RiskCog: Unobtrusive Real-Time User Authentication on Mobile Devices in the Wild

Tiantian Zhu, Zhengyang Qu, Haitao Xu, Jingsi Zhang, Zhengyue Shao, Yan Chen, *Fellow, IEEE*, Sandeep Prabhakar, and Jianfeng Yang

Abstract—Recent hardware advances have led to the development and consumerization of mobile devices, which mainly include smartphones and various wearable devices. To protect the privacy of users, various user authentication mechanisms have been proposed. In particular, biometrics has been widely used for multi-factor authentication. However, biometrics-based authentication mechanisms usually require costly sensors deployed on devices, and rely on explicit user input and Internet connection for performing user authentication. In this article, we propose a system, called RiskCog, which can authenticate the ownership of mobile devices unobtrusively and in a real-time manner by adopting a learning-based approach. Unlike previous studies on user authentication, for cross-platform deployment, maximum user privacy protection, and unobtrusive authentication, RiskCog only relies on those widely available and privacy-insensitive motion sensors to capture the data related to the users' daily device usage. It requires no users' explicit input and has no requirement on the users' motion state or the device placement. RiskCog is also usable in the environment without Internet access by performing offline user identity verification. We conduct comprehensive experiments on smartphones and smartwatches, which show that RiskCog can authenticate device users rapidly and with high accuracy.

Index Terms—User authentication, implicit authentication, mobile device, user privacy

1 Introduction

As one important application of mobile devices, mobile payments are gaining acceptance among consumers, and the popularity is expected to boom within the next couple of years. A recent global survey by the United Nations Conference on Trade and Development (UNCTD) forecasts that up to 50 percent of consumers in major markets will be using a combination of smartphones and wearables to make payments by 2019 [1]. Wearable payment has come to reality. In the city of Hangzhou in China, where the recent G20 Summit was held, people used their smartwatches to pay for public transportation.

Although mobile payments have benefited users immensely, they also introduce several security risks, which can be classified into two categories: *data-driven* and *human-driven*. Data-driven risks are usually caused by leaking sensitive credit card data on smartphones through attack vectors including client side (mobile malware) [2], network channel (man-in-the-middle attack) [3], as well as server side [4]. Human-driven risks, on the other hand, arise when parties other than the owner have access to the mobile device and utilize the payment functions for their own benefit. According to the report by LexisNexis [5], among 16 percent of merchants accepting mobile payments in 2016, mobile transactions accounted for 26 percent of their total transaction volume, and up to 33 percent of those transaction volume was contributed by fraudulent transactions.

There exist lots of research work [18], [19], [20], [21] on solving data-driven risks. However, human-driven risks still lack effective countermeasures. Typical traditional user authentication mechanisms only verify if a user knows the account credential already created at the start of using the service. Moreover, most of those authentication systems are login credential-based, which require explicit user actions, e.g., account/password input. Previous studies show that explicit PIN/pattern passcodes can be stolen by touchscreen smudges [22], shoulder attack [23] and sensor-based inferring [24]. Later on, learning-based user identification approaches were proposed [25]. They construct a model to describe the usage pattern of the authorized phone owner, such as the locations he/she frequently visits [26], [27], keystroke dynamics [28], [29], finger movements [30], voice and face snapshots [31], [32]. Compared with login credential-based mechanisms, learning-based user identification mechanisms utilize a diverse set of features from various sensors to verify user's identity, and are much harder to get bypassed. Moreover, the learning-based mechanisms can be applied to the scenario of mobile payment. Suppose Alice and Bob are colleagues. Alice leaves her phone at the desk without turning off the screen. Thus it is possible for Bob to check Alice's Facebook private activities without her consent, if the Facebook app's automatic login option is enabled. In this case,

- *T. Zhu and Z. Shao are with the College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China. E-mail: {ttzhu, szylover}@zju.edu.cn.*
- *Z. Qu, J. Zhang, Y. Chen, and S. Prabhakar are with the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL 60208. E-mail: {zhengyangqu2017, JingsiZhang2016, SandeepPrabhakar2015}@u.northwestern.edu, ychen@northwestern.edu.*
- *H. Xu is with the School of Mathematical and Natural Sciences, Arizona State University, Glendale, AZ 85306. E-mail: hxu@asu.edu.*
- *J. Yang is with the College of Electronic Information, Wuhan University, Wuhan 430072, China. E-mail: yjf.whu@gmail.com.*

Manuscript received 27 May 2018; revised 1 Dec. 2018; accepted 7 Jan. 2018.
Date of publication 14 Jan. 2019; date of current version 6 Jan. 2020.
(Corresponding author: Tiantian Zhu.)
Digital Object Identifier no. 10.1109/TMC.2019.2892440

TABLE 1
Comparison with Related Studies on Smartphone Sensor-Based Authentication

Study	Require user movement	Fixed/dynamic device placement	Scale (#Users)	Require label	Offline real-time verification (Latency)	Accuracy
RiskCog	No	Dynamic	$>1,500$	No	3237.7 ms	≈95%
Derawi et al. (2010) [6]	Yes	Fixed	<100	Yes	NA	EER = 20%
Kwapisz et al. (2010) [7]	Yes	Fixed	<50	Yes	NA	≈90%
Ho et al. (2012) [8]	Yes	Fixed	<50	Yes	NA	-
Zhu et al. (2013) [9]	No	Fixed/Dynamic	<50	Yes	24 hours	<80%
Lu et al. (2014) [10]	Yes	Dynamic	<150	No	27864 ms	EER = 14%
Kayacik et al. (2014) [11]	Yes	Fixed	<10	Yes	122000 ms	53% - 99%
Ren et al. (2015) [12]	Yes	Fixed	<50	Yes	NA	≈90%
Lee et al. (2015) [13]	Yes	Fixed	<10	Yes	20000 ms	≈90%
Sitová et al. (2016) [14]	Yes	Fixed	<100	Yes	20000 ms	EER = 7%
Lee et al. (2017) [15]	No	Fixed	<50	Yes	4000 ms	73% - 96%
Buriro et al. (2017) [16]	Yes	Dynamic	<50	Yes	NA	EER = 4%
Shen et al. (2018) [17]	Yes	Fixed	<150	Yes	8000 ms	EER = 5%

learning-based authentication approaches which are running in the background can detect the unauthorized user access implicitly, and then invoke the follow-up self-defense actions, such as privately alerting the phone owner of the suspicious access by email, rendering an empty page or demanding for retyping the password of Alice's Facebook.

In this article, we aim to develop a biometrics-based user authentication system, called RiskCog, at the device level that requires no developer support. This system allows the detection results to be reused and removes the redundancy in terms of the data collected from each individual app. In our *threat model*, each smartphone has a unique owner, and attackers attempt to access the phone illegally. Our system assumes that only the several widely used motion sensors are available on the device, which will be detailed in Section 3.2. In Table 1, we list the problems with previous learning-based approaches and summarize the following challenges:

(1) Lack of Features. Although a mobile device supports numerous sensors, which can be potentially used to fingerprint users, the fragmentation issue [33] hinders it from being deployed widely. Many sophisticated sensors are not available on some low-end devices. Moreover, some sensors have to be integrated into an app to work, e.g., the pressure sensor, which needs to be bound to a view element within an app. A device level protection requires not to have dependency on those sensors. Additionally, any feature involving user's private information is also not suitable for the sake of privacy.

(2) Data Availability & Dynamic Device Placement. Only those data collected during daily usage is usable because fingerprinting a user fundamentally depends on the user's specific pattern of handling the phone. Some smartphone users might use the phone rarely, or prefer to keep the phone on a stationary plane. In this case, when we try to collect training data, there will be enough motion events to be used for representing the authorized owner.

Previous studies [6], [7], [8], [11], [12], [13] have a strong assumption that the smart device placement should be fixed, e.g., in the trouser pocket. Sitová et al. [14], Buriro et al. [16] and Shen et al. [17] combine motion sensors and touchscreen for active smartphone authentication, but the getting the data from touchscreen needs additional permissions on Android. Zhu et al. [9] proposed a similar method to ours, which aims to construct the gesture model of how a user uses the device. The difference is that their work requires the test phase of dynamic device placement to last 24 hours, which makes the real-time detection unpractical. Also, when the body placement (and orientation) of the sensor is not fixed, the accuracy of their system will greatly degrade. Buriro et al. [16] tried to authenticate different users based on the micro-movements after an unlock event occurs on the smartphone. However, the authors failed to measure the latency on the server side and also did not propose an offline verification model in their work. Research [15] authenticates different users utilizing users' phone pick-up movements, but it only considers the pick-up signals starting from a stable state. To offer a full verification, we should not have any requirements for the device placement or user's motion state. Moreover, the app-specific pattern will challenge the classification accuracy. The user's pattern dramatically varies with different types of apps, e.g., the frequent typing gestures in a chatting app versus the rotation in a race car game.

(3) Imbalanced Dataset. When identifying the authorized device owner, we label the data of the authorized user as 1 and that of other users as 0. The binary classification task is imbalanced, since the number of positive examples is much less than that of negative examples. The imbalance ratio in our work increases with the scale of users.

(4) Unlabeled Data. The proposed prototype systems [6], [7], [8], [11], [12], [13] introduce supervised learning algorithms for the well-labeled training set. For example, the data in the training set is well labeled on whether each data sample is generated when the authorized user is using the device. In [15], Lee et al. feed the labeled data to a weighted multi-dimensional DTW algorithm. However, the well-labeled data is not always available in the practical scenarios. One possible case is that the device owner may give the mobile device to her/his family member during data collection.

(5) Constrained Mobile Environment. The remote server may become unaccessible in the disconnected/weakly connected environment, which renders the client-server model infeasible, given our aim of performing user authentication in real-time. In addition, a complicated classification model with high prediction accuracy requires heavy computation resource, and thus it is not applicable to the constrained

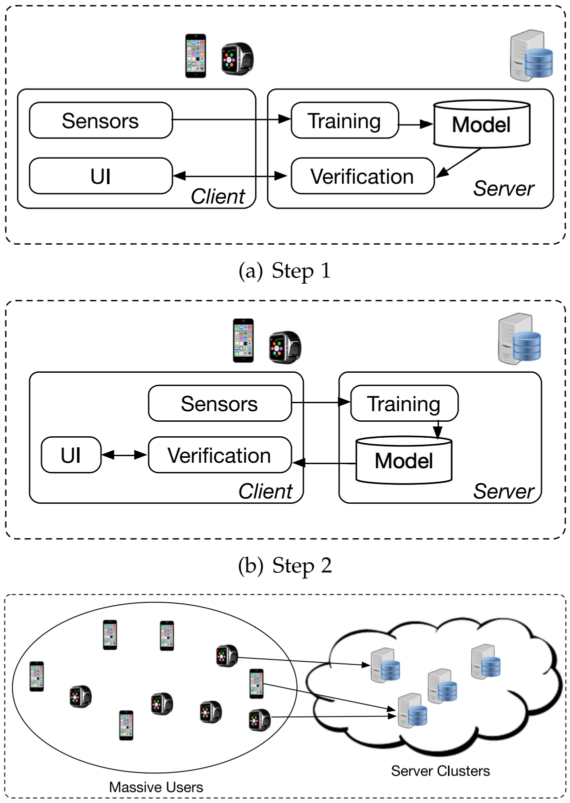


Fig. 1. Three deployment modes of our system RiskCog.

mobile devices. Offline verification seems promising in the case of disconnected Internet access. However, existing works have their limitations. The complex gait analysis proposed in [10] and [11] has high latency; and the work [9] cannot support real-time authentication well.

We design a system, called RiskCog, to provide the offline unobtrusive user identity verification service. It is based on our semi-supervised learning algorithm to identify the device owner. Each data sample is collected, preprocessed, and finally aggregated into users' implicit events (i.e., steady state and moving state when using the smartphone, raising hands to check the time on the watch). Multiple parallel classifiers are trained for these events, which are used to predict whether the authorized device owner is using the device. Our system can be deployed in three modes based on the placement of our verification module (i.e., whether the module performing user authentication is deployed on the client side or on the server side), shown in Fig. 1. In the first mode of our user authentication framework, both the training and verification modules are deployed on the server side. In the second one, the verification module is migrated to the local client to deal with the disconnected environment. In the third mode, the verification module also resides on the local client while we have a different goal, which is to scale the system towards massive end users. In summary, we made the following contributions:

- We develop an unobtrusive user authentication system, which authenticates mobile device users in real-time, simply based on users' natural usage of mobile devices, and without any kinds of user interaction. Our system has achieved a good balance among security, privacy and usability.
- We compute and finally select 56 features to generally identify the mobile device owner, by only involving motion sensors. Moreover, the feature set does not involve any in-app invocation or user privacy tracking. It is independent of user's motion state and device placement.
- We design a data collection mechanism to resolve the data availability issue with a 99.9 percent coverage rate of users' patterns, which cognitively recognizes device usage events and completely captures all the data helpful in fingerprinting usage pattern. The data that is generated when the device is put on a plane or that has the dependency on the app-specific pattern is not adopted and filtered out. We implement a preprocessing procedure that guarantees the usability of our system with dynamic device placement.
- We develop a stratified sampling method to resolve the issue of imbalanced dataset in learning-based user identification. Our sampling method maintains the temporal continuity of sensor reading and improves the representativeness of the negative samples.
- We design a semi-supervised online learning algorithm, where the classifier is trained incrementally with the data collected in chunks. It eliminates the high time latency when handling unlabeled data with unsupervised methods. By checking the consistency among the data sent to server incrementally, we can filter out the part that is not coherent with the authorized owner's fingerprint.
- We develop an unobtrusive user identification mechanism with cross-platform capability. We decouple the verification from the server side and resolve the issue of availability. Our optimization produces a light-weighted identity verification service on resource-constrained mobile devices, including smartphone and smartwatch. It only takes 3237.7 ms to finish the verification on smartphone.

We achieve high accuracy for unobtrusive user identification with wild data from a commercialized mobile payment app developed by one Internet giant company in the world. RiskCog achieves the classification accuracy values of 93.77 and 95.57 percent for the steady/moving states on smartphone, respectively. RiskCog can also effectively defend against brute-force attacks and mimicry attacks. In practice, RiskCog could be used as a third-party service to perform implicit authentication first. If it fails, other explicit authentications or effective countermeasures would be then leveraged. RiskCog can be utilized as an implicit authentication system on any privacy-sensitive applications with a 99.9 percent coverage rate of users' patterns and can reduce the number of times a user has to do explicit authentication by at least 73.2 percent. We release two Android apps[1] and an iOS app[2] which implement our user identification mechanism.

1. RiskCog Android app.
http://list.zju.edu.cn/SensorDemo-release.apk,
http://list.zju.edu.cn/app-authentication.apk.
2. RiskCog iOS app.
http://list.zju.edu.cn/riskcog-ios.zip.

TABLE 2
Availability of Acceleration Sensor (AC), Gyroscope Sensor (GY), Gravity Sensor (GR), and MS for Marketshare

Brand	MS in Q4, 2017	Device	AC	GY	GR
Apple	18.6%	iPhone X	✓	✓	✓
		iPhone 8(s)	✓	✓	✓
		iPhone 7(s)	✓	✓	✓
		Apple Watch	✓	✓	✓
Samsung	17.9%	Galaxy S8	✓	✓	✓
		Galaxy S7	✓	✓	✓
		Galaxy Note 2(3, 4)	✓	✓	✓
		Galaxy CORE Prime	✓	×	×
Huawei	9.9%	P10	✓	✓	✓
OPPO	7.7%	R11	✓	✓	✓
Xiaomi	7.5%	MI 3(4, 5)	✓	✓	✓
vivo	5.9%	Xplay6	✓	✓	✓
LG	3.3%	G5	✓	✓	✓
		Gear Watch	✓	✓	✓

The remainder of this article is organized as follows. Section 2 presents brief background knowledge. In Section 3, we cover RiskCog design in detail, which is then followed by an illustration of the implementation procedures in Section 4. Section 5 presents the overall evaluation of our system. Section 6 discusses the possibility of an attacker evading our authentication system. Section 7 surveys the relevant work. Section 8 concludes our work.

2 Background

2.1 Credential-Based Authentication and Learning-Based Authentication

Authentication is used to prevent the unauthorized parties from using sensitive services. Currently, the credential is the predominant form of an authentication system. It is known to have many security problems. It can only verify if a user knows the credential but cannot recognize whether she/he is the real owner of the device. The credential-based authentication is also vulnerable to dictionary attacks. A recent report about data breaches [34] shows that 4.1 percent of users choose "123456" as their passwords, and 79.9 percent of apps still accept weak (lower-case only letters) passwords. Florencio et al. [35] even found that a single password is typically used to access over five sites. Moreover, the credential-based authentication cannot enforce full protection and achieve usability at the same time. A fully on-demand verification requires a user's explicit input action every time the user tries to access the sensitive services. While, if the user enables the automatic login, out of the concern of usability, the identity will be verified only once.

Considering the security issues of the traditional login credential-based authentication, learning-based approaches are introduced for user identification. Such approaches are able to exactly profile the authorized device owner by a diverse set of features. At the time of verification, the system will predict the probability that the user who attempts to access the service is the owner of the device by checking the collected test samples against the trained model or the stored template. It is much harder for attackers to bypass the verification compared with the traditional authentication mechanisms, given the difficulty of mimicking a legitimate user's patterns. Moreover, it requires no explicit actions from the user, which can enforce the on-demand protection without sacrificing the usability.

2.2 User Privacy Preserving

The privacy preserving property of RiskCog is defined in the context of social impact. Previous studies have already verified the feasibility of fingerprinting mobile devices with motion sensors [36], [37]. However, device tracking does not imply the identity of its owner in social life. Our system can only provide the knowledge about the mapping between a trained model and an authorized device owner. However, it is unable to further figure out who the device owner is. Compared 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], [39]. As for geo-location, it is able to identify the owner in the physical world as stated in previous studies [27], [40].

2.3 Platform Porting & Sensor Availability

Our user identification mechanism is built based on the data collected from three motion sensors, including the acceleration sensor, gyroscope sensor, and gravity sensor. We studied the prevalence of those three sensors on 20 types of popular smart mobile devices, which are from 7 major mobile device vendors and have high market penetration. Table 2 provides the detailed results. It shows that the three motion sensors are available on all those smart devices, except ont device type, i.e., Samsung Galaxy CORE Prime, released in 2014. Thus, RiskCog can be easily migrated to various mobile platforms.

3 System Design

In this section, we first introduce the overall architecture of our proposed user authentication mechanism for mobile devices. We then discuss several important topics in its design, including sensor selection, data collection, data preprocessing, feature selection, semi-supervised online learning and user authentication.

3.1 System Overview

The architecture of our system is illustrated in Fig. 2. This framework comprises a platform of wearable devices, smartphones and online servers. To authenticate a user, selected sensors, which are embedded in wearable devices and smartphones, sense and collect user behavior related data periodically. Smartphones connect to cloud servers via WiFi or cellular links, and send the collected sensory data to the latter. Most of wearable devices currently cannot directly connect to Internet but are expected to have the Internet access capability very soon in the near future [41]. Now they need to first connect to a network proxy (e.g., a smartphone) via Bluetooth, which receives the sensory data and uploads it to cloud servers. The servers in the cloud perform a series of computing tasks including raw signal data preprocessing, feature extraction, and finally training classifier models and applying them to make authentication

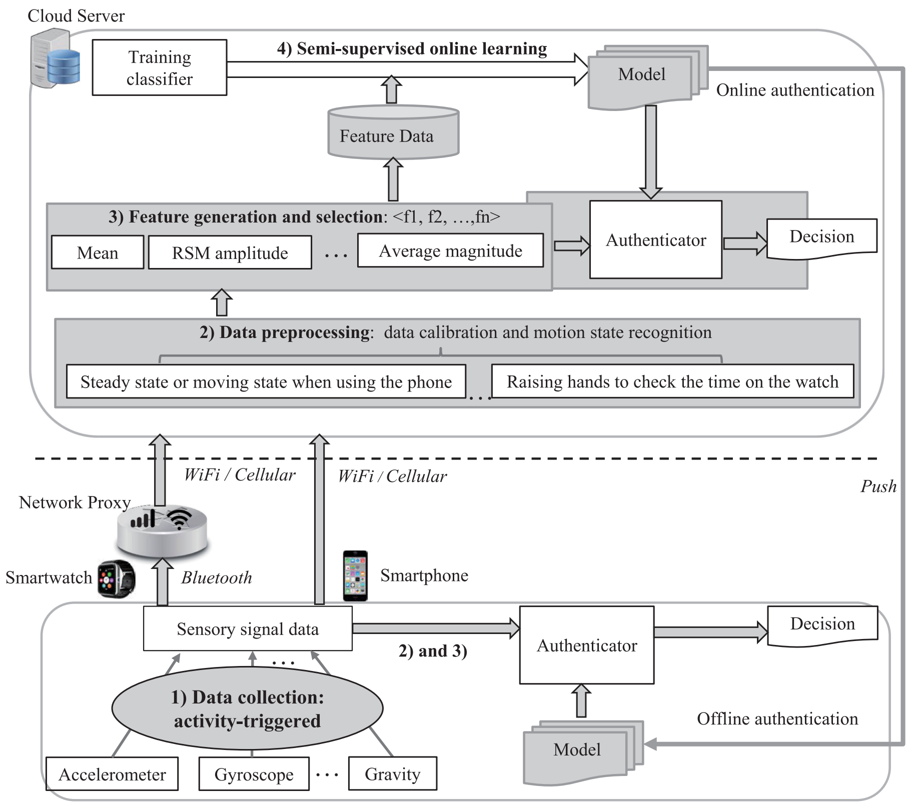


Fig. 2. System architecture; training phase starts at data collection from motion sensors and ends when the classification model is pushed to the device followed by the local identify verification.

decision. Moreover, the data preprocessing module and feature extraction module are located in both cloud server and smartphone. Note that the classifier models are periodically updated with new sensory data received from endpoint devices. Our system will train two separated authentication models from both smartphone and smartwatch. The generated model will be pushed to the client. In particular, the model of smartwatch will be stored in the paired smartphone for further authentication. In this architecture, wearable and smartphone devices are only responsible for simple data collection and calculation; the computation-intensive tasks such as semi-supervised online learning are offloaded to cloud servers to conserve the on-device battery energy and computing power. Considering the main idea of authenticating the owners of smartwatches and smartphones in our system is fundamentally the same, we will focus on smartphones to tell how our system works in the remaining sections. Next, we describe the key components in the architecture in detail.

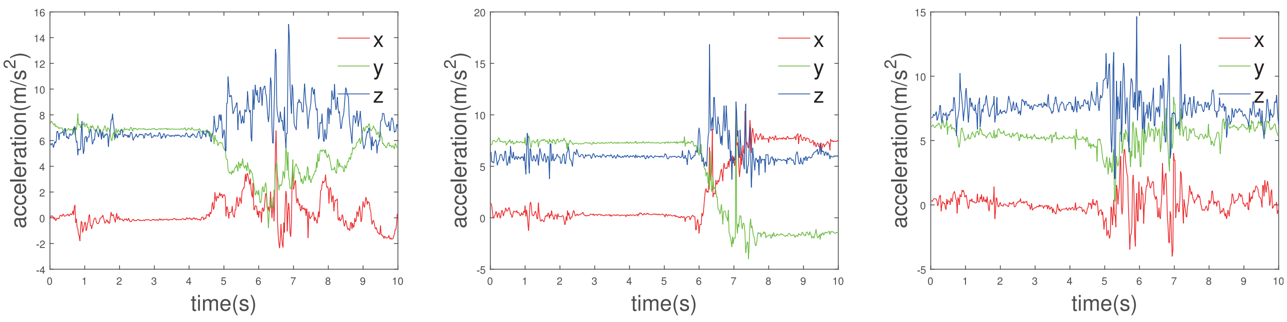
3.2 Sensor Selection

Wearable and smartphone devices are equipped with various sensors to capture kinds of information to enable ample smart functions. Since there are a mass of sensors embedded in mobile devices, we only select the sensors which satisfy the following three characteristics.

First, the sensors should be widely available. Two devices could differ in the number and type of embedded sensors no matter whether they belong to the same category, such as smart watches and smart wristbands. To design an authentication system generally used for wearable and smartphone devices, first we select the sensors widely available. Inertial sensors, such as accelerometer and gyroscope, are becoming essential parts of various mobile and wearable platforms. They have high sampling rate capability with low cost to capture users' activity [42], and are thus selected. Second, sensors need to be privacy-insensitive since the data collected by them does not contain personally identifiable information (PII). Sensors including camera, microphone, and GPS are also available or about to come popular in many wearable devices, but we do not consider such sensors in our design since they could capture users' face, voice and locations visited, which are typically considered quite privacy-sensitive. In addition, we will not consider touch screen because it can not only capture users' fingerprints but also record the contextual information about users' touch behavior on the devices. Finally, the characteristic of being environment-insensitive is urgently needed. The desired authentication system is expected to work across all environments. Thus the sensors with their performance easily affected by environment factors are not considered either. Examples of such sensors are voice sensors in the noise background, cameras in the row lighting condition, and capacitive sensors in the humidity weather. Therefore, our system RiskCog collects data from the *acceleration sensor*, *gyroscope sensor* and *gravity sensor* on smart devices. Acceleration sensor and gravity sensor are used to remove the noise data and recognize different motion states, while acceleration sensor and gyroscope sensor are used for our authentication.

3.3 Data Collection and Preprocssing

Each sensor reading includes values corresponding to the x, y, and z axes



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

Fig. 3. Acceleration sensor values in the temporal dimension. The sensor readings largely vary with different types of apps even for the same user, but they are relatively consistent during the start of an app.

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

Here, the parameter k represents the kth sensor reading in the time dimension. Take Android as an example. Android provides a class `SensorManager` [43], which allows developers to refresh the sensor data in both fixed and customized intervals/delays after registering the sensors with `registerlistener()`. There are four delays:

- $SENSOR\_DELAY\_FASTEST = 0s$.
- $SENSOR\_DELAY\_GAME = 0.02s(50\ HZ)$.
- $SENSOR\_DELAY\_NORMAL = 0.06s$.
- $SENSOR\_DELAY\_UI = 0.2s$.

Since both $SENSOR\_DELAY\_FASTEST$ and $SENSOR\_DELAY\_GAME$ can provide sufficient data within a short time. Consider the battery consumption, we choose $SENSOR\_DELAY\_GAME$ for our data collection. Different phones have different factory settings for their internal sensors, which thus generate quite different readings for the same gesture. We asked the individuals to put the device on a stationary plane to calibrate motion sensors before they ran RISKCOG. Note that the sensor readings may drift in a long period. On one hand, the sensor drift depends on the type of hardware (i.e., MEMS). In our experiment, we assume the other factors, such as the usage period and temperature, have the minimal influence on data collection. On the other hand, if some false negatives (i.e., the authorized owner is incorrectly identified as other users) occurred frequently even after a long period (about 6 months according to our experience), the user would be suggested to calibrate the error once again.

We develop a mobile application for the purpose of data collection. The app needs to detect the duration when the device is being actively used, because only the sensor readings during such a duration are effective to represent user's manner. With our experiment results shown in Fig. 3, we observe that the sensor readings largely vary with different types of apps even for the same user, which will affect the classification accuracy of the trained model. For example, the following three kinds of apps can demonstrate that the same user adopt different user gestures when playing different apps. A user rotates the phone when playing a race car game, which is mainly driven by the acceleration sensor only; a chatting app will generate a lot of typing gestures; comparatively, a news app usually generates balanced sensor readings. However, we also observe that the sensor readings are relatively consistent during the start of an app, i.e., the loading phase. After a large amount of observations, we found the duration of the loading phase lasted for 2 to 4 seconds. The choice of the duration is based on sensitivity analysis. We found that when the time was smaller than 3 seconds, the final accuracy increases as the time increases. While when the time was larger than 3 seconds, the final accuracy decreases as the time increases. Finally we chose 3 seconds as our duration for data collection.

Take Android as an example, we have a `BroadcastReceiver` to capture the system event where the screen of a device is turned on, and then it starts a `Service` [44] that periodically queries the current app in the foreground. When the currently active app is different from the one in the last query, we will recognize that a new app has been 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.

So, the data is only collected during the active daily usage. The application-specific pattern is filtered, since our goal is to develop a device-level authentication system, rather than an app-specific one. The sensory data from other sensors except the three motion sensors is also filtered.

Our preprocess includes the data calibration and motion state recognition. A user may not actually hold the phone during daily usage. We did 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 asked 20 participants to handle a phone and put the 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

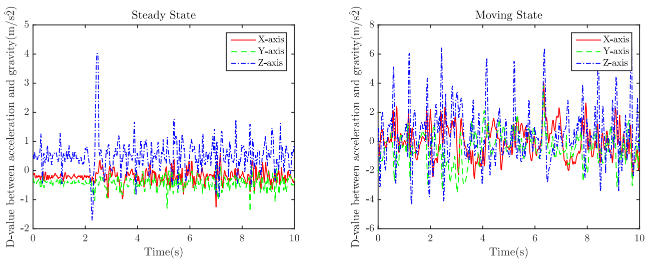


Fig. 4. D-values in the temporal dimension. In the moving state, the D-value has a higher amplitude compared to that in the steady state.

readings to our global coordinate system, which allows RiskCog to be insensitive to device orientation. We identify the gravity direction based on the values from the gravity sensor, whose opposite direction will be set as the z-axis in the global coordinate system. It is thus straightforward to decide the directions of x-axis and y-axis by Fleming's right-hand rule [45].

The sensor reading in the moving state largely differs from that in the steady state. In this paper, we define that the steady state means the smartphone is used in sitting, standing and some other stable scenarios, while the moving state represents that one uses the smartphone in walking, riding and other changing scenarios. If we use one classifier for all the motion states, there will be a huge inconsistency within the data samples of one user, which will further affect the classification accuracy. Therefore, a classifier is trained for each state of a user.

We observe that the difference between the values of acceleration sensor and those of gravity sensor, termed as *D-value*, in the moving state, has a higher amplitude compared to the *D-value* in the steady state. We define the kth D-values on three dimensions as

$$\begin{aligned} X_d(k) &= |X_a(k) - X_{gr}(k)|, \\ Y_d(k) &= |Y_a(k) - Y_{gr}(k)|, \\ Z_d(k) &= |Z_a(k) - Z_{gr}(k)|. \end{aligned}$$

We calculate the median values $X_d\tilde{(k)}$, $Y_d\tilde{(k)}$, and $Z_d\tilde{(k)}$ on three dimensions in the data collection duration. With a predefined threshold, the user's motion state is determined as steady given the condition

$$\sqrt{X_d\tilde{(k)}^2 + Y_d\tilde{(k)}^2 + Z_d\tilde{(k)}^2} < \delta.$$

Actually, *D-value* reflects the degree of the movement of the body. As shown in Fig. 4, if a smartphone is hold in a steady state (such as sitting and standing), the *D-value* would be small (around to 0) because the reading of acceleration sensor and gravity sensor would be the same. But in a moving state (such as walking and riding), the absolute *D-value* would be large because the movement of the body is intense. Especially, if the smartphone moves in a uniform speed (such as sitting in a car that moves at a constant speed, relatively static), it still can be considered as a steady state because there is no explicit movement of the body. Our system can recognize different states automatically and generate the corresponding classifiers. We found the value of δ ranging from 0 to 0.4 in steady state and ranging from 0.6 to more in moving state. We set the value of δ as 0.5 based on our experiment results.

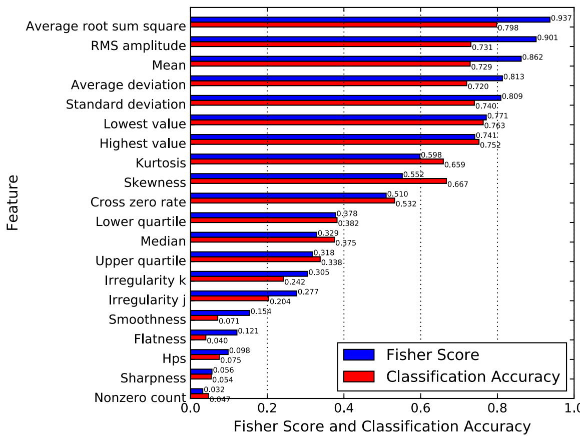


Fig. 5. The correlation between the Fisher Score and the feature's impact on the RiskCog classification average accuracy for all the features.

3.4 Feature Generation and Selection

For the classification, we only utilize the 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. We denote the ith value of the feature vector as $\mathcal{F}_i = \{F_{1i}, F_{2i}, \ldots, F_{pi}\}$, which includes p features. In order to maintain the consistency among feature vectors in the temporal domain, we utilize sliding window design with 50 percent overlapping between each pair of neighbor segments, i.e.,

$$\begin{aligned} &\{X_a(k), Y_a(k), Z_a(k), X_{gy}(k), Y_{gy}(k), Z_{gy}(k)\}_{k=1}^{10} \Rightarrow \mathcal{F}_1. \\ &\{X_a(k), Y_a(k), Z_a(k), X_{gy}(k), Y_{gy}(k), Z_{gy}(k)\}_{k=6}^{15} \Rightarrow \mathcal{F}_2. \\ &\ldots \end{aligned}$$

In 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 used a public feature extraction library LibXtract [46] to extract 20 common features, which cover multiple moments and other commonly used statistical properties of the distribution. To select the effective features, we computed the Fisher Score [47] of the extracted features to further evaluate the discriminative power given the ground truth in the laboratory settings (see details in Table 5, *dataset I*). The correlation between the Fisher Score and the feature's impact on the RiskCog classification accuracy (the average value on steady state and moving state) for all the features is shown as Fig. 5. If the corresponding feature has no discriminative power among users, then the Fisher Score will be close to zero and the classification average accuracy of the feature will be low. On the contrary, selected features are considered as good ones if their Fisher Scores are much larger than zero, and also classification with these features can achieve high accuracy. Consequently, we select the top 10 features according to the Fisher Score, which are listed in Table 3.

TABLE 3
Top 10 Features, Selected by Fisher Score

No.	Feature	Formula	Definition	Fisher Score
1	Average root sum square	$\sum_{k=1}^{K} \sqrt{x^2(k)+y^2(k)+z^2(k)}/K.$	Average magnitude of data segment	0.937
2	RMS amplitude	$\sqrt{\sum_{k=1}^{K}[x(k)]^2/K}.$	Mean of all amplitudes of data segment	0.901
3	Mean	$\sum_{k=1}^{K} x(k)/K.$	Mean value of data segment	0.862
4	Average deviation	$\sum_{k=1}^{K} \lvert x(k)-\bar{x}\rvert/K.$	Average deviation of data segment	0.813
5	Standard deviation	$\sqrt{\sum_{k=1}^{K}[x(k)-\bar{x}]^2/(K-1)}.$	Mean of the deviations of data segment	0.809
6	Lowest value	$\min\{x(k), \quad k=1,\ldots,K\}.$	minimum value of data segment	0.771
7	Highest value	$\max\{x(k), \quad k=1,\ldots,K\}.$	maximum value of data segment	0.741
8	Kurtosis	$\sum_{k=1}^{K}[(x(k)-\bar{x})/\sigma]^4/K-3.$	Width of peak for data segment	0.598
9	Skewness	$\sum_{k=1}^{K}[(x(k)-\bar{x})/\sigma]^3/K.$	Orientation of peak for data segment	0.552
10	Cross zero rate	$\sum_{k=0}^{K-1} \lvert\lvert sgn[x(k+1)]-sgn[x(k)]\rvert\rvert/K.$	The rate of sign-changes for data segment	0.510

Here, K equals 10 for our case. Replace x in the last 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 first formula provides us with 2 features from two 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 sensory data collected from one sensor is unique and different from the distribution of another sensor. We conduct our experiment involving 1,513 users to verify the assumption. 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 4, 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.

3.5 Semi-Supervised Online Learning Algorithm

Training Set. In the training phase, data is collected to generate the feature vectors. Each user is then profiled by n feature vectors, denoted by $\mathcal{F}_i, i=1,\ldots,n$. For p phone users, $n \times p$ vectors are used to train the classifier in total. The sample size refers to the number of feature vectors n. Treat the dataset of the authorized user as Class 1 and that of other users as Class 0. We then have a highly imbalanced dataset, since p is large.

TABLE 4
Cluster Separability, AVG for Average, and STD for Standard Deviation

	Steady	Moving
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

We employ the stratified sampling to handle the problem [48], which groups the vectors by one feature value. According to the feature selection ranking result, the average root sum square of acceleration (denoted as 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 are randomly drawn from each stratum. 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 have 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, i.e., a stronger capability of representing other users, but it also introduces higher latency.

One might argue that the number of positive samples is much larger than that of negative samples in the actual usage because the authorized device owner rather than other users interacts with the device for most of the time. However, a large number of negative samples are needed to allow the model to generally depict the usage manner of all users other than the authorized device owner. Otherwise, the model would be easily bypassed by brute-force attacks.

Our training set is constructed as in Fig. 6. In the figure, we can see all the samples of other users are selected averagely in each stratum with red color. We use Group1, Group2 and Group3 to represent different strata (actually we have 5 groups in the experiment as mentioned before). The ratio of the number of samples by the owner to that of other users is 1:5, which can be properly handled by normal learning algorithms. The optimal point is chosen by conducting experiment. For effective training, the number of positive instances usually needs to exceed 4,000. We carried out the experiment on 106 people (see details in Table 5)

TABLE 5
The Details of Our Datasets Where All Participants were Skilled Smartphone Users with at Least Two Years' Experience; GT for Groud Truth

Dataset	Participants	Age	Provider	Devices and Vendors	Duration	GT
Dataset I	20 individuals	20 - 60	Our laboratory	Samsung N9100 from Samsung Inc.	14 days	Yes
Dataset II	34 individuals	20 - 60	Internet company I	iPhone 7 from Apple Inc.	10 days	Yes
Dataset III	15 individuals	20 - 60	Our laboratory	Moto 360/iWatch from Motorola/Apple Inc.	14 days	Yes
Dataset IV	1513 individuals	20 - 60	Internet company II	Mi 3, Mi 4 and Redmi Note2 from Xiaomi Inc.	10 days	No
-	106 individuals	20 - 60	Internet company II	Redmi Note2, used for parameter optimization	7 days	No

based on the data collected within 5 days, and the sample size of positive instances ranges from 1,024 to 20,132 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 Fig. 8, which shows the strong relation to the sample size of positive instances. When the sample from authorized user is insufficient, less than 4,000, the performance of our classification is less satisfied with low accuracy. However, it gets improved drastically when the sample size increases. We also notice that once the sample size exceeds the threshold 4,000, the accuracy cannot be improved by simply adding more samples.

Classification Method. We choose Support Vector Machines (SVMs) with the 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) 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 Multi-Dimensional Space. Another related reason for choosing SVMs is that we have a comparatively multi-dimensional feature space with 56 features extracted from original observations. SVMs have been reported in many studies to work better with our situation [49], [50], for example, text classification. We also tried two-dimensional reduction methods [51] 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.

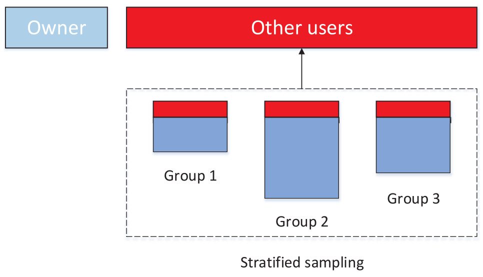


Fig. 6. Training set construction. All the samples of other users are selected on average in each stratum with red color. We use Group1, Group2, and Group3 to represent different strata.

(3) Dependent/Correlated Data. Our features are extracted from motion sensors and the readings on three dimensions are inevitably correlated, given the nature of human activity. SVM does not explicitly assume feature independence.

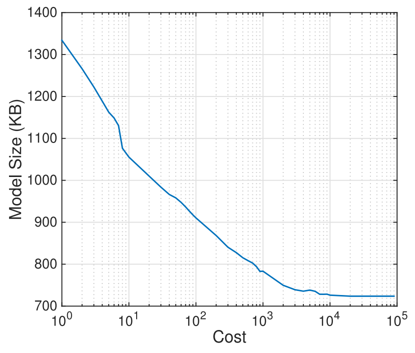
Optimization. The size of model is essential when verifying the user identity offline. Considering the limited computational resources of mobile devices, a smaller model indicates the lower CPU and memory consumption in identity verification and lower traffic when pushing the model to the smartphone. Moreover, the size of model is related to the number of support vectors in SVM learning algorithms. Thus, the phase of optimization also avoids overfitting. We conduct the grid search to find the optimal configuration of the parameters C and γ in the SVM with RBF kernel, where the parameter C trades off misclassification of training examples against simplicity of the decision surface, and the parameter γ defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close' [52]. We randomly choose 80 users, who generate most number of data samples in daily usage, from our dataset with 1,513 users that will be explained in detail in Section 5. Given the fixed convergence parameter $\epsilon = 0.01$, we change C from 1 to 90,000 and γ from 0 to 0.1. As shown in Fig. 7, 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 in logarithmic scale). The model size will reach the minimal value when γ equals to 0.01. Meanwhile, we will get a higher accuracy under a smaller model size.

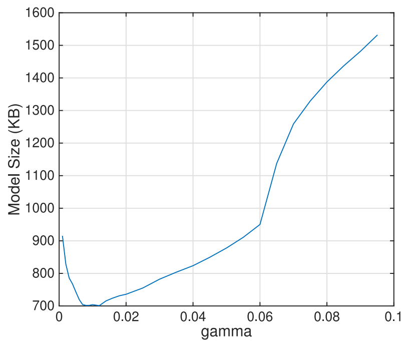
Our semi-supervised online learning algorithm is illustrated in Fig. 9. The data samples are uploaded to our server in chunks. One chunk is split into two parts for both training and testing purposes. The new training data and other users data is used to construct a training set. The online learning module takes the 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

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

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.

The authorized device owner may share her/his phone to others, such as friends and family members. We have no


(a) C (logarithmic scale, $\epsilon = 0.01$, $\gamma = 0.01$)


(b) γ ($\epsilon = 0.01$, $C = 100$)

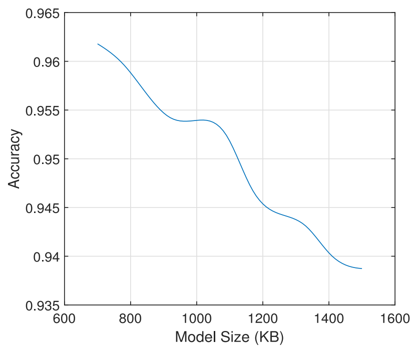

(c) Model accuracy

Fig. 7. Model size versus C, γ, and accuracy.

idea of the label ground truth. Incorporating the noisy data in the classification model would affect the prediction accuracy. 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. S_{otrain} is defined as the positive samples of device owner for training. We determine the model as ready with the condition

$$\alpha_{new} > A \text{and} Var(\alpha) < V \text{and} S_{otrain} > 4,000.$$

In our commit/rollback design, we have the classification accuracy for each chunk of data. When a) the latest prediction accuracy is higher than the threshold A, b) the variance of all the accuracy values of those chunks which are accepted previously is smaller than V, and c) the sample from authorized user is more than 4,000, we will determine the classifier training is finished. This implies that the performance converges to a stable state.

Decision. Given a feature vector $\mathcal{F}_i$, 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.

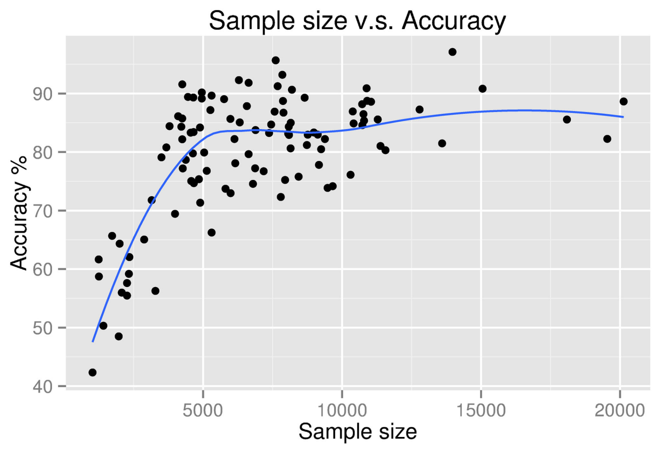


Fig. 8. Sample size requirement. Classification accuracy gets improved drastically when the sample size (0-4,000) increases.

4 Implementation

Our data collection scheme involves verifying the active device screen and the presence of applications in the foreground. We implement the `BroadcastReceiver` [53] to capture the system event where the device screen is turned on. Android provides two APIs: `getRunningAppProcesses()` and `getRunningTasks()`, to retrieve the current application running in the foreground. However, starting from Android 5.0, those APIs are deprecated. 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 ID of a running application (i.e., `pid`) to the application name on Android 5.0 and Android 6.0. Since Android 7.0, Android has locked down the permissions of `/proc`, we can't get the running process via `ps`. Instead, the list of running apps can be alternatively fetched by using `UsageStatsManager` [54] on Android 7.0 and 8.0. Our implementation allows us to intercept the active applications properly on all the existing versions of Android.

As for the wearable devices, Android wear API [55] and watchOS API [56] are used for Android and iOS, respectively. When interacting with server, Apache HttpClient and AFNetworking are integrated in our system, which help to do the network request operation. Due to some watch devices such as android watch and iWatch cannot communicate to the server directly, Android and iOS provide Google Play Service and WatchConnectivity package

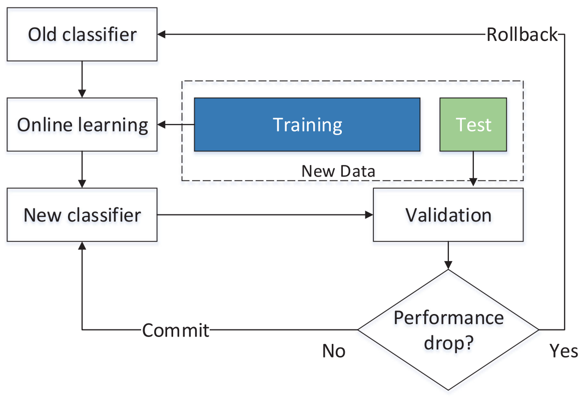


Fig. 9. Semi-supervised online learning; each chunk of data is committed if there is no classification accuracy drop and the training finishes when the classification accuracy values are stable across chunks.

respectively to get connection between watch and the corresponding phone. The server is implemented in `django` [57] and processes the data sent from client.

We implement the data preprocess module in C++. It is intended to filter the data which is ineffective to fingerprint user's pattern and distinguish the two motion states. We use `LibXtract` [46] to extract features. We train our model with `LibSvm` [58] on the server side and enforce the offline identity verification with `AndroidLibSvm` [59] on the Android platform and `LibSvm` on the iOS platform.

Overall, we implement RiskCog with over 7,000 lines of code in C++ for data processing and algorithm design, 2,000 lines of code in Python for server setup, 2,000 lines of code in Swift for iOS and 8,000 lines of code in Java for Android.

5 Evaluation

We define the ground truth as the situation that the device owner is using the smartphone. In total, we obtained four datasets for the evaluation purpose, the details of our datasets are shown in Table 5.

For the experimental data with ground truth, we solicited 20 participants to use the same phone for two weeks. Each participant generates 9240 effective samples by handling the phone in both steady and moving states (*dataset I*). 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 owner to that of other users is 1:5. The test set follows the same distribution. Moreover, we were able to obtain a labeled dataset provided by a big Internet company for the accuracy benchmarking test. That dataset was generated by 34 participants in the steady state from the iOS platform (iPhone 7) (*dataset II*). To verify the cross-platform portability of our system, we have 15 participants to wear Moto 360 and Apple Watch 3.0 for one week. Our own app records the related motion data every time the participants raise hands to check the time on the watch. For each participant, we get 6,000 effective samples (*dataset III*). Our fourth dataset is a large-scale raw dataset without ground truth, which was directly collected from the product by another Internet company with millions of users. For the ethical consideration, we include the purpose of data collection in the user agreement. All the participants were informed of this study and they were given the option to opt in or opt out. Finally we collect data from 1,513 different users for 10 days (*dataset IV*). For all the above datasets, the collection frequency is 50 Hz. 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. IMEI is used as the user identifier. Note that a portion of the data collected was filtered, which is ineffective to fingerprint the user (e.g., phone is put on a stationary plane). We define the coverage of users' patterns as the ratio of the number of effective samples to that of all samples. On (*dataset IV*), we get 283,133,354 sets of original data in total and 283,006,659 of them are effective and finally used. The coverage rate of users' patterns is 99.9 percent (i.e., 283,006,659/283,133,354), which indicates that most of the users' natural usage of mobile devices has been used by our system.

The distributions of training and test sets are identical to the experimental data. 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.
- False negative (FN). The authorized owner is incorrectly identified as other users.
- True negative (TN). Other users are correctly identified.
- Performance & Overhead. We evaluate the time latency of training and memory usage of each user 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. It shows the true positive rate against false positive rate with various classification threshold θ. Specifically, the area under the curve (AUC) is defined as $AUC = P(X_1 > X_0)$, where where X_1 is the score for a positive instance and X_0 is the score for a negative instance.

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 on the Android platform is enforced offline with `AndroidLibSvm`, where only the data generated from the device (positive sample) is 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.

5.1 Accuracy

Batch Learning-Experimental Data with Ground Truth. Since the experimental data is labeled, our online learning algorithm is not needed, and we simply train the classifier with the whole training set, in which the classification threshold θ was set to 0.5. Regarding the configuration of SVM, we set the cost value as 100 and γ as 0.01. All the trained classifiers for the 20 participants (*dataset I*) achieve high precision and recall. We summarize the performance of RiskCog on various datasets in

TABLE 6
The Performance of RiskCog on Different Datasets

		P_{owner}	R_{owner}	P_{other}	R_{other}
Dataset I	Steady	94.76%	71.76%	77.41%	96.53%
	Moving	94.15%	64.37%	74.07%	95.80%
Datatset II	Steady	88.07%	75.08%	97.28%	98.87%
Dataset III	- (Moto 360)	88.22%	93.09%	99.50%	99.11%
	- (iWatch)	95.98%	97.53%	99.82%	99.71%
Dataset IV	Steady	87.39%	73.28%	96.07%	98.43%
	Moving	89.35%	81.41%	97.81%	98.89%

TABLE 7
Comparison with Other Solutions on Our *dataset IV*

Study	Classifier	Accuracy	
		Steady	Moving
RiskCog	SVM (binary-class)	93.77%	95.57%
Zdeňka et al. (2016) [14]	SVM (one-class)	85.97%	87.05%
Lee et al. (2017) [15]	DTW	86.23%	86.74%
Buriro et al. (2017) [16]	Random forest	91.98%	92.74%
Shen et al. (2018) [17]	HMM	90.84%	90.23%

Table 6. 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 percent for the steady state. As for the moving state, the values are 94.15, 64.37, 74.07, and 95.80 percent, respectively. The average accuracy for the steady state is 84.15 percent, and that for the moving state is 80.09 percent. The results indicate that RiskCog causes low false positives, while the number of false negatives is relatively high. As mentioned before, for our training set, 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. One might argue that the differences between two people are mainly caused by the hardware of different phones [36] rather than the users manner of how to use the phone. Recalling that all the participants in the lab use the same phone, the high classification accuracy indicates that our user authentication is independent of the hardware sensor difference.

In Fig. 10, we use the ROC curve to depict the true positive rate against the false positive rate at various threshold θ. It 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 moving/steady states are 0.9043 and 0.9513. For security protection, a large FPR is more harmful to the device users than a small TPR. However, a small TPR would degrade the convenience of

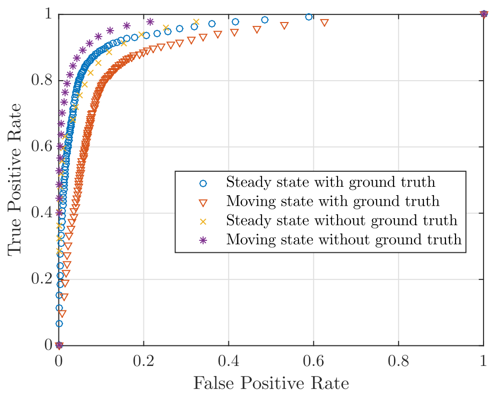


Fig. 10. ROC curve for 20 users with ground truth and 1,513 users without ground truth; the decision threshold θ varies from 0 to 1 at step growth 0.01.

using the system. RiskCog has enough space for tuning given various requirements of sensitivity and specificity.

For the 34 participants (*dataset II*) in the steady state from the iOS platform, we have the average values of P_{owner}, R_{owner}, P_{other} and R_{other} as 88.07, 75.08, 97.28, and 98.87 percent. Moreover, we achieve the average values of P_{owner}, R_{owner}, P_{other} and R_{other} as 88.22, 93.09, 99.50, and 99.11 percent for Android Watch, and 95.98, 97.53, 99.82, and 99.71 percent for iWatch (*dataset III*).

Online Learning-Raw Data without Ground Truth. In the training set, each user on average has 20,648 samples for the steady state and 9,280 samples for the moving state (*dataset IV*). The training set will be divided into 10 chunks. Regarding the conditions of accepting a chunk of data and training termination (Section 3.5), 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 Fig. 10, the areas of the two curves for moving/steady states are 0.9719 and 0.9506, respectively, for the 1,513 users without ground truth. The performance is slightly better than that in the laboratory setting, which could be attributed to the bigger size of the training set and the stratified sampling applied. It allows the classifier to differentiate the authorized user well from others.

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 percent in the steady state, and 89.35, 81.41, 97.81, and 98.89 percent in the moving state. The average accuracy values for the two states are 93.77 and 95.57 percent, respectively. Even with those challenges in the practical deployment, such as imbalanced dataset 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 full protection on the user's account. All the previous studies [6], [7], [8], [10], [11], [12], [13] rely on the features that are only available when the user is moving, such as step cycle. We also compared with the stat-of-the-art solutions in the recent years with the collected data and computed the corresponding accuracy in both our two motion states as shown in Table 7. We can see our approach performs better than all of other solutions on the large dataset with 1513 people. It's interesting to see the one-class SVM performs much worse than our binary-class SVM. The main reason is because the occasional overlapping among different person cannot be well addressed by one-class classification. Compared with

TABLE 8
Training Latency, AVG for Average, STD for Standard Deviation, and the Results are Based on the Data from 1,513 Users

		Steady	Moving
#Training samples	AVG	20,648	9,280
	STD	15,660	10,212
Training time (s)	AVG	148.36	21.21
	STD	177.23	24.06

TABLE 9
The Overhead Results on Three Different Smartphones Where 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

[15], RiskCog can reduce the number of explicit authentications a user must do by at least 73.2 percent (i.e., $99.9\% \times 73.28\%$) on average, where 99.9 percent is the coverage rate of users' patterns and 73.28 percent is the recall in steady state, while the value of study [15] is 32.9 percent. It shows that our system has a better applicability.

5.2 Overhead

We measure the latency of the training phase on the server, and the results are listed in Table 8. We set up the experiment on a server with an Intel Xeon E5 CPU and 64G memory running on Ubuntu 14.04. On average, RiskCog is able to analyze the user's motion data for 10 days (*dataset IV*) and train the classifiers corresponding to the steady/moving states within 150s. For each user, RiskCog will run on a single core with the memory usage 0.1 percent.

On the client side, we utilize the tool Emmagee[3] to assess the impact on battery consumption, CPU, and memory usage. Emmagee can sample the hardware resource usage of an app on the device. Table 9 shows the results. For the battery consumption, 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 percent 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 could be attributed 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 1,000 times on the device Mi 4, and record the average time for each step. The results are listed in Table 10. We can see the whole process can be finished within 3237.7 ms. The latency introduced by steps other than data collection is negligible.

5.3 Resistance to Brute-Force Attacks and Mimicry Attacks

For password cracking, the brute-force attack tries a huge set of possible keys against the credential verification module. For the mimicry attack, the attackers observe the user's usage manner and can impersonate and mimic the device owner's behavior. In our proposed system, 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 automatic attack and mimicry attack, respectively.

5.3.1 Automatic Attack

For the nine values collected from motion sensors, we define the starting point of data collection about the user owner 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. The bound for acceleration, gravity and gyroscope is ± 20, ± 10 and ± 0.2, respectively. 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.

Given the trained classifiers of 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 92.32 percent.

5.3.2 Mimicry Attack

We have humans to launch the mimicry attack. One classifier is trained for the authorized device owner by fingerprinting the usage manner. 19 participants observe how the device owner uses his/her smartphone and handle the owner's phone with the educated gestures for 100 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

TABLE 10
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

3. Emmagee. https://github.com/NetEase/Emmagee

TABLE 11
Sample Acquisition Time, System Usability Scale Summary for Different Methods Adapted from [61]

Method	Sample Acquisition Time (s)	SUS Score
RiskCog	3.2	84.5
PIN	3.8	78
Password	4.3	78
Voice	5.4	66
Face	3.9	75
Gesture	4.2	77
Face + Voice	5.3	46
Gesture + Voice	5.7	50

other users. RiskCog blocks the attacks by human with probability over 99.85 percent.

We can see there is difference between the percentages blocked by RiskCog in two attack scenarios. 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. From the perspective of our experiment, imitating the usage manner of the phone owner is time-consuming. Finally, we were able to gather total 4,900 data samples from each participant, 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.

We also consider a more sophisticated attack that an attacker tries to slowly shift the machine learning model. We asked 19 participants to imitate the user owner and each participant uses the owner's phone with the educated gestures for 30 times in the training phase. We found that the efficiency of attacks mainly depends on the degree of training. When the number of positive samples created by the user owner is less than 1,000, the ratio of the imposter sequences accepted by our semi-supervised online learning is 35.1 percent. While the number of positive samples is more than 4,000, the acceptance rate of the imposter sequences is 1.8 percent. It is reasonable considering that the model which is not well trained is more likely to be shifted by mimicry. The results suggest that users should take an additional effort in the training phase (e.g., setting up the password).

5.4 Usability Analysis

In Section 3.5, we have analyzed the relationship between the accuracy and the sample size of positive samples. In this section, we utilize the System Usability Scale (SUS) [60] to evaluate the usability of our system from users' perspective.

We distributed RiskCog along with a questionnaire adapted from the SUS to 20 participants (*dataset I*). The SUS has been used for measuring the system usability since 1986. It's also used by previous mobile authentication studies [61], [62]. All the participants are asked to score 10 questions with one of five responses that range from Strongly Agree (score is 10) to Strongly disagree (score is 0). The SUS score ranges from 0 to 100. Based on the research in [60], a SUS score above 68 would be considered above average.

We asked the participants to score RiskCog after having used the app for two weeks. We achieved the average SUS score of 84.5. It is higher than the average value 68. Table 11 summarizes the sample acquisition time and system usability scale for different methods adapted from [61], our score is better than the score of password (78), voice (66) and face (75) as reported in [61]. Most of the users appreciated the ease of use and effective functionality of RiskCog. We also got some negative responses, most of which are like "the training phase lasted a bit long". The main reason is that those users do not actually use the phone frequently but RiskCog needs enough data to train the classifier. In the future, we plan to exploit more potential features (e.g., using CNN, LSTM) to represent user's manner. Hopefully we can get a higher accuracy with a smaller training set (e.g., transfer learning) by those potential features.

6 Discussion

In this section, we discuss how an attacker aware of RiskCog design principle tries to evade user authentication or exploit it to, for example, make legitimate sensor data. For evasion, an attacker could exploit a user' pattern and pretend herself as the device owner.

6.1 Can Attackers Exploit Sensor Data to Pretend to Be the Legitimate User?

Since the data from the motion sensor is available for any apps to read, it is possible that attackers can exploit the data collection process of RiskCog. After extracting the sensor data, the attacker may use it to deduce usage pattern of the device owner. We argue that the above exploitation tactics can be combated through a slight modification when generating the train set, such as interchanging features in the sample vectors, e.g., $< F_1, F_2, \dots > \Rightarrow < F_2, F_1, \dots >$, which is totally invisible to the attackers. Consider the feature sequence $< F_1, F_2, \dots, F_{56} >$, we will get $56! \approx 7 * 10^{74}$ possible permutations after interchanging. Thus, even if the attacker obtains the well-pretended sensor data, he could not bypass our system as well.

6.2 How to Mitigate Deliberate API Hooking against User Authentication?

Attackers may customize the return values by hooking the motion sensor related APIs, and modify the original sensor data which was input into RiskCog. To mitigate this issue, the possible ways are i) to check if the phone is rooted. Hooking the low-level system APIs, such as getting motion sensor data, needs the root privilege of the device. Thus we can enforce root detection and deploy our service on those devices without being rooted; ii) to monitor the rationalization of raw sensor data. For example, to bypass our system, attackers may generate repeated dataset to satisfy the valid classifier. If a large number of repeated data is detected, the train process would stop and RiskCog would inform the users at the same time.

7 Related Work

In this section, we investigate the relevant studies and compare with them to highlight the novelty of our approaches.

Our Authentication does not Need User Movement or Fixed Device Placement. The use of sensor data for user authentication has been explored in recent years. Motion sensors, such as acceleration sensor, was used by [6], [7], [8], [11], [12], [13] to authenticate cell phone users. Derawi et al. [6] and Kwapisz et al. [7] made use of phone-based acceleration sensor to identify and authenticate cell phone users. Ren et al. [12] devised a user verification system leveraging the unique gait pattern derived from acceleration sensors to detect possible user spoofing in the mobile health care system. 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. [10] 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 actions 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 verification by 90 percent. All those 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. Given the rare limitations on the application scenario, our system can provide an additional 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.

We Design a Semi-Supervised Online Learning Algorithm with High Accuracy and Low Latency to Deal with the Unlabeled Data in a more Sophisticated Environment. SenSec [9] did the similar job as RiskCog. *SenSec* constantly collects sensory data from accelerometers, gyroscopes and magnetometers, and models how a user uses the device. The result showed that *SenSec* achieved an accuracy of 75 percent in identifying the users and 71.3 percent in detecting the non-owners. RiskCog gets a higher accuracy as well as a low latency compared with *SenSec*. Also, RiskCog can authenticate the owner once he/she opens an app and make the decision immediately, while the test phase of *SenSec* lasted 24 hours which made the real-time detection unpractical. Moreover, *SenSec* can not deal with the unlabeled data in a more sophisticated environment, while RiskCog can well address this issue by a semi-supervised online learning.

We Design a New Data Collection Mechanism with a High Coverage Rate of Users' Patterns. To validate that RiskCog is more tailored to user identification, we also compare our work with a recent implicit authentication system called *Secure Pick Up* [15]. *Secure Pick Up* can authenticate the user without his/her movement. Different from RiskCog, *Secure Pick Up* only extracts the pick-up movements starting from a stable state which has less representation because picking up the smartphone from a bag/pocket is very common in real world. In general, *Secure Pick Up* requires a stable picking up and can only detect 35.6 percent of users' pick-up movements. The requirement is hard to satisfy in the practical environment. On the contrary, the experiment on 1,513 users shows that RiskCog will recognize 99.9 percent (i.e., 283,006,659/283,133,354) of users' patterns. RiskCog verifies the identity, when the user is steady simply by the manner of handling the device.

We Select Features which can Represent Users' Patterns Without any In-App Invocation or User Privacy Tracking. Feng et al. [63] investigated authenticating users with touchscreen gestures, where they built a sensor glove to collect data. Touchscreen inputs were also used by [64], [65], [66] for identity verification. Sitová et al. [14], Buriro et al. [16] and Shen et al. [17] combine motion sensors and touchscreen for active smartphone authentication. They built an Android app to capture the touchscreen events because those data is only available at the application level. In contrast, RiskCog does not depend on external sensors and all required features are on the device level.

Our System has Cross-Platform Capability and Performs Better than Previous Work on Smartwatch. Motion sensors have been used on Smartwatch to identify users when the user wears the device. Previous work were mainly based on walk patterns [67] and custom gestures [68], [69]. Other studies [70], [71] combine a wearable smartwatch with a smartphone to authenticate user. Mare et al. [70] performed continuous authentication with 85 percent accuracy with a latency of 10-50 seconds. Yang et al. [68] and Lewis et al. [69] authenticated the owner utilizing explicit gestures such as circle and rotation which were customer by the users. Compared with the above work, RiskCog will make a high accuracy decision within 3 seconds once the smartwatch is being used (i.e., raising hands to check the time on the watch).

The above factors (in bold) face respective challenges, which can hardly be jointly addressed by merely adapting existing motion sensor-based authentication methods. RiskCog outperforms all the existing systems in that it overcomes these challenges all together.

8 Conclusion

In this paper, we present the system RiskCog to provide on-demand, offline unobtrusive user identity verification with a learning-based approach. The trained classifier depicts the owner's specific manner based on behavioral biometrics and usage patterns exhibited in daily device use. 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. By deploying RiskCog in the production environment on a large scale, we resolve several new issues, such as the imbalanced dataset and the training set without ground truth. With our stratified sampling method and a semi-supervised online learning method, we achieve the classification accuracy rates 93.77 and 95.57 percent among the 1,513 users for the steady state and the moving state, respectively. The authentication accuracy computed on Android Watch and iWatch are 98.71 and 99.56 percent, respectively. RiskCog can perform implicit authentication on any privacy-sensitive apps with a 99.9 percent coverage rate of users' patterns, and can reduce the number of times that a user has to do explicit authentication by at least 73.2 percent. Also, we demonstrate that

RiskCog can effectively defend against brute-force attacks and mimicry attacks.

Acknowledgments

The authors would like to thank Prof. Kai Bu for his valuable comments and careful proofreading. They also acknowledge the help from Mr. Qiang Liu in the data collection process. They would also like to thank the *IEEE Transactions on Mobile Computing* editors and reviewers for the review efforts.

References

[1] Information economy report 2017, [Online]. Available: http://unctad.org/en/PublicationsLibrary/ier2017_en.pdf, Accessed: Feb. 2018.

[2] A. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner, "A survey of mobile malware in the wild," in *Proc. 1st ACM Workshop Secur. Privacy Smartphones Mobile Devices*, 2011, pp. 3–14.

[3] U. Meyer and S. Wetzel, "A man-in-the-middle attack on UMTS," in *Proc. 3rd ACM Workshop Wireless Secur.*, 2004, pp. 90–97.

[4] The dirty dozen: 12 top cloud security threats for 2018, [Online]. Available: https://www.csoonline.com/article/3043030/security/12-top-cloud-security-threats-for-2018.html, Accessed: Feb. 2018.

[5] LexisNexis true cost of fraud mCommerce, [Online]. Available: http://lexisnexis.com/risk/downloads/whitepaper/true-cost-fraud-mobile-2014.pdf, Accessed: Oct. 2016.

[6] M. O. Derawi, C. Nickel, P. Bours, and C. Busch, "Unobtrusive user-authentication on mobile phones using biometric gait recognition," in *Proc. IEEE 6th Int. Conf. Intell. Inf. Hiding Multimedia Signal Process.*, 2010, pp. 306–311.

[7] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Cell phone-based biometric identification," in *Proc. 4th IEEE Int. Conf. Biometrics: Theory Appl. Syst.*, 2010, pp. 1–7.

[8] C. C. Ho, C. Eswaran, K.-W. Ng, and J.-Y. Leow, "An unobtrusive android person verification using accelerometer based gait," in *Proc. 10th Int. Conf. Advances Mobile Comput. Multimedia*, 2012, pp. 271–274.

[9] J. Zhu, P. Wu, X. Wang, and J. Zhang, "SenSec: Mobile security through passive sensing," in *Proc. IEEE Int. Conf. Comput. Netw. Commun.*, 2013, pp. 1128–1133.

[10] H. Lu, J. Huang, T. Saha, and L. Nachman, "Unobtrusive gait verification for mobile phones," in *Proc. ACM Int. Symp. Wearable Comput.*, 2014, pp. 91–98.

[11] H. G. Kayacik, M. Just, L. Baillie, D. Aspinall, and N. Micallef, "Data driven authentication: On the effectiveness of user behaviour modelling with mobile device sensors," in *Proc. 3rd Workshop Mobile Secur. Technol.*, 2014, pp. 1–10.

[12] Y. Ren, Y. Chen, M. C. Chuah, and J. Yang, "User verification leveraging gait recognition for smartphone enabled mobile healthcare systems," *IEEE Trans. Mobile Comput.*, vol. 14, no. 9, pp. 1961–1974, Sep. 2015.

[13] W.-H. Lee and R. B. Lee, "Multi-sensor authentication to improve smartphone security," in *Proc. IEEE Int. Conf. Inf. Syst. Secur. Privacy*, 2015, pp. 1–11.

[14] Z. Sitová, J. Šeděnka, Q. Yang, G. Peng, G. Zhou, P. Gasti, and K. S. Balagani, "HMOG: New behavioral biometric features for continuous authentication of smartphone users," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 5, pp. 877–892, May 2016.

[15] W.-H. Lee, X. Liu, Y. Shen, H. Jin, and R. B. Lee, "Secure pick up: Implicit authentication when you start using the smartphone," in *Proc. 22nd ACM Symp. Access Control Models Technol.*, 2017, pp. 67–78.

[16] A. Buriro, B. Crispo, and Y. Zhauniarovich, "Please hold on: Unobtrusive user authentication using smartphone's built-in sensors," in *Proc. IEEE Int. Conf. Identity Secur. Behavior Anal.*, 2017, pp. 1–8.

[17] C. Shen, Y. Li, Y. Chen, X. Guan, and R. A. Maxion, "Performance analysis of multi-motion sensor behavior for active smartphone authentication," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 1, pp. 48–62, Jan. 2018.

[18] S. Sen, E. Aydogan, and A. I. Aysan, "Coevolution of mobile malware and anti-malware," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 10, pp. 2563–2574, Oct. 2018.

[19] P. Wijesekera, A. Baokar, L. Tsai, J. Reardon, S. Egelman, D. Wagner, and K. Beznosov, "The feasibility of dynamically granted permissions: Aligning mobile privacy with user preferences," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 1077–1093.

[20] K. Olejnik, I. Dacosta, J. S. Machado, K. Huguenin, M. E. Khan, and J.-P. Hubaux, "SmarPer: Context-aware and automatic runtime-permissions for mobile devices," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 1058–1076.

[21] A. Continella, Y. Fratantonio, M. Lindorfer, A. Puccetti, A. Zand, C. Kruegel, and G. Vigna, "Obfuscation-resilient privacy leak detection for mobile apps through differential analysis," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2017, pp. 1–16.

[22] A. J. Aviv, K. L. Gibson, E. Mossop, M. Blaze, and J. M. Smith, "Smudge attacks on smartphone touch screens," in *Proc. 4th USENIX Conf. Offensive Technol.*, 2010, pp. 1–7.

[23] N. H. Zakaria, D. Griffiths, S. Brostoff, and J. Yan, "Shoulder surfing defence for recall-based graphical passwords," in *Proc. 7th Symp. Usable Privacy Secur.*, 2011, Art. no. 6.

[24] Z. Xu, K. Bai, and S. Zhu, "TapLogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors," in *Proc. 5th ACM Conf. Secur. Privacy Wireless Mobile Netw.*, 2012, pp. 113–124.

[25] S. Eberz, K. B. Rasmussen, V. Lenders, and I. Martinovic, "Evaluating behavioral biometrics for continuous authentication: Challenges and metrics," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, 2017, pp. 386–399.

[26] C. Bettini, X. S. Wang, and S. Jajodia, "Protecting privacy against location-based personal identification," in *Proc. Workshop Sec. Data Manage.*, 2005, pp. 185–199.

[27] W. Shi, J. Yang, Y. Jiang, F. Yang, and Y. Xiong, "SenGuard: Passive user identification on smartphones using multiple sensors," in *Proc. IEEE 7th Int. Conf. Wireless Mobile Comput. Netw. Commun.*, 2011, pp. 141–148.

[28] K. S. Killourhy and R. A. Maxion, "Comparing anomaly-detection algorithms for keystroke dynamics," in *Proc. IEEE/IFIP Int. Conf. Depend. Syst. Netw.*, 2009, pp. 125–134.

[29] H. Gascon, S. Uellenbeck, C. Wolf, and K. Rieck, "Continuous authentication on mobile devices by analysis of typing motion behavior," in *Proc. GI Conf. Sicherheit*, 2014, pp. 1–12.

[30] L. Li, X. Zhao, and G. Xue, "Unobservable re-authentication for smartphones," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2013, pp. 1–16.

[31] Z. Sitova, J. Sedenka, Q. Yang, G. Peng, G. Zhou, P. Gasti, and K. S. Balagani, "HMOG: A new biometric modality for continuous authentication of smartphone users," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 5, pp. 877–892, 2016.

[32] R. D. Findling, M. Holzl, and R. Mayrhofer, "Mobile match-on-card authentication using offline-simplified models with gait and face biometrics," *IEEE Trans. Mobile Computing*, vol. 17, no. 11, pp. 2578–2590, Nov. 2018.

[33] Android's biggest problem is operating system fragmentation, [Online]. Available: http://o.canada.com/technology/personal-tech/androids-biggest-problem-is-operating-system-segmentation, Accessed: Feb. 2018.

[34] You won't believe the 20 most popular cloud service passwords, [Online]. Available: https://www.skyhighnetworks.com/cloud-security-blog/you-wont-believe-the-20-most-popular-cloud-service-passwords/, Accessed: Feb. 2018.

[35] D. Florencio and C. Herley, "A large-scale study of web password habits," in *Proc. 16th Int. Conf. World Wide Web*, 2007, pp. 657–666.

[36] S. Dey, N. Roy, W. Xu, R. R. Choudhury, and S. Nelakuditi, "AccelPrint: Imperfections of accelerometers make smartphones trackable," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2014, pp. 1–16.

[37] A. Das, N. Borisov, and M. Caesar, "Tracking mobile web users through motion sensors: Attacks and defenses," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2016, pp. 1–15.

[38] E. Vazquez-Fernandez and D. Gonzalez-Jimenez, "Face recognition for authentication on mobile devices," *Image Vis. Comput.*, vol. 55, pp. 31–33, 2016.

[39] Alibaba uses facial recognition tech for online payments, [Online]. Available: http://www.computerworld.com/article/2897117/alibaba-uses-facial-recogn ition-tech-for-online-payments.html, Accessed: Feb. 2018.

[40] P. Golle and K. Partridge, "On the anonymity of home/work location pairs," in *Proc. Int. Conf. Pervasive Comput.*, 2009, pp. 390–397.

[41] Android Wear on Wi-Fi: Using a smartwatch without a phone nearby, [Online]. Available: http://www.computerworld.com/article/2919013/android/android-wear-on-wi-fi-using-a-smartwatch-without-a-phone-nearby.html, Accessed: Feb. 2018.

[42] A. Pande, Y. Zeng, A. K. Das, P. Mohapatra, S. Miyamoto, E. Seto, E. K. Henricson, and J. J. Han, "Energy expenditure estimation with smartphone body sensors," in *Proc. 8th Int. Conf. Body Area Netw.*, 2013, pp. 8–14.

[43] SensorManager, [Online]. Available: https://developer.android.google.cn/reference/android/hardware/SensorManager, Accessed: Feb. 2018.

[44] Android service, [Online]. Available: https://developer.android.com/guide/components/services.html, Accessed: Feb. 2018.

[45] Fleming's right-hand rule, [Online]. Available: https://en.wikipedia.org/wiki/Fleming's_right-hand_rule, Accessed: Feb. 2018.

[46] LibXtract Documentation, [Online]. Available: http://jamiebullock.github.io/LibXtract/documentation/index.html, Accessed: Feb. 2018.

[47] M. Welling, "Fisher linear discriminant analysis," Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, Tech. Rep. #3, 2005.

[48] J. E. Trost, "Statistically nonrepresentative stratified sampling: A sampling technique for qualitative studies," *Qualitative Sociology*, vol. 9, no. 1, pp. 54–57, 1986.

[49] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik, "Feature selection for SVMs," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2001, pp. 668–674.

[50] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *Proc. Eur. Conf. Mach. Learn.*, 1998, pp. 137–142.

[51] A. Ghodsi, "Dimensionality reduction a short tutorial," Ph.D. dissertation, Dept. Statist. Actuarial Sci., Univ. Waterloo, ON, CA, 2006.

[52] RBF SVM parameters, [Online]. Available: http://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html, Accessed: Feb. 2018.

[53] Android BroadcastReceiver, [Online]. Available: https://developer.android.com/reference/android/content/BroadcastReceiver.html, Accessed: Feb. 2018.

[54] Android UsageStatsManager, [Online]. Available: https://developer.android.com/reference/android/app/usage/UsageStatsManager.html, Accessed: Feb. 2018.

[55] Building Apps for Wearables, [Online]. Available: https://developer.android.com/training/building-wearables.html, Accessed: Feb. 2018.

[56] watchOS + Apps, [Online]. Available:https://developer.apple.com/watchos/, Accessed: Feb. 2018.

[57] Django, [Online]. Available: https://www.djangoproject.com, Accessed: Feb. 2018.

[58] LibSvm, [Online]. Available: https://www.csie.ntu.edu.tw/cjlin/libsvm/, Accessed: Feb. 2018.

[59] AndroidLibSvm, [Online]. Available: https://github.com/yctung/AndroidLibSvm, Accessed: Feb. 2018.

[60] System Usability Scale, [Online]. Available: https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html, Accessed: Feb. 2018.

[61] S. Trewin, C. Swart, L. Koved, J. Martino, K. Singh, and S. Ben-David, "Biometric authentication on a mobile device: A study of user effort, error and task disruption," in *Proc. 28th Annu. Comput. Secur. Appl. Conf.*, 2012, pp. 159–168.

[62] A. Buriro, B. Crispo, F. Delfrari, and K. Wrona, "Hold and sign: A novel behavioral biometrics for smartphone user authentication," in *Proc. IEEE Secur. Privacy Workshops*, 2016, pp. 276–285.

[63] T. Feng, Z. Liu, K.-A. Kwon, W. Shi, B. Carbunar, Y. Jiang, and N. Nguyen, "Continuous mobile authentication using touchscreen gestures," in *Proc. IEEE Conf. Technol. Homeland Secur.*, 2012, pp. 451–456.

[64] M. Frank, R. Biedert, E. Ma, I. Martinovic, and D. Song, "Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication," *IEEE Trans. Inf. Forensics Secur.*, vol. 8, no. 1, pp. 136–148, Jan. 2013.

[65] R. Murmuria, A. Stavrou, D. Barbará, and D. Fleck, "Continuous authentication on mobile devices using power consumption, touch gestures and physical movement of users," in *Proc. Int. Symp. Recent Advances Intrusion Detection*, 2015, pp. 405–424.

[66] V. Sharma and R. Enbody, "User authentication and identification from user interface interactions on touch-enabled devices," in *Proc. 10th ACM Conf. Secur. Privacy Wireless Mobile Netw.*, 2017, pp. 1–11.

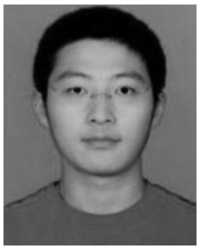
[67] A. H. Johnston and G. M. Weiss, "Smartwatch-based biometric gait recognition," in *Proc. IEEE 7th Int. Conf. Biometrics Theory Appl. Syst.*, 2015, pp. 1–6.

[68] J. Yang, Y. Li, and M. Xie, "MotionAuth: Motion-based authentication for wrist worn smart devices," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops*, 2015, pp. 550–555.

[69] A. Lewis, Y. Li, and M. Xie, "Real time motion-based authentication for smartwatch," in *Proc. IEEE Conf. Commun. Netw. Secur.*, 2016, pp. 380–381.

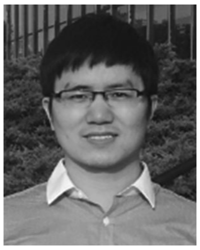
[70] S. Mare, A. M. Markham, C. Cornelius, R. Peterson, and D. Kotz, "ZEBRA: Zero-effort bilateral recurring authentication," in *Proc. IEEE Symp. Secur. Privacy*, 2014, pp. 705–720.

[71] W.-H. Lee and R. Lee, "Implicit sensor-based authentication of smartphone users with smartwatch," in *Proc. Hardware Archit. Support Secur. Privacy*, 2016, Art. no. 9.



Tiantian Zhu received the BEng degree in information security from Northeastern University, Shenyang, China, in 2014. He is currently working toward the PhD degree in the College of computer science and technology, Zhejiang University, Hangzhou, China. His research interests include mobile security, OSN security, and artificial intelligence.

Zhengyang Qu received the BE degree in electrical engineering from Shanghai Jiao Tong University, and the PhD degree in computer science from Northwestern University. He currently serves as a research scientist at Facebook. His research interests include mobile security and user privacy.



Haitao Xu received the PhD degree in computer science from the College of William and Mary, VA, in December 2015. He is an assistant professor with the School of Mathematical and Natural Sciences, Arizona State University, AZ. His research focuses on the intersection of cyber security, privacy, and data analytics.

Jingsi Zhang received the PhD degree in statistics from Northwestern University, in 2018. She is a senior data scientist at WalmartLabs. Her professional experience spans from advanced machine learning, deep learning, and large-scale distributed system to statistical analysis and data visualization.

Zhengyue Shao received the BEng degree from the Nanjing University of Aeronautics and Astronautics, China, in 2014, and the master's degree from Zhejiang University, Hangzhou, China. He currently serves as the software engineer at Microsoft. His research interests include mobile security.

Yan Chen received the PhD degree in computer science from the University of California, Berkeley, CA, in 2003. He is a professor with the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL. Based on Google Scholar, his papers have been cited more than 7,000 times and his h-index is 34. His research interests include network security, measurement, and diagnosis for large-scale networks and distributed systems. He won the Department of Energy (DoE) Early CAREER Award in 2005, the Department of Defense (DoD) Young Investigator Award in 2007, and the Best Paper nomination in ACM SIGCOMM 2010. He is a fellow of the IEEE.

Sandeep Prabhakar received the BE degree in computer science engineering from the PES Institute of Technology, and the master's degree in computer science from Northwestern University. He currently is a member of technical staff at SalesforceIQ.

Jianfeng Yang received the bachelor's, master's, and PhD degrees in information and communication engineering from Wuhan University, in July 1998, 2002, and 2009, respectively. He currently is an associate professor at Wuhan University. He worked as a visiting scholar at the Intel Company in 2012 and Northwestern University from 2015 to 2016. His research interests are in security and measurement for networking, edge computing, and high-reliability real-time wireless communication.