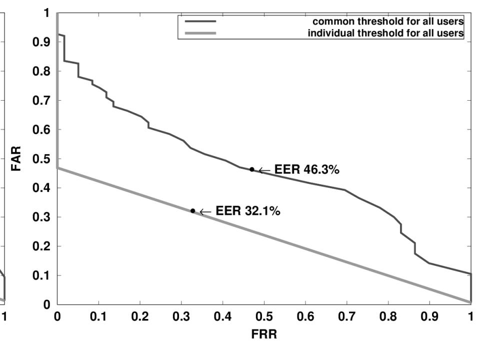
We discussed our use of common thresholds and individualized thresholds in Sect. 6.3. Further, the voting schemes we used to create the ensembles is described in Sect. 6. Figure 5a shows the detection performance results for Chrome and Fig. 5b for Facebook. It is clear from the ensemble plots that for both voting schemes, the *individual thresholds* for each user gives a significant improvement to the predictions. Further, our voting scheme of *Non-imposter consensus* is consistently outperforming the *majority scheme*. By requiring the modalities to agree by consensus when a legitimate user is present, we placed a higher cost on acceptance and thereby improved the overall performance.

We examined the distribution of False Accept Rates (FAR) by 'clamping' the False Reject Rates (FRR) to 0.01 (i.e., 1%). The results are tabulated in Table 2. The results follow intuition that we have very few users that can cause an FAR value to be higher than the average. We believe that this can be rectified with use of additional biometric modalities complementary to the three we have developed.

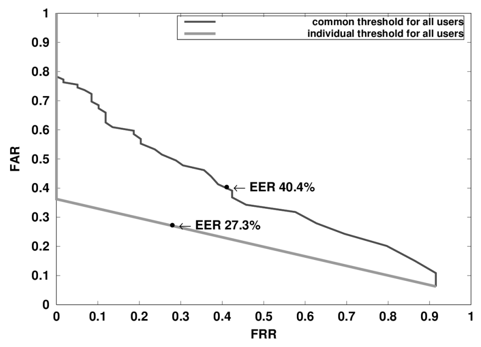
If used separately, each modality cannot generate models that offer good performance in terms of accuracy and classification results because each modality



(a) Movement modality (gyro./accel.)



(b) Touch modality



(c) Power modality

Fig. 6: ROC for Chrome App captured on all users

is a weak classifier. Put it simply, users can happen to closely resemble one of the modalities but it is extremely rare that they do so at the same time for all three modalities given our experimental results. As a result, because the modalities have non-overlapping weaknesses, together in an ensemble they form a strong identifier. We use ROC curves produced from the data of Chrome application as an example, to demonstrate that (see Fig. 6). The performance ranges in EER from 27.3% to 32.3% even with individual thresholds. Figure 5a is the corresponding ensemble plot that shows EER of 16.9% for common thresholds and EER of 6.9% with individual thresholds which are significant improvements. Modality plots for the Facebook application showed comparable performance and have been omitted here.

Results also depend on the Application context. See Table 3, Table 4 and Table 5 that show the performance of the power modality for 5 randomly selected users. These confusion matrices have been created using the technique discussed in Sect. 6.3. The light gray color represents the rate of anomaly considered "normal" to the legitimate user. Using our per-user thresholds method, if any cell in the column happens to have a fraction of anomaly less than the value in

Table 3: Randomly selected 5 users from Chrome App for power modality

		Baseline Users				
%		A	B	C	D	E
Test Users	A	3.6	15.6	19.2	10.7	28.5
	B	30.1	12.2	14.8	8.5	32.5
	C	17.1	4.2	2.5	5.0	35.8
	D	42.1	2.7	21.0	12.1	47.8
	E	19.8	8.5	8.9	0.8	6.7

Table 4: Randomly selected 5 users from Facebook App for power modality

		Baseline Users				
%		A	B	C	D	E
Test Users	A	20.4	40.8	25.1	30.6	5.7
	B	3.6	5.4	52.3	11.6	1.8
	C	13.5	71.2	7.5	90.7	2.8
	D	2.2	24.8	71.1	7.4	5.5
	E	11.9	61.4	21.5	64.0	18.2

Table 5: Randomly selected 5 users from mixture of both apps for power modality

		Baseline Users				
%		A	B	C	D	E
Test Users	A	10.8	11.3	11.3	5.4	16.3
	B	8.8	18.3	17.1	2.7	13.3
	C	7.8	3.3	15.8	2.0	15.3
	D	10.6	3.2	17.0	41.6	22.3
	E	3.9	4.5	4.7	0.5	16.0

the diagonal (colored light gray), then these would represent False Accepts, and we have colored them dark gray. It is clear from observation, that the number of dark gray boxes greatly increases in Table 5. User's behavior is not similar across Chrome and Facebook applications and thus the identification accuracy is dramatically reduced when we mix the datasets. In Fig. 7, we present the ROC curves for overall performance of the system on the combined data of Chrome and Facebook with the context removed. It can be observed that the overall performance has deteriorated.

8 Lessons Learned

It is not uncommon to encounter very noisy data when mining in the real-world. The same is the case with the behavioral biometrics dataset collected for this research. Despite moderately controlling the environment during data collection, the rate of anomalies due to noise is high, in comparison to misclassification error. Most research work in continuous authentication resort to one of two techniques: (1) Data cleansing by removing chunks of data whose class predictions contradict the ground truth during training, and (2) Assume the data to be clean, and consider the contradictions as misclassification error during testing. In truth, the nature of the data is changing so constantly that it is required to not only assume but explicitly model the noise. This research, therefore, considered the rate of anomalies generated by a user while testing against his own baseline to be a virtue of his own user profile. As seen in Table 3 and Table 4, the rate of anomalies considered "normal" for different users cover a wide range of values.

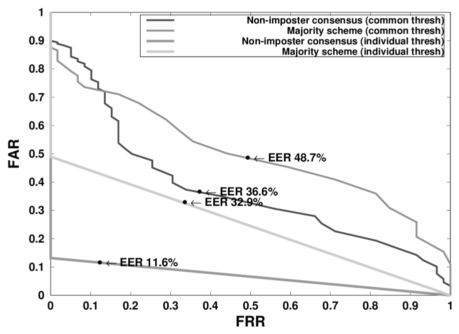


Fig. 7: ROC from ensemble model without application context

This indicates the assumption could be true, resulting in the need to build models that are error-aware.

This research also showed that authentication accuracy is affected by application context; any user's behavior differs from application to application. The performance of the authentication engine therefore depends not only on which mobile application is being used (application context), but also on whether the mobile application can be modeled well or not. Different categories of applications can be identified, as described below, that will show high false accepts and false rejects depending upon how users are meant to interact with the application.

1. Randomized UI

   Mobile Applications that have completely randomized UI will have high false rejects. Some examples include Game applications that appear randomized on all of the modalities used in this research.
2. Static UI

   On the other hand, apps that have completely static UI can have very high false accepts. Some examples include Camera, Flashlight, Sound Recorder and Navigation. While these apps appear static from the point of view of the chosen biometric modalities, some of them can be incorporated into the system by introducing modalities such as voice recognition.
3. Mixed UI

   In general, apps that have a mix of UI inputs yield the best results for authentication. Chrome and Facebook were tested in this research, but other apps in this category include email and word processors.

Since different applications need different levels of security, it can be argued that some apps that need higher data security can employ the use of a rich set of user interactions in order to benefit from an overall improved authentication performance from their users.

9 Future Work

As discussed in Sect. 4, we presented results using a dataset that was generated while controlling environmental interference. One direct progression of this work is to study the impact on performance when allowing volunteers to perform their daily routine tasks. Such a test would warrant the need for much longer data collection sessions where the volunteer uses the mobile device over multiple days. In addition, we believe that further investigation of the scalability of the individualized parameters when we increase the number of users, apps, and modalities is warranted. The current testing was done to determine the efficacy of identifying users through their device interactions. Future work will assess the usability of the approach now that the efficacy has been established.

10 Conclusion

We have introduced a novel system that performs authentication on Android mobile devices by leveraging user behavior captured though three distinct modalities – power consumption, touch gestures, and physical movement (using both accelerometer and gyroscope). To the best of our knowledge, we are one of the first research groups to propose the use of mobile power measurements and leverage the application context in the authentication process. We further demonstrated that by using individualized thresholds for rate of anomalies expected from every user, we were able to improve performance significantly. To that end, we implemented a full set of algorithms and applications for the measurement, evaluation, and deployment of the framework to Nexus 5 Android phone and demonstrated our capability to perform continuous authentication in real time. By leveraging data collected from 73 volunteer participants, we evaluated our system while the end-user was utilizing two popular applications – Chrome and Facebook. We were able to achieve good performance with an equal error rate between 6.1% and 6.9% for 59 selected users who had generated sufficient data for evaluation. Given that the approach in this paper solves the problem of noise and context in very deployable ways, it is more viable as a real-world solution than other competing approaches in literature.

References

1. Aviv, A.J., Gibson, K., Mossop, E., Blaze, M., Smith, J.M.: Smudge attacks on smartphone touch screens. Proceedings of the 4th USENIX conference on Offensive technologies. pp. 1–7. USENIX Association (2010).
2. Muslukhov, I., Boshmaf, Y., Kuo, C., Lester, J., Beznosov, K.: Know your enemy: the risk of unauthorized access in smartphones by insiders. Proceedings of the 15th international conference on Human-computer interaction with mobile devices and services. pp. 271–280. ACM (2013).
3. Karlson, A.K., Brush, A.J., Schechter, S.: Can i borrow your phone?: understanding concerns when sharing mobile phones. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 1647–1650. ACM (2009).

4. Clarke, N.L., Furnell, S.M.: Advanced user authentication for mobile devices. Computers & Security. 26, 109–119 (2007).
5. Shi, E., Niu, Y., Jakobsson, M., Chow, R.: Implicit authentication through learning user behavior. Information Security. pp. 99–113. Springer (2011).
6. Riva, O., Qin, C., Strauss, K., Lymberopoulos, D.: Progressive authentication: deciding when to authenticate on mobile phones. Proceedings of the 21st USENIX Security Symposium (2012).
7. Frank, M., Biedert, R., Ma, E., Martinovic, I., Song, D.: Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication. Information Forensics and Security, IEEE Transactions on. 8, 136–148 (2013).
8. Bo, C., Zhang, L., Jung, T., Han, J., Li, X.-Y., Wang, Y.: Continuous user identification via touch and movement behavioral biometrics. Performance Computing and Communications Conference (IPCCC), 2014 IEEE International. pp. 1–8. IEEE (2014).
9. Yampolskiy, R.V., Govindaraju, V.: Behavioural biometrics: a survey and classification. International Journal of Biometrics. 1, 81–113 (2008).
10. Kwapisz, J.R., Weiss, G.M., Moore, S.A.: Cell phone-based biometric identification. Biometrics: Theory Applications and Systems (BTAS), 2010 Fourth IEEE International Conference on. pp. 1–7. IEEE (2010).
11. Killourhy, K.S., Maxion, R.A.: Comparing anomaly-detection algorithms for keystroke dynamics. Dependable Systems & Networks, 2009. DSN 09. IEEE/IFIP International Conference on. pp. 125–134. IEEE (2009).
12. Shen, C., Cai, Z., Maxion, R.A., Xiang, G., Guan, X.: Comparing classification algorithm for mouse dynamics based user identification. 2012 IEEE Fifth International Conference on Biometrics: Theory, Applications and Systems (BTAS). pp. 61–66 (2012).
13. Zhang, L., Tiwana, B., Qian, Z., Wang, Z., Dick, R.P., Mao, Z.M., Yang, L.: Accurate online power estimation and automatic battery behavior based power model generation for smartphones. Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis. pp. 105–114. ACM (2010).
14. Murmuria, R., Medsger, J., Stavrou, A., Voas, J.M.: Mobile Application and Device Power Usage Measurements. 2012 IEEE Sixth International Conference on Software Security and Reliability (SERE). pp. 147–156 (2012).
15. Shye, A., Scholbrock, B., Memik, G.: Into the wild: studying real user activity patterns to guide power optimizations for mobile architectures. Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture. pp. 168–178. ACM (2009).
16. Barbará, D., Domeniconi, C., Rogers, J.P.: Detecting outliers using transduction and statistical testing. Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 55–64. ACM (2006).
17. Keogh, E., Lin, J., Fu, A.: Hot sax: Efficiently finding the most unusual time series subsequence. Data mining, fifth IEEE international conference on. IEEE (2005).
18. Vovk, V., Gammerman, A., Saunders, C.: Machine-learning applications of algorithmic randomness. Proceedings of the Sixteenth International Conference on Machine Learning (ICML-1999). pp. 444–453 (1999).