RiskCog: Implicit and Continuous User Identification on Smartphones in the Wild

Anonymous Submission

Abstract

Mobile payment is becoming popular. Integrating the sensitive payment functionality to the smartphone introduces new security risks, for example, the attacker pays with the victim's account using the device stolen. By depicting the device owner with a diverse set of features, such as the locations the user frequently visits and face snapshot, the existing risk management for mobile payment is harder to get bypassed than the traditional user authentication mechanism with the simple account credential. However, it involves the heavy usage of user's personal information and raises the user's concern on privacy leakage. Moreover, the current risk management countermeasure deploys at the application level, where each mobile payment service vendor has its channel of data collection individually. If a user has several payment apps installed, the duplicate copy of the usage data will be uploaded.

In this paper, we propose the system, called RISKCOG, which solves the problem of identifying the authorized device owner by the data collected from the acceleration sensor, gyroscope sensor, and gravity sensor with a learning-based approach. We find the set of 56 features that is effective to train the classifier to fingerprint the device owner. Our feature set only leverages the motion sensors, which are commonly available on smartphones and have low privacy sensitivity in the context of social impact. RISKCOG is designed as a third-party service at the device level that requires no developer support. Previous studies require the user movement and fixed device placement, which fail to provide a fully continuous protection on the sensitive account. Our feature set is independent of a user's motion state. Moreover, our data collection and preprocessing mechanism ensure that the prediction accuracy will not be affected by the app-specific gesture or the noise generated in the dynamic device placement. We are the first to achieve risk management with motion sensors in the industry product after further resolving the problems of the imbalanced dataset with our stratified sampling method and missing of ground truth with a semi-supervised learning algorithm. For the data in the wild collected from 1,513 users, RISKCOG identifies the authorized owner in steady/moving states with the accuracy values 93.77% and 95.57%. Apart from payment risk management, our system can be applied to any existing mobile apps requiring user identification, such as social apps.

I. INTRODUCTION

Smartphones provide users with various functionalities, among which the popularity of mobile payment is growing exponentially. Based on the report by eMarketer [13], 37.5 million users are expected to use mobile payments in the year 2016. Although mobile payments have benefitted users immensely, they also introduce security threats, which can be classified into two categories. Data-driven risks are usually caused by leaking of sensitive credit card data bound with smartphones from sources including client side (mobile malware), network channel (man-in-the-middle attack), as well as server side. Human-driven risks, on the other hand, arise when parties other than the owner have access to the smartphone and utilize the payment functions for their own benefit. According to the report by LexisNexis [11], for the 15% of merchants accepting mCommerce payments in 2014, mobile transactions accounted for 14% of the total transaction volume, and 21% of the volume of fraudulent transactions. The mCommerce merchants' fraud pain-points indicate that an effective solution to mobile payment security is badly needed.

The data-driven risks are well-studied and many previous studies constructed defense mechanisms, such as malware detection [41], [27], SSL analysis [60], [34], and network penetration [53], [68]. However, the latter security risk still lacks effective countermeasures. The traditional user authentication mechanism deployed in the wild only verifies whether the user knows the account credential set up previously at the start of using the service. Moreover, the login credential based authentication requires the explicit user action, for example, account/password input. Risk management is thus proposed, where learning-based approaches are applied and construct the classifier to describe the usage pattern of the authorized phone owner, such as the locations he/she frequently visits and face snapshot. Risk management is mainly applied to the scenario of payment, while it fundamentally is a complementary solution to the general web/mobile services with the requirement of user identification. Comparing with the simple login credential, the risk management mechanism utilizes a diverse set of features to verify user's identity, and it is thus much harder to get bypassed. Furthermore, it continuously verifies the user identity during the daily usage without extra explicit action.

However, the existing risk management approaches have three unaddressed problems. (1) User privacy is heavily tracked, which raises the user's concern on privacy leakage [35] and leads to an unwillingness to utilize the service to shield a mobile app. (2) The detection mechanism is deployed at the application level, and each app such as a mobile payment service vendor has its own channel of data collection individually. They cannot share their data or result, making it impossible to reuse the detection results for other apps. Furthermore, there is often much redundancy in terms of the data collected from these apps for risk management. (3) The training and verification procedures require a user to move (e.g. walk) to extract specific features [51], such as step length, and they also require the fix device placement [31], for example, the smartphone is placed in specific human body locations with

TABLE I. COMPARISON WITH RELATED STUDIES

Study	Require user movement	Fix/dynamic device placement	Scale (#Users)	Require label
RISKCOG	No	Dynamic	>1,500	No
Lu et al. [51]	Yes	Dynamic	<50 (walking detector training), <50 (supervised training), <10 (unsupervised training)	No
Derawi et al. [31]	Yes	Fix	<100	Yes
Derawi et al. [32]	Yes	Fix	<100	Yes
Ho et al. [43]	Yes	Fix	<50	Yes
Kwapisz et al. [48]	Yes	Fix	<50	Yes
Ren et al. [56]	Yes	Fix	<50	Yes
Challenge	Lack of feature	Data availability & dynamic device placement	Imbalanced data set	Unlabeled data

the stationary device orientation. These solutions are thus not able to provide a fully continuous protection when the user is steady or the device has a placement that is different from those included in the model at the training stage.

In this paper, we solve the problem of implicitly identifying the authorized phone owner based on historical data collected from mobile devices by using a learning-based approach. Unlike the explicit user authentication, user does not need to take any specific action (e.g. account registration) to assist the system for the purpose of identification. Our solution has the following three requirements: (1) least user privacy requirement; (2) deployment as a third-party service at the device level; (3) capability of enforcing verification with various motion states and dynamic device placement. In our threat model, each smartphone has a unique owner, and various other attackers attempt to operate the phone. We only assume the acceleration sensor, gyroscope sensor, and gravity sensor are available on the device and work normally. By comparing with the related studies listed in Table I, we summarize the following challenges:

- **Lack of feature.** Previous studies show the feasibility of identifying a user with her/his gait by collecting data from the motion sensors of smartphones [51], [31], [32], [43], [48], [56]. The phases of training and identity verification require the user to be in the moving state, such as walking, running, which provides the features, such as step cycle, speed. To offer a continuous verification, we should not have any requirement of user movement. Our system intends to recognize the authorized owner by the manner of handling the device. Thus, those features that are only available in the moving state are not usable.
  Preserving the portability of our solution introduces the difficulty regarding feature availability. Although Android supports numerous sensors, which can be potentially utilized to fingerprint users' pattern, the fragmentation issue [7] largely hinders it from being deployed universally. Specifically, many sophisticated sensors are not available on some brands/types of device, e.g. the temperature sensor. Moreover, a portion of sensors has to be integrated into an application to work, for example, the pressure sensor needs to be bound to an application view element for collecting the area and force of user's touch event. That requires extra developer support at the application level. Due to the least user privacy requirement, any feature involving user's personal information in the context of social impact is not available.
- **Data availability & dynamic device placement.** A sufficient training data set is necessary when establishing a classifier to recognize the phone owner precisely. Only the data collected from the motion sensors during daily usage contribute to the classifier training because fingerprinting user fundamentally depends on the user's specific pattern in handling the phone. It is thus inevitable that some users produce less training samples. For example, she/he might use the phone rarely, or prefer to keep the phone on a stationary plane during usage, in which no motion event can be leveraged to construct the classifier to represent the authorized owner. Moreover, even for the same device owner, his/her pattern of handling the phone dramatically varies with different types of applications, e.g., the frequent typing gestures in a chatting application compared with the phone rotation in a race car game. The application-specific pattern will challenge the classification accuracy.
  Previous studies [31], [32], [43], [48], [56] also have the strong assumption of fix device placement, for example, the smartphone needs to be put in the pocket of trousers and the orientation remains unchanged, which can hardly be achieved in a practical situation.
- **Imbalanced data set.** When identifying the authorized phone owner, we labeled the data of the authorized user as 1 and that of other users as 0. The resulting binary classification task is imbalanced as the number of positive examples is much less than that of negative examples. The traditional fraud detection problem also deals with the imbalanced data set, where a very big part of the data set, usually over 90%, describes normal activities and only a small fraction of the data records gets classified as fraud. The imbalance ratio in our project increases when the system is applied to more and more users, where the ratio normally exceeds 1:1000. To the best of our knowledge, there is no related literature discussing or resolving the issue of the imbalanced data set, given the much smaller number of experiment subjects compared to us.
- **Unlabeled data.** The proposed prototype systems [31], [32], [43], [48], [56] introduce supervised learning algorithms for the well-labeled training set, for example, whether each data sample is generated when the authorized user is using the phone. However, there is no ground truth whether the collected data is generated by the authorized owner or not in the practical scenario. If we simply apply the unsupervised learning mechanism to the classification problem without a definitive number of clusters, high time latency will be introduced, which hinders our solution from being deployed as a real-time service.

We design a system, called RISKCOG, to provide a fully continuous and implicit user identity verification service. It is based on semi-supervised learning to identify phone owner simply based on the acceleration sensor, gyroscope sensor,

and gravity sensor with no requirement of user's motion state and device placement. The data collection mechanism only reads the motion sensors when the phone is recognized as being used actively, which makes sure all the useful data in identifying user are completely captured while the remaining data are filtered out. The data from motion sensors are then preprocessed, which are calibrated by removing those collected when the phone is put on a flat surface. Each data sample is further classified by whether the user is in a steady or moving state. After that, two parallel classifiers are trained for these two states based on a set of 56 features computed from the raw motion sensor data. Considering the lack of ground truth, we assume that the data is generated by the owner, and split them into chunks, and then serve them to the training algorithm in the online mode. By observing the alignment amongst the classifiers trained with data in the continuous chunks, the learning algorithm adaptively chooses to accept or reject the predetermined ground truth.

We made the following contributions to overcome the challenges mentioned above.

- We find 56 features from the motion sensors of smartphones, which are effective to recognize the owner of a device accurately. Moreover, The feature set does not involve any in-app invocation or user privacy tracking. And it is independent of user's motion state. Thus, our system does not need developer support and has high portability and it is able to enforce identity verification just by a user's manner of handling the device even in the steady state.
- We design a data collection mechanism to resolve the data availability issue, which cognitively recognizes phone usage events and completely captures all the data helpful in fingerprinting usage pattern. The useless data that are generated when the device is put on a plane or have the dependency on the application-specific pattern are filtered out. We implement a preprocessing procedure that includes data calibration and user motion state recognition. That guarantees the usability of our system with dynamic device placement.
- To solve the issue of imbalanced data set with a huge number of negative samples from other users, we apply stratified sampling to construct the negative sample part in our training set. The sampling method proposed maintains the temporal continuity of sensor reading, which is very helpful to model the user's pattern. It helps to improve the representativeness of the negative sample by reducing sampling error. But instead of sampling a single data point, we adopt sample pieces which cover samples within a certain time period, thus preserving time consistency. When the data set is on a moderate scale, we also tried different instance selection methods. The general principle is to filter out noisy instances and keep boundary instances that might have a big contribution to the classification. However, the instance selection methods are too inefficient to be useful in the real situation, when more than thousands of users are involved.
- We design a semi-supervised online learning algorithm, where the classifier is trained incrementally through data collected in chunks. It allows us to eliminate the high time latency when handling unlabeled data with unsupervised methods. By checking the consistency among the data sent to server incrementally, we are able to filter out the part that is not coherent with the authorized owner's fingerprint.

To the best of our knowledge, we are the first to achieve high accuracy for implicit user identification with wild data collected from an industry mobile payment app by one of the biggest mobile service vendors in the world, which has over 300 million users. In our evaluation on 1,513 users, RISKCOG achieves the classification accuracy values of 93.77% and 95.57% for the steady state and moving state, respectively. The performance of our device-level implicit user identification solution implies the possible application scenarios other than mobile payment risk management. Details will be discussed in Section III. We also produce an Android app including our user identification mechanism for general usage apart from payment. Due to the requirement of anonymous submission, we will open-source the app after getting published.

The remainder of this paper is organized as follows. Section II presents a brief background of our work. In Section III, we cover RISKCOG design in detail , which is then followed by an explanation of the implementation procedures in Section IV. Section V deals with the overall evaluation of our system. Section VI and VII include discussion about relevant work and Section VIII includes our concluding remarks.

II. BACKGROUND

A. Authentication, risk management, mobile payment

Authentication is used to prevent the unauthorized parties from using the sensitive services. Currently, the credential is the predominant form of an authentication system, given its ease of deployment. However, it is known to have many security problems. First, it is only able to verify whether the user knows the credential rather than recognize whether she/he is the owner of the device. The credential-based authentication is thus vulnerable to dictionary attacks. For example, a recent report about data breaches [22] shows that 4.1% of users choose "123456" as their passwords, and 79.9% of apps still accept weak (lower-case only letters) passwords. Florencio et al. [36] even found that a single password is typically used to access over five sites. Furthermore, the credential-based authentication cannot enforce the continuous protection and achieve usability at the same time. A fully continuous verification requires a user's explicit input action, every time the user accesses the sensitive services. While if the user uses the functionalities, such as automatic login, out of the concern of usability, the identity will only be verified once.

Considering the security issues of the traditional login credential based authentication, risk management is thus proposed. It introduces learning approaches to training a model, which is able to exactly describe the authorized device owner by a diverse set of features, such as face snapshot and the locations that the user frequently visits. At the time of verification, the system will collect the test samples and predict the probability that the user who attempts to access the service is the owner of the device by checking the alignment between the test samples with the trained model. It is much harder for the attackers to bypass the verification compared with the traditional authentication mechanisms, given the difficulty of mimicking a legitimate user's patterns. Moreover, the processes of model training and verification require no explicit actions

TABLE II. AVAILABILITY OF ACCELERATION SENSOR (AC), GYROSCOPE SENSOR (GY), AND GRAVITY SENSOR (GR); MS FOR MARKETSHARE IN 2016

Brand	MS	Device	AC	GY	GR
Samsung	22.2%	Galaxy S7	D	D	D
		Galaxy S6	D	D	D
		Galaxy Note 2(3, 4)	D	D	D
		Galaxy CORE Prime	D	×	×
Apple	16.8%	iPhone 6s Plus	D	D	D
		iPhone 6(s)	D	D	D
		iPhone 5(s)	D	D	D
Huawei	9.3%	P9	D	D	D
Xiaomi	5.8%	MI 3(4, 5)	D	D	D
LG	5.0%	G5	D	D	D
OPPO	3.9%	R9	D	D	D

from the user, and it is thus to enforce the continuous protection without sacrificing the usability.

Mobile payment has been gaining popularity due to the convenience during the past years. The credit card bound with the user's bank account is added to the client apps provided by the mobile payment service vendors. To make a transaction, a user may either use the point-of-sale (POS) systems with the near-field communication (NFC) technology [14] or invoke the payment-related interfaces deployed on remote servers. User authentication is needed in all the scenarios involving sensitive functionalities, among which mobile payment is a critical one. For example, if a device without a strong authentication being enforced is stolen, an attacker may compromise the victim's account and cause financial loss.

B. Privacy

The privacy preserving property of RISKCOG is defined in the context of social impact. Previous studies found the feasibility of fingerprinting mobile devices with motion sensors [33], [30]. However, device tracking does not imply the identity of its owner in social life. We only know the mapping between a trained model and an authorized device owner. However, it is unable to further figure out who the device owner is. Compare with the motion sensor data, other types of features used in the existing risk management systems are more sensitive. It is straightforward to know who the user is when face recognition is utilized for the purpose of authentication [65], [1]. As for geo-location, it is able to identity the owner in the physical world as stated in previous studies [25], [40], [24], [44], [47].

C. Platform porting & sensor availability

Based on the learning-based approach, our user identification only requires the acceleration sensor, gyroscope sensor, and gravity sensor. Thus, RISKCOG can be easily migrated to various mobile platforms (e.g., Android and iOS with leading marketshare). Moreover, we survey the required sensors' availability on the types of devices with high market penetration by several major smartphone vendors. As shown in Table II, the three motion sensors are available on the popular devices surveyed, except Samsung Galaxy CORE Prime that was related in 2014.

D. Android

Android apps are distributed in the form of an Android application package (APK). The installation package is a JAR archive that includes compiled Java source files in Dalvik bytecode [8], a compiled manifest file stating the files packed in the distribution and required permissions, and other resources, such as layouts, images, and the native libraries in `.so` format. The application bytecode is classified into four basic components regarding purpose and lifecycle: *Activities, Services, Content Providers, and Broadcast Receivers*. Each component is a different point through which the system can enter the application.

The Android platform provides numerous type of sensors [4], which can be classified into three broad categories: motion sensors (e.g., gravity sensor), environment sensors (e.g., humidity sensor), and position sensors (e.g. magnetometer). A recent report about Android fragmentation issue [3] shows 599 distinct Android brands with 11,868 distinct devices in 2013 and 18,796 distinct devices in 2014. The availability of other advanced sensors varies largely with different types of devices, and we thus select our features based on the common motion sensors. The sensor reading operations in Android follow the callback manner. The developers register callback functions to the interface `SensorEventListener`, and receive the sensor events when sensor values change or when sensor accuracy changes.

III. SYSTEM DESIGN

The architecture of our system is illustrated in Figure 1. Each device is uniquely identified by the IMEI number. The client app deployed on the device periodically collects data from the acceleration sensor, gyroscope sensor, and gravity sensor. The data is incrementally uploaded to the remote server. After preprocessing, the training set is constructed. The training set includes both the data from the authorized phone owner labeled as 1 and the data from other users labeled as 0. Our semi-supervised online learning algorithm is based on the assumption that most of the data uploaded from one device originate from the authorized owner. We thus have a well labeled training set for the negative instances collected from other devices. But it is infeasible to obtain reliable labels for positive instances from the authorized user because the device may be used by the owner's friends or family members during the data collection phase. However, we expect the mislabeling rate to be low. The online learning mechanism cognitively detects and filters the data not aligning with the user's fingerprint by checking its statistical consistency against historical data. The specific usage manner of each phone owner will be fingerprinted as the corresponding classifier. When the classification accuracy of the classifier is higher than a predefined threshold and is stable across consecutive chunks of data, we recognize it as being fully trained. After that, when the various mobile payment vendors receive the transaction requests from the smartphone, they just need to query whether the phone is being used by the authorized owner during the recent activities from RISKCOG's server. Compared with the deployment model of existing mobile risk management solution, the unified solution of RISKCOG eliminates the cost where each mobile payment vendor maintains its user fraud detection mechanism in the backend. Moreover, the data used

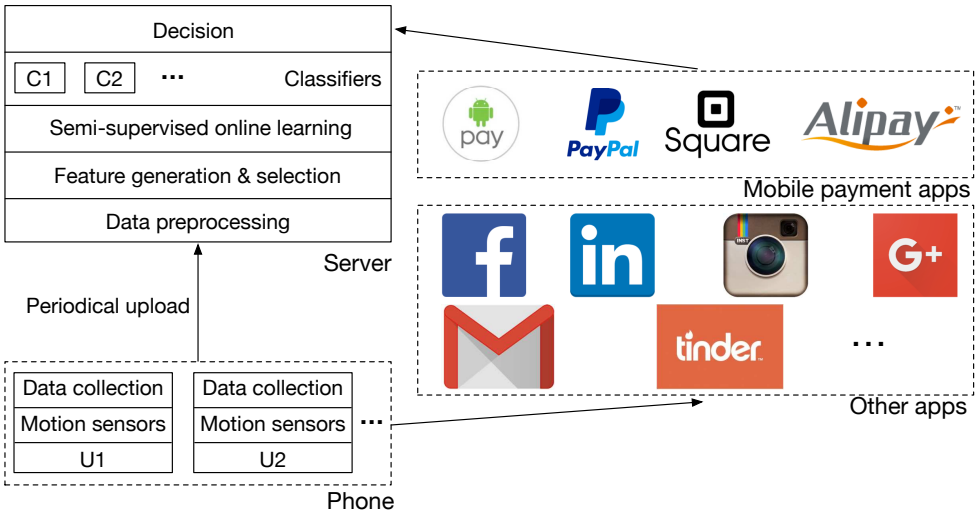
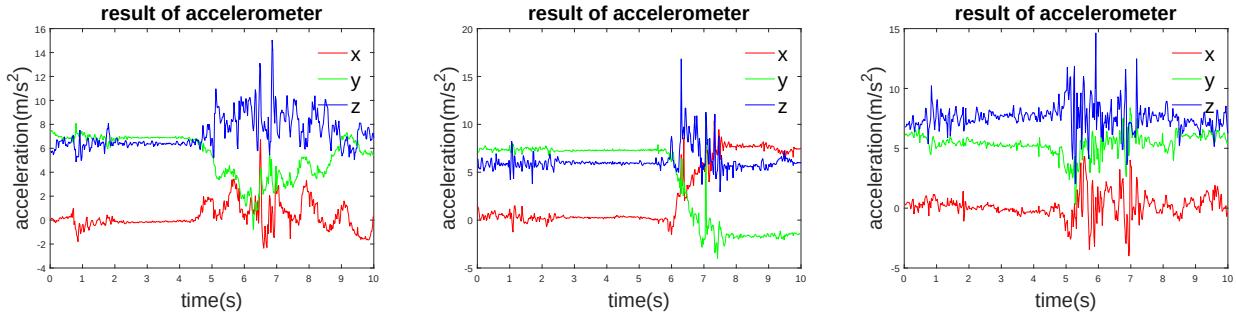


Fig. 1. System architecture



(a) Chat app (b) Game app (c) News app

Fig. 2. Acceleration sensor values in the temporal dimension

for user identification need to be uploaded only once rather than be sent to each vendor's server separately.

Other application scenarios. RISKCOG is proposed mainly to address the drawbacks of existing mobile payment risk management mechanisms. It fundamentally investigates the difference among user's manner of handling device. The specified classifier implicitly verifies whether the user performing any sensitive operation on the smartphone is the authorized device owner. Our privacy preserving solution only requires the insensitive data from motion sensors and is deployed as third-party service without developer support. RISKCOG is thus complementary to any mobile application that requires user authentication. For example, Alice and Bob are close friends. Alice leaves her phone at Bob's home. Bob can check Alice's Facebook private activity without her consent, if the Facebook application's automatic login option is enabled. However, if RISKCOG is applied, it will detect the unauthorized user. The mobile application provider or the end-user can customize the follow-up behavior of the application, such as notifying the phone owner or rendering an empty page.

This section describes 4 sub-modules of the overall system: (1) Sensor data collection and preprocessing; (2) Feature generation and selection; (3) Online learning algorithm; (4) Decision.

A. Data collection and preprocessing

RISKCOG collects data from acceleration sensor, gyroscope sensor, and gravity sensor. Each sensor reading includes values corresponding to the x, y, and z axes as follows.

$$\{X_a(k), Y_a(k), Z_a(k)\},$$
$$\{X_{gy}(k), Y_{gy}(k), Z_{gy}(k)\},$$
$$\{X_{gr}(k), Y_{gr}(k), Z_{gr}(k)\}.$$

Here, the parameter k represents the $k-th$ acceleration, gyroscope and gravity reading in the time dimension, respectively.

We develop a mobile application for the purpose of data collection. It needs to precisely detect the duration when the device is being actively used because only the values of motions sensors collected during this duration are effective to represent user's manner. Through our experiment shown in Figure 2, we observe that the sensor readings largely vary with the different types of applications even for the same user, which will affect the classification accuracy of the trained model. For example, a user keeps rotating his/her phone when playing a race car game driven by the acceleration sensor; a lot of typing gestures are generated when using a chatting application; the device is stable when a news application is used. However, the sensor readings are relatively consistent during the start of an application, in other words, the loading

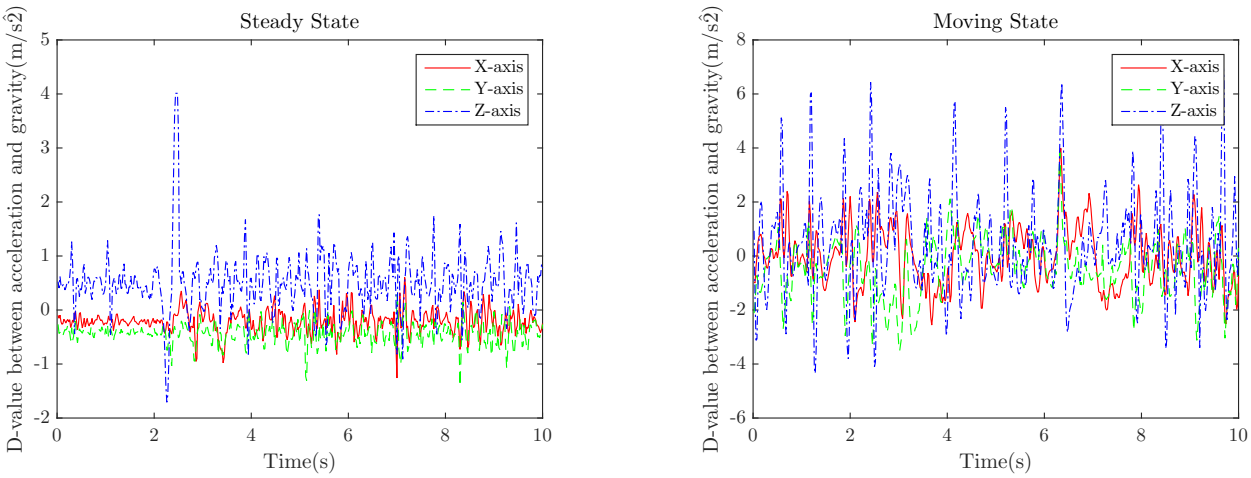


Fig. 3. Difference between values from acceleration sensor and gravity sensor

phase.

We have a `BroadcastReceiver` to capture the system event where the screen of a device is turned on, and then it starts a `Service` [5] that periodically queries the current application in the foreground. When the currently active application is different from the one in the last query, we will recognize that a new application is started. The data collection will keep running for 3 seconds if both of the two conditions are satisfied:

- The screen of the phone is on.
- A new application is running in the foreground.

The data collected are thus generated during the active daily usage and the application-specific pattern is also filtered.

We preprocess the raw data mainly based on the gravity sensor, which includes the data calibration and motion state recognition. A user may not actually hold the phone during daily usage, and we thus observe that a portion of data is ineffective to reflect the difference among various users' patterns, even if we apply the two conditions above in the data collection stage. Our data calibration phase has the following condition regarding the gravity sensor values in three dimensions, and it allows RISKCOG to remove the data in those situations, such as keeping the device flat on a desk. The boundaries of the gravity sensor readings on three dimensions are determined by experiment.

$$\{-1.5 < X_{gr}(k) < 1.5\}$$
$$\cap\{-1.5 < Y_{gr}(k) < 1.5\}$$
$$\cap\{9 < |Z_{gr}(k)| < 10\}$$

After removing the data samples that are ineffective to represent the pattern of device owner, we project those sensor readings to our global coordinate system, which allows RISKCOG to be insensitive to device orientation. We first identify the gravity direction based on the values read from the gravity sensor on three dimensions, whose opposite direction will be set as the z-axis in the global coordinate system. It is thus straightforward to decide the remaining x-axis and y-axis by Fleming's right-hand rule. We should note that all the features used in the training and test phases are based on the sensor readings after projection.

Moreover, the usage pattern extracted differs to a large extent when the user is moving as opposed to when the user is stationary, which is shown in Figure 3. In the moving state, the difference between the values of acceleration sensor and those of gravity sensor (D-value) has a higher amplitude compared to the D-value in the steady condition. In addition, we observe the D-value in the moving state has the periodic property.

We also find that those two motion states can be further divided into sitting and lying. Although the data samples in the sitting state and the lying state have different dimensions align with the gravity direction, they are very similar to each other after being projected to the global coordinate system. We perform an experiment and find the result of recognizing sitting and lying separately, or mixed them together have the similar accuracy on user identification. Moreover, it has extra time latency to classify the additional state. We thus only consider the steady state and the moving state in RISKCOG. In our preprocess, we classify the user's data samples into two motion states based on a predefined threshold of the amplitude and the appearance of the periodic feature by conducting the discrete Fourier transform (DFT). One classifier is trained for each state of one unique user. When the data samples are verified regarding whether the phone is using by the authorized owner, they are first classified on the motion state and checked against the corresponding classifier. If we use one classifier for all the motion states, there will be huge inconsistency within the data samples of one user that will affect the classification accuracy and divergence of the classifier in online learning.

B. Feature generation and selection

For the classification, we only utilize data collected from acceleration and gyroscope sensors. Therefore, feature generation and selection are carried out with readings of these two sensors. Since standard classification methods cannot be directly applied to time-series data, we first extract the feature vectors from the raw time series data. To fulfill this, we divide the raw time series data into 0.2-second segments and extract features based on the 10 sensor readings within each segment.

Denote the ith value of the feature vector as

$$\mathcal{F}_i = \{F_{1i}, F_{2i}, \cdots, F_{pi}\},$$

which includes p features. In order to maintain the consistency among feature vectors in the temporal domain, we utilize sliding window design with 50% overlap between each pair of neighbor segments, i.e.

$$\begin{aligned} \{X_a(k), Y_a(k), Z_a(k), X_{gy}(k), Y_{gy}(k), Z_{gy}(k)\}_{k=1}^{10} &\Rightarrow \mathcal{F}_1 \\ \{X_a(k), Y_a(k), Z_a(k), X_{gy}(k), Y_{gy}(k), Z_{gy}(k)\}_{k=5}^{15} &\Rightarrow \mathcal{F}_2 \\ &\cdots \end{aligned}$$

According to our experiment, if the length of sliding window is too long, it will result in a loss of accuracy. Meanwhile, if the length is too short, It's time-consuming to process the raw data.

Our system is developed based on the assumption that the distribution of sensor data is unique for each user. And it can be utilized as the digital fingerprints to identify the authorized user of mobile devices. We generate a total of 80 possible features, which covers multiple moments and other commonly used statistical properties of the distribution. To investigate the effectiveness of each single feature to depict the device owner's pattern, we separately utilize each potential feature to establish a classification model based on SVM and evaluate the accuracy of these 80 models given the ground truth in the laboratory settings. And then, we rank features based on their independent classification capabilities and select the top ranked ones to form the final feature set. The detailed formulas for the selected features are listed in ascending order of the prediction accuracy. Our feature selection reduces the feature space and further improves the RISKCOG's latency performance on training and test.

1 Mean: $\sum_{k=1}^{K} x(k)/K$

2 Standard Deviation: $\sqrt{\sum_{k=1}^{K}[x(k) - \bar{x}]^2/(K-1)}$

3 Average Deviation: $\sum_{k=1}^{K} |x(k) - \bar{x}|/K$

4 Skewness: $\sum_{k=1}^{K}[(x(k) - \bar{x})/\sigma]^3/K$

5 Kurtosis: $\sum_{k=1}^{K}[(x(k) - \bar{x})/\sigma]^4/K - 3$

6 Lowest value: $\min\{x(k), \quad k = 1, \cdots, K\}$

7 Highest Value: $\max\{x(k), \quad k = 1, \cdots, K\}$

8 Cross zero rate: $\sum_{k=0}^{K-1} ||sgn[x(k+1)] - sgn[x(k)]||/K$

9 RMS Amplitude: $\sqrt{\sum_{k=1}^{K}[x(k)]^2/K}$

10 Average root sum square: $\sum_{k=1}^{K} \sqrt{x^2(k) + y^2(k) + z^2(k)}/K$

Here, K equals 10 for our case. Replace x in the first 9 formulas with $X_a(k), Y_a(k), Z_a(k), X_{gy}(k), Y_{gy}(k), Z_{gy}(k)$ respectively and they will render us 54 features. The last formula provide us with 2 features from 2 sensors. In total, 56 features are extracted and used for the classification purpose. The remaining 24 features have high correlation with the selected 56 features, which cannot contribute to the classification accuracy and increase the latency of training and test.

We also try principal component analysis [57] on the original feature set, and utilize the first 50 principal components for the classification. However, this feature space reduction method is much slower than the one we used. And the final performance of both methods is comparable.

C. Semi-supervised online learning algorithm

RISKCOG utilizes supervised learning to distinguish an authorized user from others based on each motion state, beginning with a training phase followed by testing.

During the training phase, raw data from a smart phone are collected for generating the feature vectors. Each user has n feature vectors, denoted by $\mathcal{F}_i, i = 1, \cdots, n$. For p phone users, $n \times p$ vectors can be used to train the classifier all together. From now on, the sample size refers to the number of feature vectors n. Treating the data set of the authorized user as Class 1 and that of all the other users as Class 0, we deal with a severely imbalanced data set given p is large.

We employ the stratified sampling to handle this problem [63], which groups the vectors by one feature value. According to the feature selection ranking result, the average root sum square of acceleration (ARSSA) readings is the most important feature for classification. Therefore, we carry out stratified sampling based on these feature vectors of all the other users. We also observe that the temporal continuity of sensor reading is actually helpful to depict the authorized owner's pattern of handling the device. Our sampling method thus needs to keep this property. To be specific, we select the 1st, 100th, 200th,... ARSSA values for each user, sort all $n \times (p-1)/100$ values and divide them into 5 equal size strata. Then, an equal amount of samples is randomly drawn from each strata. To preserve the time consistency, each chosen sample along with 99 samples after it are all selected to form the negative sample set. By doing so, negative samples including in the training set has better representativeness of the $p-1$ users. And the final model becomes more robust than simple random sampling. Regarding the number of the stratum, a larger number brings us a fine-grained sampling, in other words, a stronger capability of representing other users. However, it introduces higher latency.

Our training set is constructed as the Figure 4. The ratio of the number of samples by the phone owner to that of other users is 1:5, which can be properly handled by normal learning algorithms, such as SVM. The optimal point is chosen by experiment. For effective training, the sample size of positive instances usually needs to exceed 4000. The specifically required sample size may vary among different targeting users. It greatly depends on the users' habit of using the mobile phone. If they constantly allow the other users, such as friends

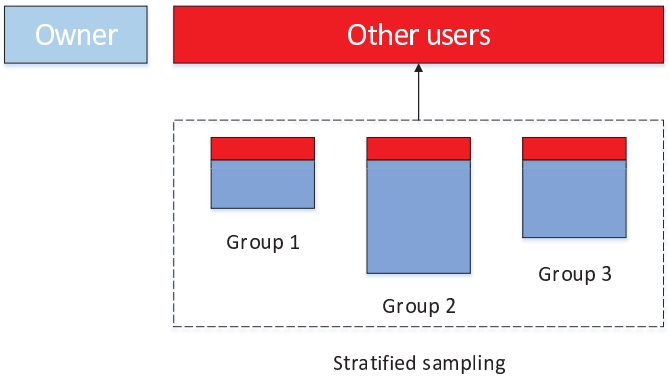


Fig. 4. Training set construction

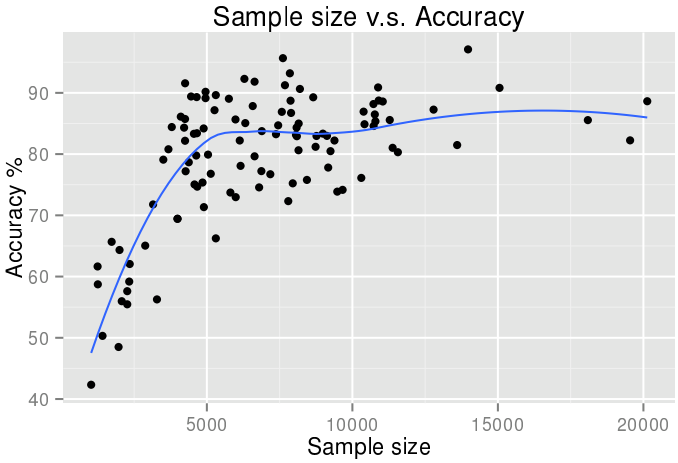


Fig. 5. Sample size requirement

and family members to access their phones, the mislabeling problem will be more severe in the training set. Due to the label noise, more samples are in need for our algorithm to wash out the noisy feature vectors and form the fingerprint for the authorized owner himself.

We carried out the experiment on 106 people based on data collected in 5 days, and the sample size of positive instances ranges from 1024 to 20132 depending on the frequency of the usage. Classification model has been developed for each person, and tested on an equal sized testing set constructed by data collected from the next 2 days. Accuracy for each model has been evaluated and presented in Figure 5, which shows the strong relation to the sample size of positive instances. When the sample from authorized user is insufficient, less than 4000, the performance of our classification is less satisfied with low accuracy. However, it gets improved drastically when sample size increases. We also notice that once the sample size exceeds the threshold 4000, accuracy will not get improved by simply adding more samples.

We choose Support Vector Machines (SVMs, hereafter) with Gaussian radial basis kernel function as our classification method for the following reasons:

- **Nonlinear classification boundary.** After analyzing our data, we expect our features to be nonlinear and the problem is not linearly separable, and thus, we skip the most commonly utilized Logistic regression (LR) method in the field of user fraud detection. SVMs use a different loss function (Hinge) from LR, where they try to maximize the margin between two classes. The SVM with a nonlinear kernel (e.g. RBK) will help us build a nonlinear classification boundary.
- **Comparatively high dimensional space.** Another related reason for choosing SVMs is that we have a comparatively high dimensional feature space with 56 features subtracted from original observations. And SVMs have been reported in a lot of references to work better with this kind of situation [67], [46], for example, text classification. We also tried two-dimensional reduction methods [39] before applying SVM, principal component analysis and variable selection based on prediction capability of each feature. And there is no significant improvement in the results, which suggested SVM can handle 56 features pretty well based on our enormous sample size.
- **Dependent data structure.** The random forest [62] or other ensemble methods seem to be the most popular choice nowadays. These types of methods operate by constructing a multitude of decision trees at training time and outputting the class that is the mean prediction (regression) of the individual trees. However, they are not suitable for our data set because random sampling involved. The original data we collected is high-frequency time series. Although we preprocess the data to alleviate the serial correlation of observations by averaging over every 0.2 seconds, data dependency still exists to some level, which affects how we split our data set. The random forest requires us to randomly split data into out-of-bag (OOB) and in-bag samples for each decision tree. This random splitting will lead to bias in the OOB error estimate of the forest if the autocorrelation of the series is strong.

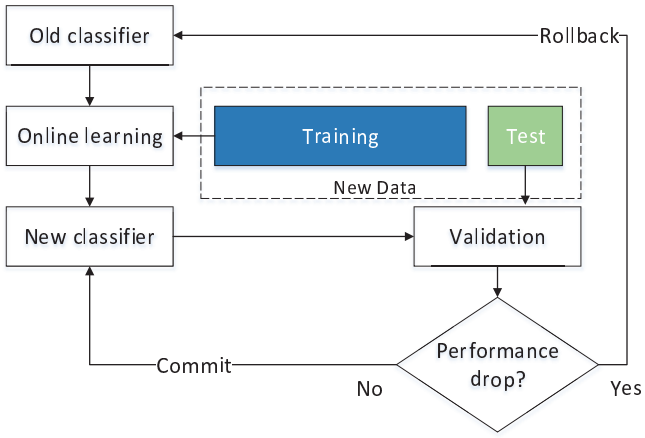


Fig. 6. Semi-supervised online learning

The flowchart of our semi-supervised online learning algorithm is illustrated in Figure 6. The collected data samples are uploaded to our server in chunks. One chunk is split into two parts for both training and testing purposes. The new training data and other users data are used to construct a training set.

The online learning module takes the cached old classifier and training set as inputs and produces a new classifier. The new classifier does a validation on the test samples from the data chunk and other users data. The old classification accuracy and the new one are represented as α_{old} and α_{new}, respectively. The condition to commit the new classifier is expressed as follows. The parameter λ ranging from 0 to 1 is the factor to quantify the weight of new classification accuracy. The value β is the threshold to represent the normal performance variation rather than that caused by data inconsistency.

$$\lambda\alpha_{new} + (1-\lambda)\alpha_{old} > \alpha_{old} - \beta.$$

In the daily usage, the authorized device owner may share her/his phone to others, such as friends and family members. In the practical deployment, we have no idea of the label ground truth. Incorporating the noisy data in the classification model would affect the prediction accuracy. However, we observe the misalignment between other parties' manner of handling the device and that of the owner. Our design of commit/rollback allows RISKCOG to detect and filter the noisy data.

Another problem is to identify whether the classifier is fully trained, in other words, whether it can be used for the purpose of user identification. We propose the following conditions.

$$\alpha_{new} > A \text{ and } Var(\alpha) < V.$$

In our commit/rollback design, we have the classification accuracy for each chunk of data. When the latest prediction accuracy is higher than the threshold A and the variance of all the accuracy values of those chunks which are accepted previously is smaller than V, we will identify the classifier training is finished. This implies that the performance converges to a stable state.

D. Decision

Given a feature vector corresponding with one slide window in duration $w = 0.2$ second, the trained classifier outputs the probability whether the owner is using the phone p, which is used to get the binary decision d as

$$d = \begin{cases} 1, & \text{if } p > \gamma \\ 0, & \text{else.} \end{cases}$$

Here, γ is the decision threshold from 0 to 1.

Through proposing RISKCOG, we also generalize the issues as follows which need to be considered when solving real-world problems with learning-based approaches.

- **Feature selection.** Apart from being effective to fingerprint the objects, the availability is a key factor for the broadness of the solution. Those features that are only available in certain scenarios are excluded in our system, for example, the step cycle that can only be fetched when the user is moving, and data from the pressure sensor that is not provided on some brands/types of device.
- **Data collection.** It is inevitable that the data are collected in a noise surrounding environment. Asking the explicit input from a user, such as a predefined gesture, can alleviate this problem. However, it affects the user experience. Our design passively recognizes a common and stable data collection environment, where a new application is started. The samples which contain noise and affect the prediction accuracy are either filtered or preprocessed before the training phase. For example, the samples which are collected when the device is put on a stationary plane are removed, and the remaining samples are projected to a global coordinate system.
- **Training set sampling.** When the scale of the solution gets large, the data set will become highly imbalanced. A sampling method is needed to preserve the representability of the training set and allow the classic machine learning algorithms, such as SVM, to be usable.
- **Ground truth.** Unlike the controlled laboratory environment, the reliable label or ground truth cannot be expected from the data collected in the wild. Unsupervised learning algorithms, such as the expectation-maximization (EM) algorithm, can be leveraged. However, solving classification problems without a definitive number of classifiers. We assume the existence of label and improve its reliability by investigating the statistical consistency against historical data.

IV. IMPLEMENTATION

Our data collection scheme involves verifying the active device screen and the presence of applications in the foreground. We implement the `BroadcastReceiver` [2] to capture the system event where the device screen is turned on. Android provides the two APIs `getRunningAppProcesses()` and `getRunningTasks()` to retrieve the current application running in the foreground. However, starting from Android 5.0, those APIs are deprecated and cannot return the information of other applications. The list of running apps can be alternatively fetched by using `UsageStatsManager` [6]. However, this requires users to grant application the permission in system settings. To preserve the portability of RISKCOG on the fragmented Android devices, we invoke the system command line tool `ps` and implement a parser to map the running `pid` to the application name. Our implementation allows us to intercept the active applications properly on all the existing versions of Android without requiring any permission.

We first implement the data preprocess module in C++. It is intended to filter the data which are ineffective to fingerprint user's pattern, e.g. the phone put on a stationary plane, and distinguish the two motion states. For the feature generation, we utilize the public tool `LibXtract` [15] to extract the 80 possible features.

We use `Weka` [21] and `LibSvm` [12] to train our model for each person. Firstly, we utilize sliding window design with 50% overlap between each pair of neighbor segments. Then, we convert all data to the format of `arff` which is acceptable by `Weka`. In SVM, our current implementation is based on the radial basis function. Regarding the online learning algorithm, we leverage the open-source library `Vowpal Wabbit` [20] in C++. The framework is able to update the classifier adaptively based on the new chunks of data, which supports multiple learning algorithms and is the most stable libraries with similar functionalities to our knowledge.

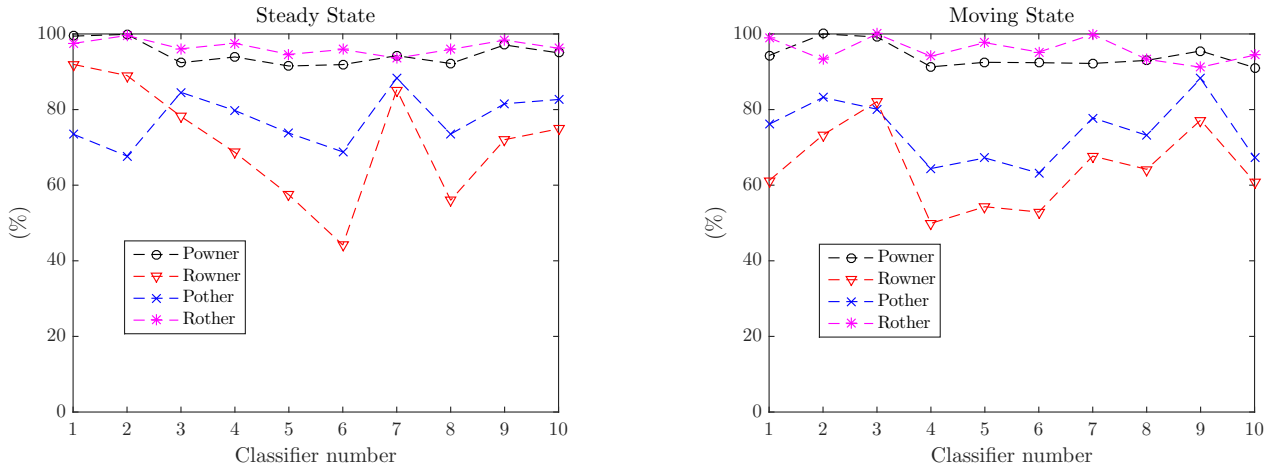


Fig. 7. Accuracy performance on experimental data with ground truth for 10 participants

Overall, We implement RISKCOG with over 5,000 lines of code in C++ and 2,000 lines of code in Java.

V. EVALUATION

In this section, we will first formulate the evaluation metrics and introduce the organization of our data set, which includes the experimental data collected from participants in a laboratory environment and raw data provided by the industry mobile payment vendor. Next, we will present the classification accuracy performance for both the offline learning and online learning, and the overhead of RISKCOG. Finally, we assess the robustness of our system with brute-force attacks.

A. Data set and metrics

Our data set includes two parts. For the experimental data with ground truth, we have 10 participants who are not the authors of this paper and use the same phone for one day. Each participant generates 9240 samples by handling the phone in both steady and moving states. For each user, we split the data samples into the training set and the test set. The ratio of the number of samples in the training set to that in the test set is 4:1. In the training set, the ratio of the number of samples from the authorized owner to that of other users is 1:5. The test set follows the same distribution.

For the raw data without ground truth, they are directly collected from the risk management product by an internet company with millions of users. In our collection scheme, the collection frequency is 50Hz. We collect data from 1,513 different users for 10 days. Each data collection phase lasts 3 seconds. For the sake of traffic usage and battery consumption in the production environment, there are at most 20 data collection phases (60 seconds) in one hour. The IMEI is the user identifier. We note that a portion of the data collected will be filtered, which are ineffective to fingerprint the user (e.g., phone is put on a stationary plane, as discussed in Section III-A). We assume the data collected from a smartphone is generated by the authorized owner. The distributions of training and test sets are identical to the experimental data mentioned above.

Regarding the ethical considerations, we let the participants know what they will do in the experiment and how the experimental data with ground truth will be used. For the data in the wild, we include these issues in the user agreement. All the data in the evaluation are used with the confirmation from human subjects.

The following metrics are used throughout our evaluation.

- True positive (TP). The authorized owner is correctly identified.
- False positive (FP). Other users are incorrectly identified as the authorized owner.
- False negative (FN). The authorized owner is incorrectly identified as other users.
- True negative (TN). Other users are correctly identified.
- Performance & Overhead. We first evaluate the time latency of training and test on the server side. Then we check the battery consumption, network traffic, CPU, and memory of the phone when our client app is installed.

The classification performance of RISKCOG is depicted with the following values: precision for phone owner P_{owner}, recall for phone owner R_{owner}, precision for others P_{other}, recall for others R_{other}, and classification accuracy.

$$\begin{aligned} P_{owner} &= TP/(TP+FP) \\ R_{owner} &= TPR = TP/(TP+FN) \\ P_{other} &= TN/(TN+FN) \\ R_{other} &= TN/(TN+FP) \\ FPR &= FP/(FP+TN) \\ Accuracy &= \frac{TP+TN}{TP+FP+FN+TN} \end{aligned}$$

The ROC curve reflects the overall performance of RISKCOG, which shows the true positive rate against false positive rate with various classification threshold γ. Specifically, the binary classification outputs the probability p of whether the test sample is generated by the authorized phone owner. The sample will be classified as true if p is larger than γ.

		Steady	Moving
#Training samples	AVG	20648	9280
	SD	15660	10212
Training time (s)	AVG	148.36	21.21
	SD	177.23	24.06
#Test samples	AVG	5162	2320
	SD	3915	2553
Test time (s)	AVG	35.09	7.13
	SD	55.17	19.85

TABLE III. RUNTIME LATENCY; AVG FOR AVERAGE; SD FOR STANDARD DEVIATION; THE RESULTS ARE BASED ON THE RAW DATA FROM 1,513 USERS.

B. Accuracy

1) Offline Learning - experimental data with ground truth: Because the experimental data are labeled, our online learning algorithm is not needed in this scenario. We simply train the classifier with the whole training set by using `Weka`, where the classification threshold γ is set to 0.5. Regarding the configuration of SVM, we set the cost value as 1.0 and `gamma` as 0.0. Figure 7 shows the classification performance of our system for users in the steady state and the moving state, respectively. All the trained classifiers for the 10 participants have the values P_{owner} and R_{other} higher than 90% in both of the two motion states. In particular, the average values of P_{owner}, R_{owner}, P_{other} and R_{other} are 94.76%, 71.76%, 77.41%, and 96.53% for the steady state. As for the moving state, the values are 94.15%, 64.37%, 74.07%, and 95.80%. The average accuracy for the steady state is 84.15%, and that for the moving state is 80.09%. The results indicate that RISKCOG achieves the low false positives, while the number of false negatives is relatively high. In our training set organization, we set the ratio of the number of positive samples (owner) to that of negative samples (other users) as 1:5. It means the classifier can accurately recognize the unauthorized users' patterns, while some gestures of the authorized owner will be missed. In the scenario of risk management, a false positive means the illegal access to the user's account that is more important than a false negative (false alarm) damaging the user experience. Thus, we pay more attention to restricting the false positives when configuring our system.

In Figure 8, we utilize the ROC curve to describe the true positive rate against the false positive rate at various threshold γ, which starts from 0 to 1 with step growth 0.01. Given the value of γ, we calculate the average values of TPR and FPR for all the participants. The areas of the two curves for motion/steady states are 0.9513 and 0.9043, which means that RISKCOG has enough space for tuning given various requirements of sensitivity and specificity.

2) Online Learning - raw data without ground truth: After the data preprocessing, the number of samples in the training/test set is summarized in Table III. The training set will be divided into 10 chunks, which are utilized to get the classifier based on the semi-supervised online learning algorithm. Regarding the conditions of accepting a chunk of data and training termination (Section III-C), we set the parameters λ, β, A, and V as 0.5, 0.1, 0.8, and 0.05, where we observe the average number of chunks taken to finish the online learning is 5.8.

In Figure 8, the areas of the two curves for the moving state and the steady state are 0.9719 and 0.9506 for all the 1,513

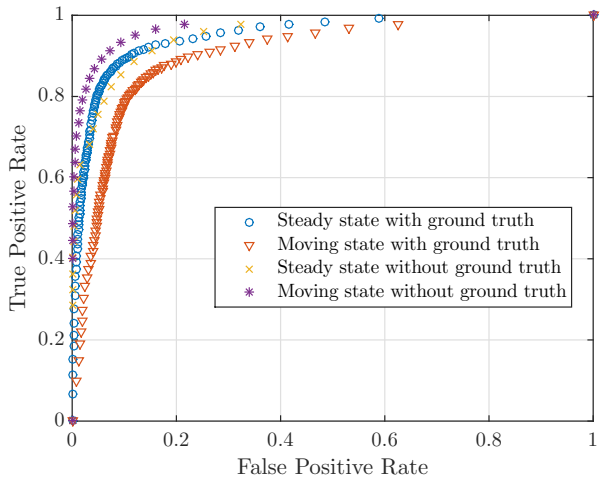


Fig. 8. ROC curve for 10 participants with ground truth and 1,513 users without ground truth

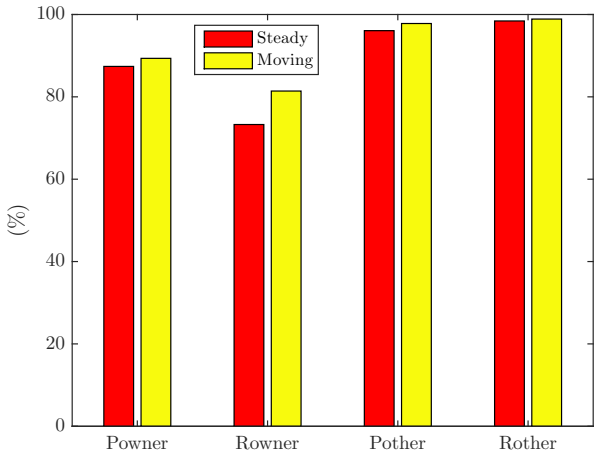


Fig. 9. Accuracy performance on raw data without ground truth for 1,513 users

users without ground truth. The performance is slightly better than that in the laboratory setting. It is attributed to the bigger size of the training set and the stratified sampling applied, which allows the generated classifier to well differentiate the authorized device owner from others. Figure 9 fully illustrates the classification accuracy of RISKCOG for all the 1,513 users in the wild. Our system achieves the average values of P_{owner}, R_{owner}, P_{other} and R_{other} as 87.39%, 73.28%, 96.07%, and 98.43% for the steady state, and 89.35%, 81.41%, 97.81%, and 98.89% for the moving state. And the average accuracy values for the two states are 93.77% and 95.57%. Even with those challenges in the practical deployment, such as imbalanced data set and unlabeled data, our design including the stratified sampling and semi-supervised online learning algorithm allows RISKCOG to have the performance that is similar to that in the laboratory setting. The prediction accuracy for the steady state is slightly lower than that for the moving state, from which we can see that our feature set is nearly independent of user's motion state and RISKCOG is able to provide the fully

TABLE IV. THE OVERHEAD RESULTS ON THREE DIFFERENT SMARTPHONES; THE EXPERIMENT LASTS THREE HOURS.

Phone Type	Battery Consumption (mAh)	Network Traffic (KB/h)	CPU (%)	Memory (MB)
MI 4	128.68/3080	58.93	0.1	10.88/2864.29
Samsung N9100	132.48/3000	56.12	0.1	14.32/2778.23
Sony C2104	113.82/1750	60.47	0.1	9.23/840.66

continuous protection on the user's account. All the previous studies [51], [31], [32], [43], [48], [56] rely on the features, which are only available when the user is moving, such as step cycle.

C. Overhead

We measure the runtime latencies of the training and the test phases, which are listed in Table III. We set up the test on a server with an Intel Xeon E5 CPU and 64G of physical memory running on Ubuntu 14.04. On average, RISKCOG is able to analyze the user's motion data for 10 days and train the classifiers corresponding to the steady/moving states within 170s.

On the client side, we utilize the tool Emmagee[1] for the benchmarking test regarding the impact on battery consumption, network traffic, CPU, and memory usage introduced by RISKCOG. Emmagee is a mobile application that is able to sample the hardware resource usage of an app on device. The client application only needs to collect and upload the real-time sensor data. We use three different models of phones to test the resource consumption, and the whole experiment lasts three hours. Table IV shows the related results where we record the average value of each type of resource during the experiment duration. Only one percent of the battery is required by our application in one hour. We can see that RISKCOG has tiny resource consumption regarding battery, CPU, and memory. The network traffic is acceptable, especially when user has WiFi connection. Moreover, we can apply data compression to further optimize the network usage.

D. Resistance against brute-force attacks

For password cracking, the brute-force attack tries a huge set of possible keys against the credential verification module. We verify the identity of user by a sophisticated set of features collected from motion sensors, and the attacks in our scenario are thus based on a large set of sensor data generated by users other than the authorized device owner. To evaluate the robustness of trained classifiers, we set up two types of brute-force attacks.

1) Automatic attack: The sensor data generator first randomly selects one dimension (x, y, z axes) that aligns with the gravity direction. The acceleration on that dimension is set randomly within a range around positive/negative gravity acceleration value. For the nine values collected from motion sensors, the starting points are generated around baseline values within predefined ranges, which we called initial deviation range. Here we define this point as the initial point.

We observe that the motion events by the human attackers who try to bypass our check have the property of temporal continuity. Our random data generator also follows this rule, where the step deviation range is computed by evaluating the deviation of the current slot to the previous slot within a range. Moreover, the generated data is bounded with the lower bound and upper bound to guarantee that they confirm to the laws of physics.

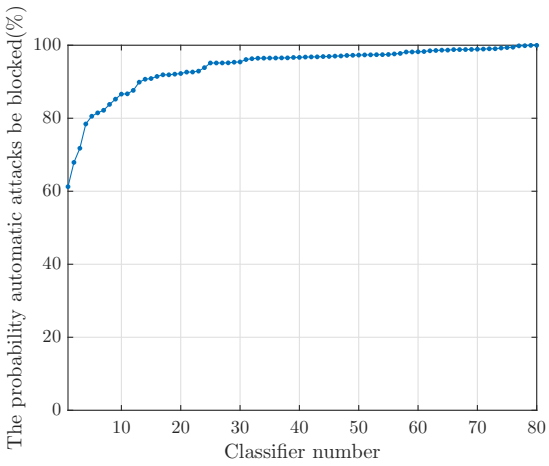


Fig. 10. Automatic brute-force attack results

In daily usage, users are possible to change their ways of handling the phone, which would break the continuity of sensor data. We thus define a continuous interval, in which the consecutive samples are continuous, and entering a new continuous interval involves the generation of a new initial point. The setup of the random data generator is summarized in Table V, which is defined by observing the data from human users.

On one hand, our brute-force attack simulates the human behavior as much as possible. On the other hand, it is extremely difficult for a human to launch such an attack by playing with the phone for two reasons:

- Amount of data. The sampling frequency of our data collection mechanism is 50Hz, and we only collect the effective data for one minute per hour. Thus, total 72,000 data samples can be gathered within one day at most, which is far lower than those generated by our brute-force attack.
- Data coverage. Manually handling the smartphone only involves a limited number of gestures. However, of the nine values collected from the acceleration sensor, gyroscope sensor, and gravity sensor, our brute-force attack generates the samples which have even distributions and fully cover the ranges consistent with physics.

Given the trained classifiers of the 80 users who are randomly selected from the overall 1,513 users, we generate the fake data including 600K samples and check the percentage of samples which are correctly labeled as unauthorized. The

[1]Emmagee. https://github.com/NetEase/Emmagee

TABLE V. SETUP OF AUTOMATIC BRUTE-FORCE ATTACK; CONTINUOUS INTERVAL LENGTH IS 10000; GRAVITY DIRECTION IS RANDOMLY CHOSEN FROM THE THREE DIMENSIONS OF THE ACCELERATION SENSOR.

	Baseline Values	Initial Deviation Range	Step Deviation Range	Lower Bound	Upper Bound
Gravity (x,y,z)	+9.8/-9.8	+0.06/-0.06	+0.2/-0.2	-11.0	+11.0
Acceleration (x, y, z)	0	+3.0/-3.0	+0.2/-0.2	-5.0	+5.0
Gyroscope (x, y, z)	0	+0.2/-0.2	+0.05/-0.05	-0.2	+0.2

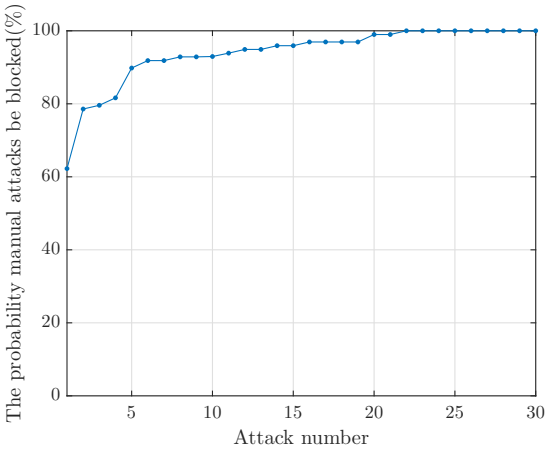


Fig. 11. Manual brute-force attack results

results of our automatic brute-force attack are illustrated in Figure 10. The average percentage of samples successfully blocked by our system is 90.44%

2) Manual attack: We also have humans to launch the brute-force attack. One classifier is trained for the authorized device owner by fingerprinting the usage manner. 3 participants handle the same phone with various gestures for 10 times, where a participant generates 84 samples each time. Those samples are checked against the classifier, and we identify the percentage of samples which are correctly labeled as other users. Figure 11 shows the evaluation results. Similarly, RISKCOG blocks the attacks by human with probability over 93.85%

VI. DISCUSSION

In this section, we first discuss two scenarios which might allow the attackers to evade our detection. In the first scenario, the motion sensor data are fetched from the related system APIs. The attackers are able to customize their return values by hooking those APIs. For example, RISKCOG may always read the same values from motion sensors and the verification will be bypassed. However, hooking the low-level system APIs requires the root privilege of the device, which is a too strong assumption. Moreover, we can enforce root detection and selectively deploy our service on those devices without being rooted.

In the second scenario, the data from the motion sensors are publicly readable. Any app can fetch the data, which can be utilized to reproduce our learning process, get the classifier, and even allow the attacker to deduce the specific usage manner of the authorized owner. We can manipulate the training set, such as interchange features in the sample vectors, e.g. $< F_1, F_2, ... > \Rightarrow < F_2, F_1, ... >$, which is totally invisible to the attackers. When RISKCOG attempts to identify the user, the verification phrase will not be affected if the same rule of manipulating sample vector is applied.

The existing RISKCOG is deployed in the client-server model, which requires the network connection to verify the user identity against the trained model. One may argue that the adversary can bypass the check by disabling the network. However, the majority of the sensitive services protected by RISKCOG themselves requires the network connection, such as account login, which are anyway not accessible in this case. Even if local verification is required, we can push the trained classifier to the client side when the device is online. The verification phase can be migrated to the local client by leveraging some light-weighted machine learning frameworks, such as TensorFlow [19], which are suitable for mobile devices. Thus, the verification can still operate when the device is offline. The training phase cannot be deployed on the client side because exposing the trained classifier to other local third-party applications introduces extra security risks.

VII. RELATED WORK

In this section, we will discuss the relevant studies, including payment security, gait recognition, and sensor-based user authentication.

Payment security. LevelUp [10] is a company providing smartphone-based credit card authentication services. The user's phone is bound with his/her credit card. LevelUp randomly generates QR Codes based on the credit card information uploaded to the server by the user. The merchant has the merchant version of LevelUp mobile app, where the phone is treated as a card reader. Upon a transaction, a user presents the QR Code to the merchant, and the merchant scan it with his/her device's camera, which is uploaded to the server and verifies if the transaction succeeds. Square Payment [17] and Paydiant [16] adopt the similar design based on QR Codes. Square Payment additionally uses geofencing to locate a user and display customers' name and photo on merchant reader's screen. Paydiant provides a special software designed for a certain version of magnetic card readers to display a QR Code for customers. Google Wallet [9] utilizes the NFC technology where a user is able to proceed the transaction by tapping her/his cell phone near a specially designed device. The smartphone and its counterpart talk through radio communication and exchange information. Moreover, TagPay Platform [18] adopts Near Sound Data Transfer (NSDT) that enables the communication between two smartphones though the speaker via the sound channel. The learning-based risk management mechanism has not been applied to any of the solutions above.

Gait recognition. The vision-based approaches [55], [42], [49], [58] with the background segmentation technique were proposed to recognize human's gait, where the camera with suitable optics of acquiring the gait data is required. Some

studies also investigated recognizing gait by sensors on the floor [45], [54]. The newest solution was based on wearing the motion sensors on the body. Some work used to record the acceleration data for human physical activity classification [59], [66], [50]. Accelerometer data were also used for gait identification [37], [61]. Brezmes et al. tried to identify different human activities by cell phone sensor [26]. Other methods, described by Chu et al. and Marasovic et al, proposed a gesture recognition algorithm based on the acceleration feature [52], [28]. Coskun et al. explored how much the position information of mobile devices increased the accuracy of recognizing activities, and on average the results were similar for position specific and position independent recognition [29]. These studies focus on classifying a user's daily activity and gait, rather than verifying the identity. Our approach differs from these above, we do not focus on gait recognition and simply classify the gait as steady state and moving state when users are using their phone. The experiment results show that only two classes of the gait performs well on user authentication.

Sensor-based user authentication. The use of sensor data for user authentication has been explored in recent years. In 2005, Ailisto et al. first used wearable accelerometers to identify people, they placed the device behind at a person's waist and the recognition accuracy was about 80% [23]. Gafurov et al. placed the device on the hip, arm and ankle to record the acceleration for unobtrusive user authentication [38]. Derawi et al. proposed a stable cycle detection to analyze acceleration signal [31], Trung et al. used accelerometers and gyroscopes to evaluate the identification accuracy separately [64]. With time, some researchers began to record the accelerometers with mobile phones, Kwapisz et al. and Derawi et al. made use of phone-based acceleration sensor to identify and authenticate cell phone users [48], [32]. Ren et al. proposed a user verification system leveraging unique gait patterns derived from acceleration readings to detect possible user spoofing in mobile healthcare system [56]. These approaches require that the sensors are placed in specified body locations and the samples in the training set are well labeled. Lu et al. [51] overcame these limitations by projecting the data samples onto a global coordinate system for the resilience to device orientation and handling the unlabeled data with an unsupervised learning algorithm. However, it relies on user inputs to update the model and reduce false negatives. While our semi-supervised online learning algorithm not only requires no user action but also has lower latency when processing unlabeled data compared with an unsupervised one. All the proposed solutions require the user's movement during learning and verification because the model depends on the features, such as step cycle. RISKCOG is able to verify the identity when the user is steady simply by the manner of handling the device. Given the rare limitations on the application scenario, our system can provide a fully continuous protection on a user's sensitive services. Furthermore, we consider a set of users that is significantly larger than in most of the prior studies, which challenges us with the issue of imbalanced data set.

VIII. CONCLUSION

In this paper, we present the system RISKCOG to provide a fully continuous and implicit user identity verification with a learning-based approach. Regarding the issue of privacy preserving in the context of social impact, our feature set only involves the insensitive motion sensors, which are commonly available on fragmented Android devices. It supports deploying our service at the device level with high portability. The trained classifier depicts the owner's specific manner of handling his/her smartphone. Unlike previous related studies, we have no requirement on the user's motion state or the device placement. Our data collection and data preprocessing modules ensure that those samples that are effective to differentiate users are extracted. Moreover, we collect the data from motion sensors after recognizing the start event of a new application, which makes sure the trained classifier will not be affected by application-specific gestures. As deploying RISKCOG in the production environment on a large scale, several new issues are exposed, such as the imbalanced data set and training set without ground truth. We design our stratified sampling method where the number of negative samples in the training set is comparable to that of positive samples and the representability is maintained. Furthermore, we design a semi-supervised online learning method, which allows us to eliminate the high time latency when handling unlabeled data with unsupervised methods. The design allows us to be the first to resolve the user identification problem implicitly with data collected from industry vendor. We achieve the classification accuracy values 93.77% and 95.57% among the 1,513 users for the steady state and the moving state. RISKCOG can also be applied to any mobile application other than payment, such as social applications.

REFERENCES

[1] Alibaba uses facial recognition tech for online payments. http://www.computerworld.com/article/2897117/alibaba-uses-facial-recognition-tech-for-online-payments.html.

[2] Android BroadcastReceiver. https://developer.android.com/reference/android/content/BroadcastReceiver.html.

[3] Android fragmentation report august 2014 - opensignal. http://opensignal.com/reports/2014/android-fragmentation/.

[4] Android Sensors Overview. https://developer.android.com/guide/topics/sensors/sensors_overview.html.

[5] Android service. https://developer.android.com/guide/components/services.html.

[6] Android UsageStatsManager. https://developer.android.com/reference/android/app/usage/UsageStatsManager.html.

[7] Android's biggest problem is operating system fragmentation. http://o.canada.com/technology/personal-tech/androids-biggest-problem-is-operating-system-segmentation.

[8] Dalvik bytecode. https://source.android.com/devices/tech/dalvik/dalvik-bytecode.html.

[9] Google wallet. https://www.google.com/wallet/.

[10] LevelUp. https://www.thelevelup.com/.

[11] LexisNexis True Cost of Fraud mCommerce. http://lexisnexis.com/risk/downloads/whitepaper/true-cost-fraud-mobile-2014.pdf.

[12] LibSvm. https://www.csie.ntu.edu.tw/~cjlin/libsvm/.

[13] Mobile payments will triple in the US in 2016. http://www.emarketer.com/Article/Mobile-Payments-Will-Triple-US-2016/1013147.

[14] Near field communicationi. https://en.wikipedia.org/wiki/Near_field_communication.

[15] Novel Malware XcodeGhost Modifies Xcode, Infects Apple iOS Apps and Hits App Store. https://github.com/jamiebullock/LibXtract.

[16] Paydiant. http://www.paydiant.com/.

[17] Square payments. https://squareup.com/pos/payments.

[18] TagPay platform. http://en.tagpay.fr/tagpay-platform/.

[19] TensorFlow - an Open Source Software Library for Machine Intelligence. https://www.tensorflow.org/.

[20] Vowpal Wabbit. https://github.com/JohnLangford/vowpal_wabbit.

[21] Weka. http://www.cs.waikato.ac.nz/ml/weka/.

[22] You won't believe the 20 most popular cloud service passwords. https://www.skyhighnetworks.com/cloud-security-blog/you-wont-believe-the-20-most-popular-cloud-service-passwords/.

[23] H. J. Ailisto, M. Lindholm, J. Mantyjarvi, E. Vildjiounaite, and S.-M. Makela. Identifying people from gait pattern with accelerometers. In *Defense and Security*, pages 7–14. International Society for Optics and Photonics, 2005.

[24] A. R. Beresford and F. Stajano. Location privacy in pervasive computing. *IEEE Pervasive computing*, 2(1):46–55, 2003.

[25] C. Bettini, X. S. Wang, and S. Jajodia. Protecting privacy against location-based personal identification. In *Workshop on Secure Data Management*, pages 185–199. Springer, 2005.

[26] T. Brezmes, J.-L. Gorricho, and J. Cotrina. Activity recognition from accelerometer data on a mobile phone. In *Distributed computing, artificial intelligence, bioinformatics, soft computing, and ambient assisted living*, pages 796–799. Springer, 2009.

[27] I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani. Crowdroid: behavior-based malware detection system for android. In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, pages 15–26. ACM, 2011.

[28] H.-C. Chu, S.-C. Huang, and J.-J. Liaw. An acceleration feature-based gesture recognition system. In *SMC*, pages 3807–3812, 2013.

[29] D. Coskun, O. D. Incel, and A. Ozgovde. Phone position/placement detection using accelerometer: Impact on activity recognition. In *ISSNIP*, pages 1–6, 2015.

[30] A. Das, N. Borisov, and M. Caesar. Tracking mobile web users through motion sensors: Attacks and defenses. In *Proceedings of the 23rd Annual Network and Distributed System Security Symposium (NDSS)*, 2016.

[31] M. O. Derawi, P. Bours, and K. Holien. Improved cycle detection for accelerometer based gait authentication. In *IIH-MSP*, pages 312–317, 2010.

[32] M. O. Derawi, C. Nickel, P. Bours, and C. Busch. Unobtrusive user-authentication on mobile phones using biometric gait recognition. In *IIH-MSP*, pages 306–311, 2010.

[33] S. Dey, N. Roy, W. Xu, R. R. Choudhury, and S. Nelakuditi. Accelprint: Imperfections of accelerometers make smartphones trackable. In *NDSS*. Citeseer, 2014.

[34] S. Fahl, M. Harbach, T. Muders, L. Baumgärtner, B. Freisleben, and M. Smith. Why eve and mallory love android: An analysis of android ssl (in) security. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 50–61. ACM, 2012.

[35] A. P. Felt, S. Egelman, and D. Wagner. I've got 99 problems, but vibration ain't one: A survey of smartphone users' concerns. In *ACM SPSM*, 2012.

[36] D. Florencio and C. Herley. A large-scale study of web password habits. In *Proceedings of the 16th international conference on World Wide Web*, pages 657–666. ACM, 2007.

[37] J. Frank, S. Mannor, and D. Precup. Activity and gait recognition with time-delay embeddings. In *AAAI*. Citeseer, 2010.

[38] D. Gafurov, K. Helkala, and T. Søndrol. Biometric gait authentication using accelerometer sensor. *Journal of computers*, 1(7):51–59, 2006.

[39] A. Ghodsi. Dimensionality reduction a short tutorial. *Department of Statistics and Actuarial Science, Univ. of Waterloo, Ontario, Canada*, 37:38, 2006.

[40] P. Golle and K. Partridge. On the anonymity of home/work location pairs. In *International Conference on Pervasive Computing*, pages 390–397. Springer, 2009.

[41] M. Grace, Y. Zhou, Q. Zhang, S. Zou, and X. Jiang. Riskranker: scalable and accurate zero-day android malware detection. In *MobiSys*. ACM, 2012.

[42] J. Han and B. Bhanu. Individual recognition using gait energy image. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(2):316–322, 2006.

[43] C. C. Ho, C. Eswaran, K.-W. Ng, and J.-Y. Leow. An unobtrusive android person verification using accelerometer based gait. In *Proceedings of the 10th International Conference on Advances in Mobile Computing & Multimedia*, pages 271–274. ACM, 2012.

[44] B. Hoh, M. Gruteser, H. Xiong, and A. Alrabady. Enhancing security and privacy in traffic-monitoring systems. *IEEE Pervasive Computing*, 5(4):38–46, 2006.

[45] J. Jenkins and C. Ellis. Using ground reaction forces from gait analysis: body mass as a weak biometric. In *Pervasive Computing*, pages 251–267. Springer, 2007.

[46] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer, 1998.

[47] J. Krumm. Inference attacks on location tracks. In *International Conference on Pervasive Computing*, pages 127–143. Springer, 2007.

[48] J. R. Kwapisz, G. M. Weiss, and S. A. Moore. Cell phone-based biometric identification. In *BTAS*, pages 1–7, 2010.

[49] Z. Liu and S. Sarkar. Improved gait recognition by gait dynamics normalization. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6):863–876, 2006.

[50] X. Long, B. Yin, and R. M. Aarts. Single-accelerometer-based daily physical activity classification. In *EMBC*, pages 6107–6110, 2009.

[51] H. Lu, J. Huang, T. Saha, and L. Nachman. Unobtrusive gait verification for mobile phones. In *Proceedings of the 2014 ACM international symposium on wearable computers*, pages 91–98. ACM, 2014.

[52] T. Marasović and V. Papić. Accelerometer-based gesture classification using principal component analysis. In *SoftCOM*, pages 1–5, 2011.

[53] B. F. Murphy. Network penetration testing and research. 2013.

[54] K. Nakajima, Y. Mizukami, K. Tanaka, and T. Tamura. Footprint-based personal recognition. *Biomedical Engineering, IEEE Transactions on*, 47(11):1534–1537, 2000.

[55] M. S. Nixon, J. N. Carter, D. Cunado, P. S. Huang, and S. Stevenage. Automatic gait recognition. In *Biometrics*, pages 231–249. Springer, 1996.

[56] Y. Ren, Y. Chen, M. C. Chuah, and J. Yang. User verification leveraging gait recognition for smartphone enabled mobile healthcare systems. *Mobile Computing, IEEE Transactions on*, 14(9):1961–1974, 2015.

[57] L. Rocchi, L. Chiari, and A. Cappello. Feature selection of stabilometric parameters based on principal component analysis. *Medical and Biological Engineering and Computing*, 42(1):71–79, 2004.

[58] S. Sarkar, P. J. Phillips, Z. Liu, I. R. Vega, P. Grother, and K. W. Bowyer. The humanid gait challenge problem: Data sets, performance, and analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(2):162–177, 2005.

[59] M. Sekine, T. Tamura, M. Akay, T. Fujimoto, T. Togawa, and Y. Fukui. Discrimination of walking patterns using wavelet-based fractal analysis. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 10(3):188–196, 2002.

[60] D. Sounthiraraj, J. Sahs, G. Greenwood, Z. Lin, and L. Khan. Smv-hunter: Large scale, automated detection of ssl/tls man-in-the-middle vulnerabilities in android apps. In *NDSS*, 2014.

[61] S. Sprager and D. Zazula. A cumulant-based method for gait identification using accelerometer data with principal component analysis and support vector machine. *WSEAS Transactions on Signal Processing*, 5(11):369–378, 2009.

[62] V. Svetnik, A. Liaw, C. Tong, J. C. Culberson, R. P. Sheridan, and B. P. Feuston. Random forest: a classification and regression tool for compound classification and qsar modeling. *Journal of chemical information and computer sciences*, 43(6):1947–1958, 2003.

[63] J. E. Trost. Statistically nonrepresentative stratified sampling: A sampling technique for qualitative studies. *Qualitative sociology*, 9(1):54–57, 1986.

[64] N. T. Trung, Y. Makihara, H. Nagahara, Y. Mukaigawa, and Y. Yagi. Performance evaluation of gait recognition using the largest inertial sensor-based gait database. In *IAPR*, pages 360–366, 2012.

[65] E. Vazquez-Fernandez and D. Gonzalez-Jimenez. Face recognition for authentication on mobile devices. *Image and Vision Computing*, 2016.

[66] N. Wang, E. Ambikairajah, B. G. Celler, and N. H. Lovell. Accelerometry based classification of gait patterns using empirical mode decomposition. In *ICASSP*, pages 617–620, 2008.

[67] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for svms. 2000.

[68] J. Yeo. Using penetration testing to enhance your company's security. *Computer Fraud & Security*, 2013(4):17–20, 2013.