RISKCOG: Unobtrusive 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 face snapshot, the existing user identification 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. Moreover, the current user identification countermeasure deploys at the application level, where each mobile payment service vendor has its channel of data collection individually. They cannot share their data, making it impossible to reuse the detection results for other apps.

In this paper, we propose the system RISKCOG, which solves the problem of identifying the authorized device owner by the data collected from the motion sensors with a learning-based approach. 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. Our feature set is independent of a user's motion state and has no requirement of user movement or fixed device placement. Moreover, we resolve the issues of the imbalanced dataset with our stratified sampling and missing of ground truth with a semi-supervised learning algorithm. Along with the design of offline verification, our system can protect the user in any challenging scenario, even in the industry product. For the data collected from 1,513 users, RISKCOG identifies the authorized owner in steady/moving states with the accuracy values 93.77% and 95.57%.

1 Introduction

Smartphones provide users with various functionalities, among which the popularity of mobile payment is growing exponentially. Based on the report by eMarketer [9], 37.5 million users are expected to use mobile payments in the year 2016. Although mobile payments have benefitted users immensely, they also introduce several security threats. Among them, human-driven risks, 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 [7], 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.

Human-driven risks still lack effective countermeasures. The traditional user authentication mechanism 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. Learning-based user identification approaches are proposed. They construct the model to describe the usage pattern of the authorized phone owner, such as the locations he/she frequently visits and face snapshot. Comparing with the simple login credential, the user identification mechanism utilizes a diverse set of features to verify user's identity, and it is thus much harder to get bypassed. The learning-based approach can be applied to the scenario of payment, while it fundamentally is a new solution to the general web/mobile services with the requirement of user identification. 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, However, the learning-based approaches will detect the unauthorized user implicitly. 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.

The ideal user identification system should work anywhere, anytime, for any users, and be easily adopted, which entail the following requirements:

- **Privacy-preserving.** Heavily tracking user privacy will raise the user's concern on privacy leakage [21], e.g., face snapshot, location.
- **Device-level detection as a generic service to multiple apps.** A generic service allows the detection results to be reused and removes the redundancy in terms of the data collected from these apps.
- **Work with any user at any status.** The training and verification procedures should not depend on a user's motion state or the device placement, which achieves the continuous protection on the user's account.
- **Work with Internet scale users.** When the user base gets large, an effective sampling method is needed to avoid a biased training set.
- **Train automatically without labelled data.** The assumption that each training sample includes the

Table 1: Comparison with related studies

Study	Require user movement	Fixed/dynamic device placement	Scale (#Users)	Require label	Offline real-time verification
RISKCOG	No	Dynamic	>1,500	No	Low latency: 3237.7 ms
Lu et al. [31]	Yes	Dynamic	<50 (walking detector training), <50 (supervised training), <10 (unsupervised training)	No	High latency: 27863.586 ms
Derawi et al. [19]	Yes	Fixed	<100	Yes	No
Ho et al. [25]	Yes	Fixed	<50	Yes	No
Kwapisz et al. [29]	Yes	Fixed	<50	Yes	No
Ren et al. [33]	Yes	Fixed	<50	Yes	Not implemented
Related challenge	Lack of feature	Data availability & dynamic device placement	Imbalanced data set	Unlabeled data	Constrained mobile environment

ground truth is unrealistic. It is unknown whether the samples for training are collected when the device is handled by the authorized device owner or not.

- **Work at constrained environment even when being disconnected.** Leveraging the sever's computational resource for both training and test is limited by the server availability issue. The requests are sent frequently and query to the server is not reliable as the user base gets larger. The backend server is possible to have memory/disk failures, and network participations. This issue will become even severer in the disconnected or weakly connected environment, e.g. subway, crowd. Given our intention to deliver the real-time user authentication, it is not feasible to heavily rely on the remote server in our deployment.

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. The first two requirements have motivated some recent gesture-based user identification studies [31, 19, 25, 29, 33]. They recognize a user based on the data collected from motion sensors during the daily usage. However, none of them can fulfill the other four requirements. By comparing with the related studies listed in Table 1, we summarize the following challenges:

(1) Lack of feature. 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 [6] 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 app to work, e.g., the pressure sensor needs to be bound to a view element for collecting the area and force of user's touch event. That requires extra developer support at the app level. Due to the least user privacy requirement, any feature involving user's personal information in the context of social impact is also not available.

(2) 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 apps, e.g., the frequent typing gestures in a chatting app compared with the phone rotation in a race car game. The app-specific pattern will challenge the classification accuracy.
Previous studies [19, 25, 29, 33] also have the strong assumption of fixed device placement, for example, the smartphone needs to be put in the pocket of trousers and the orientation remains unchanged. To offer a continuous verification, we should not have any requirement of user movement.

(3) 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 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, previous related studies have the much smaller number of experiment subjects compared to us.

(4) Unlabeled data. The proposed prototype systems [19, 25, 29, 33] 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, the well-labeled data is not

available in the practical scenario. For example, the device owner may give the phone to her/his family member during data collection. 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.

(5) Constrained mobile environment. The sophisticated learning algorithm with high time complexity can be used on the server side, such as the random forest algorithm [36]. During the CPU-intensive task of identity verification, the whole model is loaded into the memory of mobile devices. A complicated classification model with high prediction accuracy requires heavy computation resource, which cannot be supported by the constrained mobile devices. We thus need to investigate the tradeoff between classification accuracy and computational resource requirement and optimize our learning algorithm to deliver a light-weighted and real-time offline identity verification service. To the best of our knowledge, the work by Lu et al. [31] is the only one that investigates the offline verification. However, it takes about half a minute to finish the identity verification, where the complex gait analysis based on Universal Background Model (UBM) dominants the identification latency 13993.379 ms.

We design a system, called RISKCOG, to provide the offline user identity verification service. It is based on semi-supervised learning algorithm to identify phone owner simply based on the motion sensors with no requirement of user's motion state and device placement. Each data sample is collected, preprocessed, and further classified by whether the user is in a steady or moving state. After that, two parallel classifiers are trained for these two states. 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/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, which thus does not require the fixed device placement.
- 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 app-specific pattern are filtered out. We implement a preprocessing procedure 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 [37] 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.
- 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.
- We decouple the verification from the server side and resolve the issue of availability in the disconnected environment. Our optimization of learning algorithm produces a light-weighted identity verification service on resource-constrained mobile devices. It only takes 3237.7 ms to finish the verification.

To the best of our knowledge, we are the first to fulfill all the requirements of user identification solution discussed above and achieve high accuracy for unobtrusive 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. We also produce an Android app including our user identification mechanism for general usage apart from payment [1].

The remainder of this paper is organized as follows. Section 2 presents a brief background of our work. In Section 3, we cover RISKCOG design in detail, which is then followed by an explanation of the implementation procedures in Section 4. Section 5 deals with the overall evaluation of our system. Section 6 and 7 include discussion about relevant work and Section 8 includes our concluding remarks.

2 Background

2.1 Authentication, and user identification

Authentication is used to prevent the unauthorized parties from using the sensitive services. Currently, the credential is the predominant form of an authentication sys-

[1] RISKCOG client app. `http://139.224.207.24/SensorDemo-release.apk`

tem, 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 [11] 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. [22] 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, user identification introduces learning-based approaches. It 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 from the user, and it is thus to enforce the continuous protection without sacrificing the usability.

2.2 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 [20, 18]. 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 involved are more sensitive. It is straightforward to know who the user is when face recognition is utilized for the purpose of authentication [38, 1]. As for geo-location, it is able to identity the owner in the physical world as stated in previous studies [14, 24, 13, 26, 28].

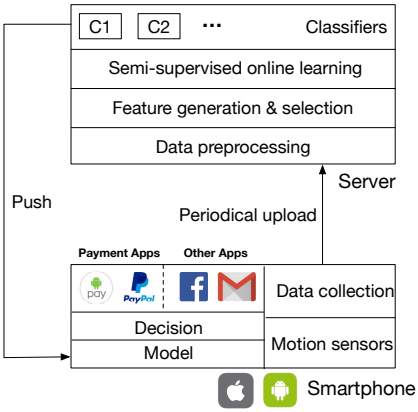


Figure 1: System architecture

Table 2: Availability of acceleration sensor (AC), gyroscope sensor (GY), and gravity sensor (GR); MS for marketshare

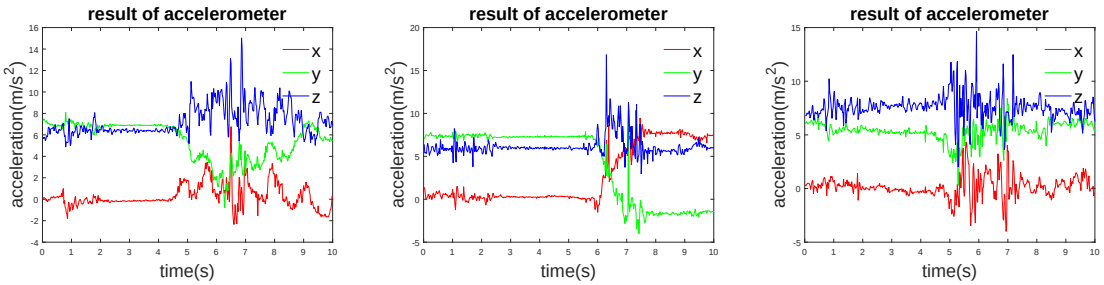
Brand	MS in 2016	Device	AC	GY	GR
Samsung	22.2%	Galaxy S7	D	D	D
		Galaxy S6	D	D	D
		Galaxy Note 2(3, 4)	D	D	D
		Galaxy CORE Prime	D	×	×
Apple	16.8%	iPhone 6s Plus	D	D	D
		iPhone 6(s)	D	D	D
		iPhone 5(s)	D	D	D
Huawei	9.3%	P9	D	D	D
Xiaomi	5.8%	MI 3(4, 5)	D	D	D
LG	5.0%	G5	D	D	D
OPPO	3.9%	R9	D	D	D

2.3 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. In Table 2, the three motion sensors are available on the popular devices surveyed, except Samsung Galaxy CORE Prime that was released in 2014.

3 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 target device labeled as 1 and the data from other devices 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



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

Figure 2: Acceleration sensor values in the temporal dimension

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, our semi-supervised online learning mechanism does not require the explicit label. It 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, the trained model will be pushed to the device. When the various mobile payment vendors receive the transaction requests from the smartphone, they just need to locally query whether the phone is being used by the authorized owner during the recent activities. Compared with the deployment model of existing learning-based user identification, 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 for user identification need to be uploaded only once rather than be sent to each vendor's server separately.

3.1 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:

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

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

We develop a mobile application for the purpose of data collection. It needs to 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. With our experiment shown in Figure 2, we observe that the sensor readings largely vary with the different types of apps 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 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` [3] 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 are thus collected 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. We have a participant to handle the phone and put a phone on a stationary plane. Then we get the boundaries of the gravity sensor readings on three dimensions by minimizing the errors

of device placement prediction.

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

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. 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. 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.

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. 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.

3.2 Feature generation and selection

For the classification, we only utilize data collected from acceleration and gyroscope 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

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

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.

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

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.

We generate a total of 56 features, which covers multiple moments and other commonly used statistical properties of the distribution. We separately utilize each potential feature to establish a classification model and evaluate the accuracy of these 56 models given the ground truth in the laboratory settings. It verifies the effectiveness of each single feature to depict the device owner's pattern. We also rank features based on their independent classification capabilities, which is further utilized in our stratified sampling.

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

$$\sum_{k=1}^{K}\sqrt{x^2(k)+y^2(k)+z^2(k)}/K$$

Table 3: Cluster separability; AVG for average; STD for standard deviation

	Sit	Walk
AVG(r_i)	**7.27**	**5.31**
STD(r_i)	2.24	1.93
AVG($dc_{i,j}$)	**10.66**	**18.83**
STD($dc_{i,j}$)	4.47	10.02

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.

Our system is developed based on the assumption that the distribution of sensor data is unique for each user. It can be utilized as the digital fingerprints to identify the authorized user of mobile devices. To verify this fundamental assumption, we conduct our experiment based on 1,513 users. Each user produces data samples which are 56-dimensional vectors. We calculate the mean value of each dimension and denote the center of all samples from user i as c_i. We record the average euclidean distance from each data sample by user i to the center as radius r_i. Then we denote the euclidean distance between each pair of centers as $dc_{i,j}$. As shown in Table 3, the average radius is much smaller than the average distance between centers of clusters. We thus verify the cluster is separable by our large-scale experiment.

3.3 Semi-supervised online learning algorithm

3.3.1 Training set

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 $\mathscr{F}_i, i = 1, \cdots, n$. 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 [37], 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.

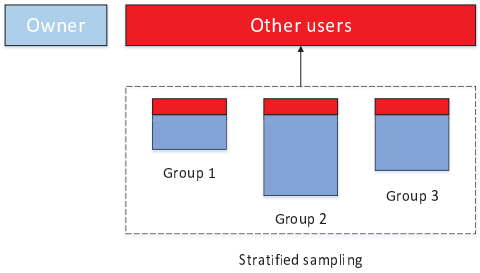


Figure 3: Training set construction

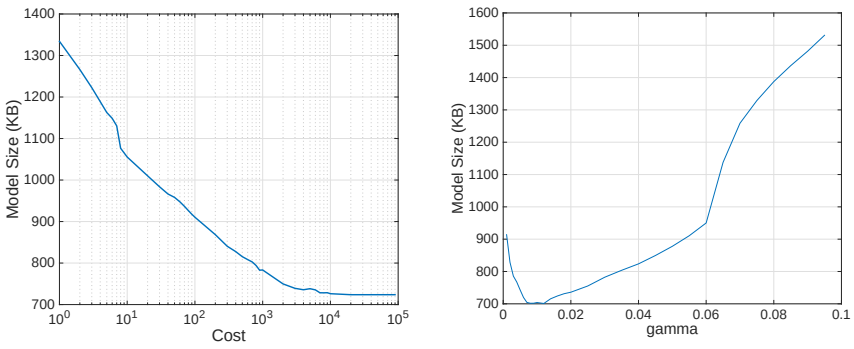
Our training set is constructed as the Figure 3. The ratio of the number of samples by the phone owner to that of other users is 1:5, which can be properly handled by normal learning algorithms, such as SVM. The optimal point is chosen by experiment. For effective training, the number of positive instances usually needs to exceed 4000. The specifically required number of instances 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 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 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.

3.3.2 Classification method

We choose Support Vector Machine (SVM) with radial basis function (RBF) kernel as our classification method for the following reasons:

(1) 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)



(a) C (logarithmic scale, $\varepsilon = 0.01$, $\gamma = 0.01$) (b) γ ($\varepsilon = 0.01$, $C = 100$)

Figure 4: Model size v.s. C and γ (Accuracy $\geq 90\%$)

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 will help us build a nonlinear classification boundary.

(2) 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. SVMs have been reported in many studies to work better with our situation [40, 27], for example, text classification.

(3) Dependent data structure. The random forest [36] 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 collected data 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 randomly splits 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.

3.3.3 Optimization

The size of model is essential when verifying the user identity offline. Considering the limited computational resources of mobile devices, a smaller model indicates the lower CPU and memory consumption in identity verification and lower traffic when pushing the model from the server to the smartphone. We conduct the grid search to find the optimal configuration of the parameters C and γ in the SVM with RBF kernel. We choose 80 users, who generate most number of data samples in daily usage, from our data set with 1,513 users that will be explained in detail in Section 5. Given the fixed parameter $\varepsilon = 0.01$, we change C from 1 to 90,000 and γ from 0 to 0.1. As shown in Figure 4, we find the model size decreases with the value of C. As the value of C exceeds 100, the system will get tiny decrement in the model size (cost C is depicted in the logarithmic scale). The model size will reach the minimal value when the value of γ equals to 0.01.

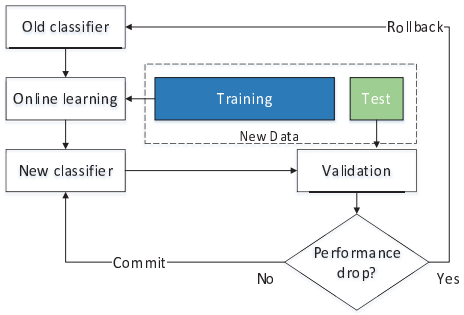


Figure 5: Semi-supervised online learning

The flowchart of our semi-supervised online learning algorithm is illustrated in Figure 5. 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 the misalignment 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.

3.4 Decision

Given a feature vector corresponding with one sliding 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

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

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

4 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` [4]. 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 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` [10] to extract the 56 features. We use `LibSvm` [5] to train our model for each person and enforce the offline identity verification with `AndroidLibSvm` [8].

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

5 Evaluation

5.1 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 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 3.1). 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 in our evaluation.

- True positive (TP). The authorized owner is correctly identified.
- False positive (FP). Other users are incorrectly identified as the authorized owner.

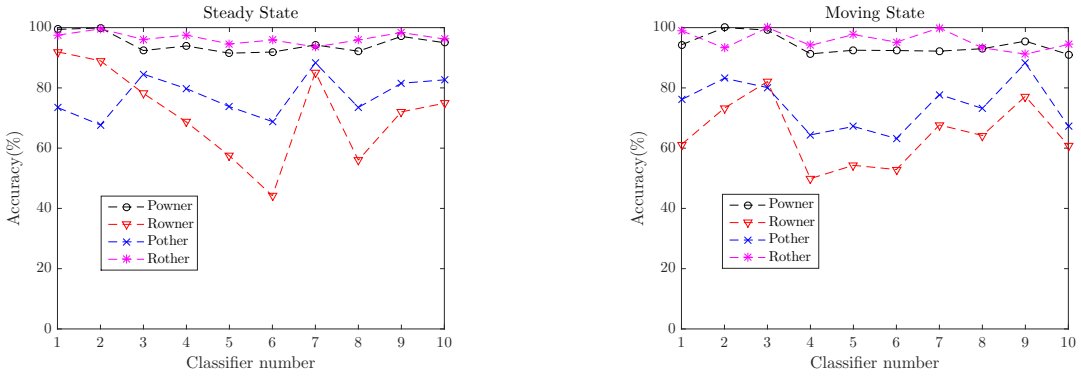


Figure 6: Accuracy performance on experimental data with ground truth for 10 participants

- 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 on the server side. Then we check the battery consumption, CPU, and memory usage of the phone when our client app collects data and verifies the user's identity.

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 θ. 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 θ.

The training module is deployed on the server side with `LibSVM`, where both the positive samples and negative samples are available. As discussed above, our evaluation of accuracy also involves both positive samples and negative samples and is conducted on the server. In the architecture of RISKCOG, the user identity verification is enforced offline with `AndroidLibSvm`, where only the data generated from the device (positive sample) are available. We verify that both `LibSVM` and `AndroidLibSvm` produce the same prediction result given the identical prediction model and data sample as input. It is thus valid to utilize our evaluation of accuracy to depict the effectiveness of RISKCOG to enforce user identity verification locally.

5.2 Accuracy

5.2.1 Batch 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, where the classification threshold θ is set to 0.5. Regarding the configuration of SVM, we set the cost value as 100 and γ as 0.01. Figure 6 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, i.e. the illegal access, while some gestures of the authorized owner will be missed. In user identification, a false positive, i.e. the illegal access to the user's account is more critical than a false negative (false alarm). Thus, we pay more attention to restricting the false positives when configuring our system.

		Steady	Moving
#Training samples	AVG	20648	9280
	STD	15660	10212
Training time (s)	AVG	148.36	21.21
	STD	177.23	24.06

Table 4: Training latency; AVG for average; STD for standard deviation; the results are based on the data from 1,513 users.

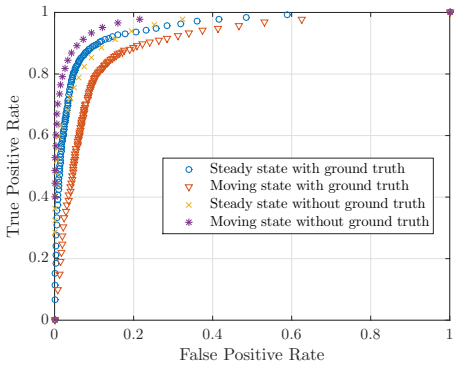


Figure 7: ROC curve for 10 participants with ground truth and 1,513 users without ground truth

In Figure 7, we use the ROC curve to depict the true positive rate against the false positive rate at various threshold θ, which starts from 0 to 1 with step growth 0.01. Given the value of θ, 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. RISKCOG thus has enough space for tuning given various requirements of sensitivity and specificity.

5.2.2 Online learning - raw data without ground truth

After the data preprocessing, the number of samples in the training set is summarized in Table 4. 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 3.3), 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 7, 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 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 8 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 continuous protection on the user's account. All the previous studies [31, 19, 25, 29, 33] rely on the features, which are only available when the user is moving, such as step cycle.

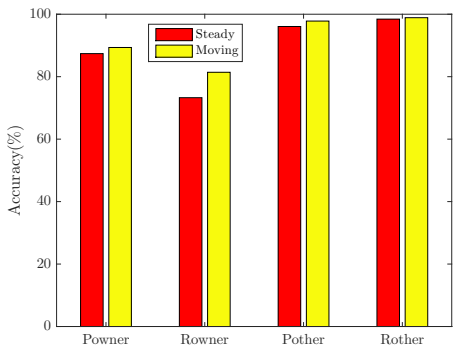


Figure 8: Accuracy performance on raw data without ground truth for 1,513 users

Table 6: Latency of offline user identity verification

Procedure	Time (ms)
Data collection	3211.6
Data preprocessing	0.5
Feature extraction	12.3
Decision	13.3
Overall	3237.7

5.3 Overhead

We measure the runtime latencies of the training phase on the server side, which are listed in Table 4. We set up the experiment 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 150s.

On the client side, we utilize the tool Emmagee[2] for the benchmarking test regarding the impact on battery consumption, CPU, and memory usage introduced by RISKCOG. Emmagee is a mobile app that is able to sample the hardware resource usage of an app on the device. Table 5 shows the related results. For the battery con-

[2]Emmagee. `https://github.com/NetEase/Emmagee`

Table 5: The overhead results on three different smartphones; the measurement of battery consumption lasts three hours.

		Data Collection		Identity Verification	
Phone Type	Battery Consumption (mAh)	CPU (%)	Memory (MB)	CPU (%)	Memory (MB)
Samsung N9100	132.5/3000	10.34	14.4	8.80	21.4
Sony Xperia Z2	113.8/3200	1.82	18.0	9.00	26.0
MI 4	128.7/3080	1.30	14.0	12.00	24.3

sumption, we let one participant use the client app for three hours, which includes both data collection and offline identity verification. Only one percent of the battery is required by our app in one hour. The CPU usage is over 10% on the device Samsung N9100 during data collection. The case does not happen on other two phones. It is possible that the higher CPU utilization is related to the low-level system implementation. Our optimization of SVM setup reduces the resource consumption of the CPU-intensive offline verification.

We also check the latency of offline user identity verification. We execute the procedures: data collection, data preprocessing, feature extraction, and decision for 1000 times on the device Samsung N9100 and record the average time for each step. The results are listed in Table 6. We can see the whole process can be finished within 3237.7 ms. The latencies introduced by steps other than data collection are negligible.

5.4 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.

5.4.1 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. 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 current slot differs from the previous slot by the small step deviation range. Moreover, the generated data is bounded with the lower bound and upper bound to guarantee that they confirm to the laws of physics.

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 7, 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:
(1) Amount of data. Our sampling frequency 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.
(2) 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 average percentage of samples successfully blocked by our system is 90.44%.

5.4.2 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. 10 participants handle the same phone with various gestures for 40 times, where a participant generates 49 samples each time. Those samples are checked against the classifier, and we identify the percentage of samples which are correctly labeled as other users. RISKCOG blocks the attacks by human with probability over 99.85%.

6 Discussion

We 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

Table 7: 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

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.

7 Related Work

Gait recognition. The state-of-the-art gait recognition solution was based on wearing the motion sensors on the body. Some work used to record the acceleration data for human physical activity classification [34, 39, 30]. Accelerometer data were also used for gait identification [23, 35]. Brezmes et al. tried to identify different human activities by cell phone sensor [15]. Other methods, described by Chu et al. and Marasovic et al, proposed a gesture recognition algorithm based on the acceleration feature [32, 16]. 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 [17].

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% [12]. Kwapisz et al. and Derawi et al. made use of phone-based acceleration sensor to identify and authenticate cell phone users [29, 19]. Ren et al. proposed a user verification system leveraging unique gait patterns derived from acceleration readings to detect possible user spoofing in the mobile healthcare system [33]. 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. [31] 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, and achieved the offline user identity verification. 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 a lower latency of handling unlabeled data in training compared with an unsupervised one. Moreover, with a radically different design without involving complex UBM and the extra step of feature extraction, we reduce the latency of offline verification by 90%. All the proposed solutions require the user's movement because the model depends on the features, such as step cycle. RISKCOG can verify the identity when the user is steady by the manner of handling the device.

8 Conclusion

In this paper, we present the system RISKCOG to provide a fully continuous and offline user identity verification with a learning-based approach. The trained classifier depicts the owner's specific manner of handling his/her smartphone based on the data collected from motion sensors. Unlike previous related studies, we have no requirement on the user's motion state or the device placement. Plus the offline real-time identity verification that allows our system to be usable in the disconnected environment, RISKCOG can protect user anywhere and anytime. As deploying RISKCOG in the production environment on a large scale, we resolve several new issues, such as the imbalanced data set and training set without ground truth with our stratified sampling method and a semi-supervised online learning method. We achieve the classification accuracy values 93.77% and 95.57% among the 1,513 users for the steady state and the moving state.

References

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

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

[3] Android service. https://developer.android.com/guide/comp

[4] Android UsageStatsManager. https://developer.android.com/reference/android/app/u

[5] AndroidLibSvm. https://github.com/yctung/AndroidLibSvm

[6] Android's biggest problem is operating system fragmentation. http://o.canada.com/technology/personal-tech/android-biggest-problem-is-operating-system-segmentat

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

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

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

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

[11] 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-ser

[12] Heikki J Ailisto, Mikko Lindholm, Jani Mantyjarvi, Elena Vildjiounaite, and Satu-Marja Makela. Identifying people from gait pattern with accelerometers. In *Defense and Security*, pages 7–14. International Society for Optics and Photonics, 2005.

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

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

[15] Tomas Brezmes, Juan-Luis Gorricho, and Josep 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.

[16] Hung-Chi Chu, Sheng-Chih Huang, and Jiun-Jiam Liaw. An acceleration feature-based gesture recognition system. In *SMC*, pages 3807–3812, 2013.

[17] Doruk Coskun, Ozlem Durmaz Incel, and Atay Ozgovde. Phone position/placement detection using accelerometer: Impact on activity recognition. In *ISSNIP*, pages 1–6, 2015.

[18] Anupam Das, Nikita Borisov, and Matthew 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.

[19] Mohammad O Derawi, Claudia Nickel, Patrick Bours, and Christoph Busch. Unobtrusive user-authentication on mobile phones using biometric gait recognition. In *IIH-MSP*, pages 306–311, 2010.

[20] Sanorita Dey, Nirupam Roy, Wenyuan Xu, Romit Roy Choudhury, and Srihari Nelakuditi. Accelprint: Imperfections of accelerometers make smartphones trackable. In *NDSS*. Citeseer, 2014.

[21] Adrienne Porter Felt, Serge Egelman, and David Wagner. I've got 99 problems, but vibration ain't one: A survey of smartphone users' concerns. In *ACM SPSM*, 2012.

[22] Dinei Florencio and Cormac 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.

[23] Jordan Frank, Shie Mannor, and Doina Precup. Activity and gait recognition with time-delay embeddings. In *AAAI*. Citeseer, 2010.

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

[25] Chiung Ching Ho, Chikkannan Eswaran, Kok-Why Ng, and June-Yee 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.

[26] Baik Hoh, Marco Gruteser, Hui Xiong, and Ansaf Alrabady. Enhancing security and privacy in traffic-monitoring systems. *IEEE Pervasive Computing*, 5(4):38–46, 2006.

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

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

[29] Jennifer R Kwapisz, Gary M Weiss, and Samuel A Moore. Cell phone-based biometric identification. In *BTAS*, pages 1–7, 2010.

[30] Xi Long, Bin Yin, and Ronald M Aarts. Single-accelerometer-based daily physical activity classification. In *EMBC*, pages 6107–6110, 2009.

[31] Hong Lu, Jonathan Huang, Tanwistha Saha, and Lama Nachman. Unobtrusive gait verification for mobile phones. In *Proceedings of the 2014 ACM international symposium on wearable computers*, pages 91–98. ACM, 2014.

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

[33] Yanzhi Ren, Yingying Chen, Mooi Choo Chuah, and Jie Yang. User verification leveraging gait recognition for smartphone enabled mobile healthcare systems. *Mobile Computing, IEEE Transactions on*, 14(9):1961–1974, 2015.

[34] Masaki Sekine, Toshiyo Tamura, Metin Akay, Toshiro Fujimoto, Tatsuo Togawa, and Yasuhiro Fukui. Discrimination of walking patterns using wavelet-based fractal analysis. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 10(3):188–196, 2002.

[35] Sebastijan Sprager and Damjan 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.

[36] Vladimir Svetnik, Andy Liaw, Christopher Tong, J Christopher Culberson, Robert P Sheridan, and Bradley 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.

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

[38] Esteban Vazquez-Fernandez and Daniel Gonzalez-Jimenez. Face recognition for authentication on mobile devices. *Image and Vision Computing*, 2016.

[39] Ning Wang, Eliathamby Ambikairajah, Branko G Celler, and Nigel H Lovell. Accelerometry based classification of gait patterns using empirical mode decomposition. In *ICASSP*, pages 617–620, 2008.

[40] Jason Weston, Sayan Mukherjee, Olivier Chapelle, Massimiliano Pontil, Tomaso Poggio, and Vladimir Vapnik. Feature selection for svms. 2000.