Implicit Smartphone User Authentication with Sensors and Contextual Machine Learning

Wei-Han Lee and Ruby B. Lee
Princeton University
Email: {weihanl, rblee@princeton.edu }

***Abstract*—Authentication of smartphone users is important because a lot of sensitive data is stored in the smartphone and the smartphone is also used to access various cloud data and services. However, smartphones are easily stolen or co-opted by an attacker. Beyond the initial login, it is highly desirable to re-authenticate end-users who are continuing to access security-critical services and data. Hence, this paper proposes a novel authentication system for implicit, continuous authentication of the smartphone user based on behavioral characteristics, by leveraging the sensors already ubiquitously built into smartphones. We propose novel context-based authentication models to differentiate the legitimate smartphone owner versus other users. We systematically show how to achieve high authentication accuracy with different design alternatives in sensor and feature selection, machine learning techniques, context detection and multiple devices. Our system can achieve excellent authentication performance with 98.1% accuracy with negligible system overhead and less than 2.4% battery consumption.**

I. Introduction

Increasing amounts of private and sensitive information are stored in our smartphones. 92.8% of Android smartphone users store private information in their smartphones [1], [2]. Smartphones have also become personal computing platforms for users to access cloud services, e.g., e-banking and online social networks. Hence, smartphones are very attractive targets for attackers to get access to personal and valuable information. User authentication is essential to prevent the privacy, confidentiality and integrity breaches possible through attacks on the smartphone.

Current login mechanisms use explicit authentication, which requires the user's participation, e.g., passwords and fingerprints. Iris scanning [3] and facial recognition [4], [5] can also be used for explicit authentication. However, re-authentication to access very sensitive information via explicit authentication mechanisms is not convenient [6] for smartphone users. Hence, after the user passes the initial authentication, the system does not authenticate the user again. This creates a significant risk for adversaries to take control of the users' smartphones, after the legitimate users' initial login. This enables the adversaries to access proprietary or sensitive data and services, whether stored in the cloud or in the mobile device itself.

To protect smartphone data and cloud-based services from adversaries who masquerade as legitimate users, we propose a secure re-authentication system, which is both *implicit* and *continuous*. An implicit authentication method does not rely on the direct involvement of the user, but is closely related to her behavior recorded by the smartphone's built-in hardware, e.g., sensors, GPS and touchscreen. An implicitly continuous re-authentication method should keep authenticating the user, in addition to the initial login authentication, without interrupting users. This can detect an adversary once he gets control of the smartphone and can prevent him from accessing sensitive data or services via smartphones, or inside smartphones.

Our system, called SmarterYou, exploits one of the most important differences between personal computers and smartphones: a variety of sensors built into the smartphone, such as the accelerometer and gyroscope. SmarterYou also exploits the increasing number of wearable devices with Bluetooth connectivity and multiple sensors, e.g., smartwatches.

SmarterYou has the following advantages compared with previous smartphone authentication methods: (1) Instead of the explicit one-time authentication on log-in, e.g., using passwords, fingerprints or touchscreen patterns [7], [8], SmarterYou enables implicit, continuous authentication as a background service, when the users use smartphones. This can also be used in addition to the explicit authentication methods. (2) We do not require user's permissions. Many past approaches require the user's permission to get access to the hardware in the smartphone, e.g., GPS [9] and microphone [10]. Access to these hardware require permission because they contain private information of the user. (e.g., her location and phone conversations). (3) Some past work had high authentication errors [11], [12]. Our approach can have accuracy up to 98.1%. (4) Many approaches utilize the touchscreen to analyze user's writing or sliding patterns. However, the touchscreen information may leak out sensitive information, e.g., passwords or PINs [13], [14]. (5) Many past approaches only work under some specific context [15], [16], [17], [12], [18]. In SmarterYou, we utilize multiple contexts to improve authentication accuracy, and also design a context detection method that is user-agnostic.

In this paper, we utilize context detection techniques and multiple mobile devices to achieve accurate authentication performance stealthily, efficiently, and continuously. Also, we protect cloud-customers' services and data from malicious end-users using smartphone sensors. We also provide a systematic evaluation of the design alternatives for our system, in terms of sensors, features, contexts, multiple devices and machine learning algorithms. Our key contributions are:

- Design of an implicit authentication system, SmarterYou, by combining a user's information recorded in the smartphone and wearable devices. Our system continuously monitors a

TABLE I
COMPARISON OF OUR METHOD WITH OTHER IMPLICIT AUTHENTICATION (IF THE INFORMATION IS GIVEN IN THE PAPER CITED, OTHERWISE IT IS SHOWN AS N.A. (NOT AVAILABLE)) FOR AUTHENTICATION ACCURACY, FALSE ACCEPT RATE (FAR) AND FALSE REJECT RATE (FRR).

	Modality	Performance			# of Users
		Accuracy	FAR	FRR	
[17] Trojahn et al. 2013	Touchscreen	n.a.	11%	16%	18
[19] Frank et al. 2013	Touchscreen	96%	n.a.	n.a.	41
[20] Li et al. 2013	Touchscreen	95.7%	n.a.	n.a.	75
[21] Feng et al. 2012	Touchscreen & accelerometer & gyroscope	n.a.	4.66%	0.13%	40
[22] Xu et al. 2014	Touchscreen	$> 90\%$	n.a.	n.a.	31
[23] Zheng et al. 2014	Touchscreen & accelerometer	96.35%	n.a	n.a.	80
[15] Conti et al. 2011	accelerometer & orientation	n.a.	4.44%	9.33%	10
[24] Kayacik et al. 2014	accelerometer & orientation & magnetometer & light	n.a.	n.a.	n.a.	4
[11] Zhu et al. 2013	accelerometer & orientation & magnetometer	75%	n.a.	n.a.	20
[16] Nickel et al. 2012	accelerometer	n.a.	3.97%	22.22%	20
[25] Lee et al. 2015	accelerometer & orientation & magnetometer	90%	n.a.	n.a.	4
[26] Yang et al. 2015	accelerometer	n.a.	15%	10%	200
[9] Buthpitiya et al. 2011	GPS	86.6%	n.a.	n.a.	30
SmarterYou (this paper) 2017	accelerometer & gyroscope	98.1%	2.8%	0.9%	35

user's behavior and re-authenticates the user in an accurate, efficient, and stealthy manner.

- Design of a user-agnostic context detection approach to differentiate various usage contexts of the user. We determine the minimum number of contexts that give the best improvement in authentication performance.
- Design and evaluation of alternatives for all aspects of an efficient authentication method based on sensor measurements used as behavioral patterns. We consider the minimum number of sensors for high authentication accuracy, the best features in both time and frequency domains, the benefit of using multiple devices with sensors, the advantages of user context-specific authentication models, and alternative machine learning algorithms. To the best of our knowledge, this is the first systematic evaluation of design alternatives for sensor-based user authentication.
- SmarterYou also provides automatic and continuous retraining if the user's behavioral pattern changes over time. We also evaluate the performance overhead and battery consumption of the system. Our approach can achieve high authentication accuracy up to 98.1% with negligible system overhead and less than 2.4% battery consumption.

II. BACKGROUND AND RELATED WORK

Traditional authentication approaches are based on possession of secret information, such as passwords. Also, physiological biometrics based approaches make use of distinct personal features, such as fingerprint or iris patterns. Recently, behavior-based authentication utilize the distinct behavior of users.

There are many different physiological biometrics for authentication, such as face patterns [5], fingerprints [27], and iris patterns [3]. However, physiology-based authentication requires user participation in the authentication. Thus, they are more useful for initial login authentication instead of implicit and continuous authentication and re-authentication.

Behavior based authentication assumes that people have distinct, mostly stable, patterns for a certain behavior, such as gesture pattern [17], gait [16] and GPS patterns [9]. Behavior-based authentication exploits users' behavioral patterns to authenticate a user's identity. Below we review past work in this area and summarize them in Table I.

Touchscreen-based Smartphone Authentication.

Trojahn et al. [17] developed a mixture of a keystroke-based and handwriting-based mechanisms to realize authentication through the touchscreen sensor. Their approach has achieved 11% false accept rate (FAR) and 16% false reject rate (FRR). Frank et al. [19] utilize 22 analytic features from sliding traces to differentiate users. Their result can achieve 4% equal error rate. Li et al. [20] exploited five basic movements (sliding up, down, right, left and tapping) on the touchscreen and their related combinations as the user's behavioral pattern features, to perform authentication. Their result shows that sliding up can achieve the best accuracy of 95.7%. Feng et al. [21] utilize touchscreen with sensor gloves to record the fine-grained information of gestures. They can achieve up to 4.66% FAR and 0.13% FRR. Xu et al. [22] combine the slide, keystroke, handwriting and pinch to authenticate the user. Zheng et al. [23] combine the accelerometer with the touchscreen to authenticate a user when the user is entering her PIN.

Touchscreen-based authentication can achieve high accuracy. However, Serwadda et al. [28] showed that gesture styles could be observed and replicated automatically. Also, the touchscreen information contains sensitive information, e.g., the attacker may use the touchscreen information to find out the user's passwords [13].

Sensor-based Smartphone Authentication.

Conti et al. [15] proposed to authenticate a user using the arm movement patterns, sensed by the accelerometer and orientation sensor, while the user is making a phone call. Their method achieved 4.4% FAR and 9.3% FRR. Kayacik et al. [24] proposed a light-weight, and temporally & spatially aware user behavioral model for user authentication based on both hard and soft sensors. However, they did not quantitatively show their authentication performance. SenSec [11] constantly collects data from the accelerometer, gyroscope and magnetometer, to construct gesture models while the user is using the device. SenSec has shown it can

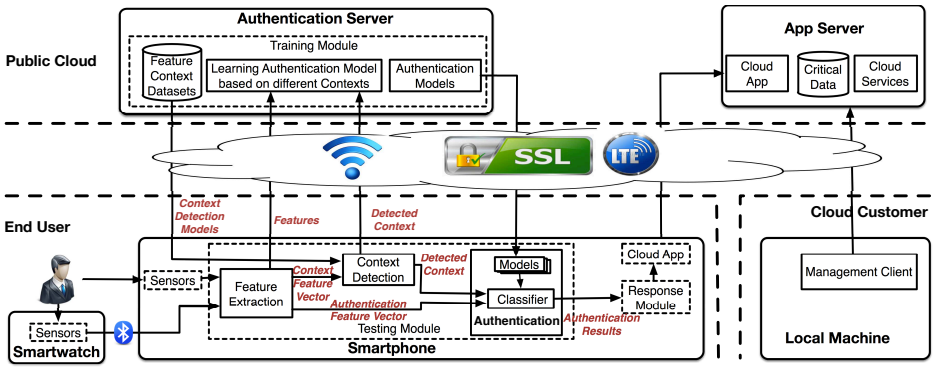


Fig. 1. SmarterYou architecture including the cloud-based training module and smartphone-based authentication module

achieve 75% accuracy in identifying owners. Nickel et al. [16] proposed an accelerometer-based behavior recognition method to authenticate a smartphone user through the k-NN algorithm. They can achieve 3.97% FAR and 22.22% FRR. Lee et al. [25] showed that using more sensors can improve the authentication performance. They monitored the users' living patterns and utilized SVM as a classifier for user authentication. Their result achieves 90% accuracy. Yang et al. [26] propose a hand waving biometric-based authentication method that utilise users' waving patterns for locking and unlocking the smartphone by using the accelerometer. They can achieve 15% FAR and 10% FRR on average. In [9], a geo-based authentication is proposed for modeling a user's mobility pattern. They use the GPS sensor to demonstrate that the system could detect abnormal activities (e.g., a phone being stolen) by analyzing a user's location history, and they can achieve 86.6% accuracy. However, the GPS information is sensitive, thus its use requires explicit user permission.

Different from these past methods, our SmarterYou system has broader contexts and the highest authentication accuracy (98.1%), with low computational complexity.

Continuous and Context-based Authentication.

Riva et al. [10] built a prototype to use face recognition, proximity, phone placement, and voice recognition to progressively authenticate a user. However, their objective is to decide when to authenticate the user and is thus orthogonal to our setting. Their prototype has a 42% reduction in requested explicit authentication, but this was conducted with 9 users only. Their scheme also requires access to sensors that need users' permissions, limiting their applicability for implicit, continuous authentication proposed in our system.

Existing continuous authentication approaches [17], [16], [15] focused on a specific usage context and would fail if the attacker who steals the smartphone does not perform under the specific usage context. In contrast, our system can automatically detect a context in a user-agnostic manner and can continuously authenticate a user based on various authentication models. That is, our system can authenticate the users without requiring any specific usage context, making it more applicable in real world scenarios.

Authentication with Wearable Devices.

Recently, wearable devices have emerged in our daily lives. However, limited research has been done on authenticating users by these wearable devices. Mare et al. [29] proposed ZEBRA which is a bilateral recurring authentication method. The signals sent from a bracelet worn on the user's wrist are correlated with the terminal's operations to confirm the continued presence of the user if the two movements correlate according to a few coarse-grained actions. To the best of our knowledge, there is no smartphone authentication research proposed in the literature that combines a wearable smartwatch with a smartphone to authenticate a user, as we do.

III. Threat Model and Assumptions

We consider an attacker who has physical access to a smartphone. The smartphone may even have passed an initial explicit login authentication, giving the attacker opportunity to access secure or private information on the phone and in the cloud using the phone. Confidentiality, integrity, authentication and privacy breaches are considered.

Wearable devices are gaining popularity, e.g., smartwatches and fitbits. They also contain many sensors e.g., accelerometer, gyroscope, ambient light and heartbeat sensors, and can communicate with smartphones via Bluetooth. We assume each smartwatch (and smartphone) is associated with one owner/user and that users do not share their smartwatches (and smartphones). We assume the communication between the smartwatch and smartphone is secure. We do not assume that users always have their smartwatch with them, so authentication based on smartphone alone is in scope.

While network access is required for authentication model training, or retraining after behavioral drift, network access is not required for user authentication (testing) when the smartphone is being used.

IV. System Design

A. Architecture Overview

Figure 1 shows the proposed SmarterYou architecture. It includes three hardware devices: the user-owned wearable device (e.g., smartwatch), the smartphone, and the authentication server in the cloud.

1) Wearable IoT device: In SmarterYou, we consider a two-device authentication configuration, which includes a smartphone and a user-owned wearable device. We use a smartwatch as an example, but other types of wearable devices, e.g., health sensors, can also be applied to SmarterYou. SmarterYou is designed for implicit authentication on the smartphone, where the smartwatch serves as important auxiliary information for improving authentication accuracy. The smartwatch keeps monitoring a user's raw sensors' data and sends the information to the smartphone via Bluetooth. Our system works if only the smartphone is present, but we will show that it works even better if the smartwatch is also present.

2) Smartphone: Similar to the smartwatch, the smartphone also monitors the user's sensor data. It runs the authentication testing module as a background service in the smartphone. In the testing module, the feature extraction component receives the sensor data from the smartphone and smartwatch. Then it extracts fine-grained time-frequency features from the raw data, and forms two feature vectors: the context feature vector and the authentication feature vector, and feeds them into the context detection component and the authentication component, respectively. The context detection component decides which context the user is in and sends the detected context to the authentication component.

The authentication component consists of a classifier and multiple authentication models. The classification algorithm we selected is the kernel ridge regression (KRR) algorithm [30], but other machine learning algorithms can also be used. An authentication model is a file containing parameters for the classification algorithm and determines the classifier's functionality. Using different authentication models for different contexts, the classifier can authenticate the user based on the authentication feature vector under different contexts. When a detected context and an authentication feature vector is fed in, the classifier chooses the corresponding authentication model and makes a classification.

When the classifier in the authentication component generates the authentication results, it sends these results to the Response Module. If the authentication results indicate the user is legitimate, the Response Module will allow the user to use the cloud apps to access the critical data or cloud services in the app server. Otherwise, the Response Module can either lock the smartphone or refuse accesses to security-critical data, or perform further checking. Our system can be used with existing explicit authentication methods, e.g., passwords or fingerprints. If the attacker is locked out, the system requires explicit authentication.

3) Authentication Server: SmarterYou includes a training module, which is deployed in the Authentication Server in the cloud, because it requires significant computation and must consider the privacy of the training data set, which includes data from other users. When a legitimate user first enrolls in the system, she downloads the context detection model from the Authentication Server and then the system keeps collecting the legitimate user's authentication feature vectors and detected contexts for training the authentication models. Our system deploys a trusted Authentication cloud server to collect sensors' data from all the participating legitimate users. To protect a legitimate user's privacy, the users' data are anonymized. In this way, a user's training module can use other users' feature data but has no way to know the other users' identities. The training module uses the legitimate user's authentication feature vectors and other people's authentication feature vectors in the training algorithm to obtain the authentication models based on different contexts. After training, the authentication models are downloaded to the smartphone. The training module does not participate in the authentication testing process and is only needed for retraining when the device recognizes a user's behavioral drift, which is done online and automatically. Therefore, our system does not pose a high requirement on the communication delay between the smartphone and the Authentication Server.

B. System Operation

SmarterYou is based on the observation that users' behavioral patterns are different from person to person, and vary under different usage contexts, when they use smartphones and smartwatches. Instead of authenticating the user with one unified model as in [15], [16], [17], [12], [18], [31], it is better to explore different finer-grained models to authenticate the user based on different usage contexts. For example, using a user's walking behavioral model to authenticate the same user who is sitting while using the smartphone is obviously not accurate. In Table VII, we show that considering contexts provides better accuracy. To be applicable in real world scenarios, we assume that the context information is user-agnostic: we can detect the context of the current user prior to authenticating her (as validated in Section V-E). Under each context, each user has distinct behavioral characteristics. SmarterYou utilizes such characteristics to implicitly authenticate the users. Our system can be used with other context detection methods [32], [33]. Context detection is an interesting research area e.g., Chen et al. [32] show that they can achieve up to 99% accuracy in context detection. In this paper, we show that by considering even simple contexts, we can improve the authentication accuracy significantly. More contexts, appropriately chosen, may further improve the authentication accuracy.

There are two phases for learning and classifying the user's behavioral pattern: *enrollment phase* and *continuous authentication phase*.

Enrollment Phase: Initially, the system must be trained in an enrollment phase. When users want to use the apps in the smartphone to access sensitive data or cloud services, the system starts to monitor the sensors and extract particular features from the sensors' data and label them with a context based on the context detection approach in Section V-E. This process continues and the data should be stored in a protected buffer in the smartphone until the distribution of the collected features converges to an equilibrium, which means the size of data can provide enough information to build a user's profile. This is about 800 measurements for our method, as shown in Section V-F3. At this time, one can assume that 1) the user got

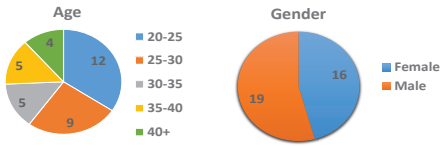


Fig. 2. Demographics of the participants

used to her device and her device-specific 'sensor-behavior' no longer changes, and 2) the system has observed sufficient information to have a stable estimate of the true underlying behavioral pattern of that user. The system can now train the authentication classifiers under various contexts and switch to the continuous authentication phase.

Continuous Authentication Phase: Once the authentication classifiers are trained and sent to the smartphone, the smartphone can start the authentication phase. This is done only in the smartphone, so network availability is not required. Based on the sensor data, SmarterYou first decides which context the user is in and then uses the authentication classifier for the detected context. The authentication classifier then decides whether these sensors' data are coming from the legitimate user. The authentication classifier can also be automatically updated when the legitimate user's behavioral pattern changes with time.

Post-Authentication: If the authentication feature vector is authenticated as coming from the legitimate user, this testing passes and the user can keep accessing the sensitive data in the smartphone or in the cloud via the smartphone. When an attacker tries to access a smartphone of a legitimate user, the system automatically de-authenticates him. Once SmarterYou decides that the smartphone is now being used by someone other than the legitimate user, the system can perform defensive responses as described earlier. Similarly, if the legitimate user is misclassified, several mechanisms for re-instating her are possible, such as two-channel or multi-factor authentication, or requiring an explicit login again, possibly with a biometric, to unlock the system.

Retraining Models: The behavioral patterns of SmarterYou users could be changed some time after the initial model training. So it is necessary to retrain users' models to prevent false alarms due to legitimate behavioral drift. SmarterYou provides a model retraining mechanism, which can automatically and continuously retrain the models based on the authentication performance. We define a metric called Confidence Score (CS) to measure if it is necessary to retrain the model. If so, SmarterYou will again upload the legitimate user's latest authentication feature vectors to the cloud server, and update the new models from the training module. It is important to note that adversaries can also exploit this mechanism to retrain the authentication models and achieve accesses to sensitive data with the smartphone. We use multi-factor authentication to prevent these potential vulnerabilities (details in Section V-I).

TABLE II
FISHER SCORES OF DIFFERENT SENSORS.

	Smartphone	Smartwatch
Acc(x)	3.13	3.62
Acc(y)	0.8	0.59
Acc(z)	0.38	0.89
Mag(x)	0.005	0.003
Mag(y)	0.001	0.0049
Mag(z)	0.0025	0.0002
Gyr(x)	0.57	0.24
Gyr(y)	1.12	1.09
Gyr(z)	4.074	0.59
Ori(x)	0.0049	0.0027
Ori(y)	0.002	0.0043
Ori(z)	0.0033	0.0001
Light	0.0091	0.0428

C. Security Protections

Protecting data in transit. Since sensitive data are being transmitted between smartwatches, smartphones and cloud servers, secure communications protocols must be used to provide confidentiality and integrity protection against network adversaries. For instance, an initialization key is exchanged when the smartwatch is paired with the smartphone using Bluetooth. New keys derived from this key can also be used to encrypt and hash the raw data transmitting between smartwatch and smartphone via Bluetooth. The communication channels between smartphones and cloud servers are protected by SSL/TLS protocols.

Protecting data at rest (i.e., in storage). For data stored in the smartphones or cloud servers, cryptographic encryption and hashing operations are used to prevent the attackers from stealing or modifying data.

Protecting data and code at runtime. The smartphone and Authentication Server must also provide a secure environment for running the SmarterYou authentication System. Since most smartphones use ARM processors, smartphones can exploit the ARM TrustZone [34] feature to place the authentication Testing Module in the Secure World and isolate it from other apps in the Normal World. Since cloud servers tend to use Intel processors, the trusted Authentication Server can set up secure enclaves by using Intel Sofware Guard eXtensions (SGX) [35] for the training and retraining modules for SmarterYou, and for securely accessing and using sensitive behavioral measurements from many smartphone users.

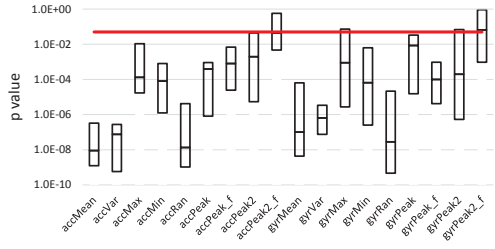
V. DESIGN ALTERNATIVES

Although we have outlined the basic architecture for our system, there are many design parameters that have yet to be chosen. Our goal is to get the highest authentication accuracy using the most commonly available sensors and computationally simple algorithms, to facilitate rapid deployment. What sensors should we use? What features of the raw sensor data streams are best? Can sensors from different devices help improve accuracy? Can contexts improve authentication accuracy, and if so, what are the simplest contexts that give the best accuracy? Which machine learning algorithms are best? Below, we systematically evaluate alternatives for each of these design choices.

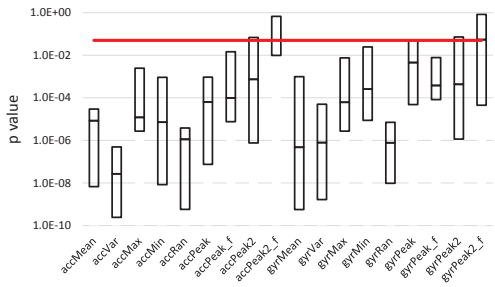
TABLE III

CORRELATIONS BETWEEN EACH PAIR OF FEATURES. THE UPPER TRIANGLE IS THE CORRELATION BETWEEN FEATURES IN THE SMARTPHONE, WHILE THE LOWER TRIANGLE IS THE CORRELATION BETWEEN FEATURES IN THE SMARTWATCH.

		Accelerometer								Gyroscope							
		Mean	Var	Max	Min	Ran	Peak	Peak_f	Peak2	Mean	Var	Max	Min	Ran	Peak	Peak_f	Peak2
Accelerometer	Mean		0.39	0.35	**0.59**	0.27	-0.12	-0.15	0.31	0.30	0.17	0.11	-0.26	0.13	0.27	0.04	0.34
	Var	0.11		0.28	-0.26	**0.90**	0.35	0.30	0.41	0.12	-0.29	0.10	0.16	-0.33	0.25	0.20	0.18
	Max	0.42	0.37		-0.22	**0.78**	0.35	0.23	0.43	0.07	0.32	0.39	0.16	0.19	0.25	0.29	0.23
	Min	0.31	-0.23	-0.36		-0.34	-0.44	-0.43	0.14	0.18	-0.10	0.38	0.05	-0.32	0.32	0.15	0.05
	Ran	0.43	**0.94**	**0.59**	0.22		0.28	0.47	0.37	0.22	0.11	0.35	0.21	0.12	0.18	-0.08	0.30
	Peak	-0.02	0.21	0.24	-0.33	-0.04		0.19	0.03	0.35	0.23	0.31	0.09	0.30	0.28	0.37	0.21
	Peak_f	0.28	-0.04	0.39	0.16	0.43	0.21		0.09	0.27	0.34	0.10	0.30	0.20	0.05	0.11	0.17
	Peak2	-0.16	-0.08	0.33	0.44	0.17	0.26	-0.32		0.16	0.31	0.15	0.09	0.29	0.12	0.23	0.28
Gyroscope	Mean	0.33	0.31	0.20	0.27	0.27	0.34	0.09	0.16		0.20	0.31	0.39	0.36	0.21	-0.06	-0.18
	Var	0.18	0.35	0.18	0.05	0.35	0.05	0.19	0.12	0.31		-0.15	0.04	**0.95**	0.08	0.34	-0.51
	Max	0.32	0.34	0.36	0.30	-0.17	0.33	0.22	0.17	-0.27	0.21		0.37	**0.68**	0.42	-0.27	0.38
	Min	0.32	0.16	0.18	0.04	0.29	0.24	0.12	0.13	-0.14	0.47	0.37		0.34	0.18	0.15	-0.34
	Ran	0.13	0.04	-0.19	0.11	0.29	0.09	-0.24	0.19	0.03	**0.89**	**0.60**	-0.14		-0.21	0.12	0.23
	Peak	0.07	0.15	-0.33	0.18	0.35	0.06	0.33	0.30	0.25	-0.18	0.41	0.02	-0.38		0.33	0.16
	Peak_f	0.21	0.17	0.23	0.26	-0.13	0.13	0.36	0.16	0.32	0.36	-0.20	0.32	0.24	0.18		0.07
	Peak2	0.33	0.07	0.30	0.16	0.32	0.21	0.35	0.15	-0.29	0.12	-0.10	0.39	0.34	0.12	-0.19	



(a) Smartphone



(b) Smartwatch

Fig. 3. KS test on sensor features.

A. Experimental settings

We perform different types of experiments with 35 participants, using Nexus 5 smartphones and Moto 360 smartwatches. We recorded the demographics (gender, and age range) of the participants and show them in Figure 2. We collected sensor data from different sensors in the smartphone and the smartwatch, with a sampling rate of 50 Hz. The different types of experiments (free-form usage, lab experiments and attacker usage) will be discussed in detail in the sub-sections they are used, as we attempt to answer the above questions on the design parameters of our implicit authentication system. All experimental results in the following sub-sections are based on the free-form use of the smartphone and smartwatch for two weeks, except the experiments for context detection (where lab conditions are used) and the masquerading attacks (where attacker usage is imitated). Free-form usage means the users can use the devices with no restrictions, as they normally would in their daily lives.

In our collected data for the machine learning algorithms, we used 10-fold cross-validation to generate the training data and testing data sets for evaluating the authentication performance, i.e., $9/10$ data would be used as the training data and the remaining $1/10$ is used as the testing data. To extensively investigate the performance of our system, we repeated such cross-validating mechanisms for 1000 iterations and averaged the experimental results.

We also discuss the complexity of our system and the impact on the battery drainage (Section V-H). Finally, we discuss re-training authentication models (Section V-I) due to users' behavioral drift.

B. Which sensors to use?

Mobile sensing technology has matured to a state where collecting many measurements through sensors in smartphones is now becoming quite easy through, for example, Android sensor APIs. Mobile sensing applications, such as the CMU MobiSens[36], run as a service in the background and can constantly collect sensors' information from smartphones. Sensors can be either hard sensors (e.g., accelerometers) that are physically-sensing devices, or soft sensors that record information of a phone's running status (e.g., screen on/off). Thus, practical sensors-based user authentication can be achieved today. But which sensors should we select?

We use Fisher scores (FS)[37] to help select the most promising sensors for user authentication. FS is one of the most widely used supervised feature selection methods due to its excellent performance. The Fisher Score enables finding a subset of features, such that in the data space spanned by the selected features, the distances between data points in different classes are as large as possible, while the distances between data points in the same class are as small as possible. Table II shows the FS for different sensors that are widely implemented in smartphones and smartwatches. We found that the magnetometer, orientation sensor and light sensor have

lower FS because they are influenced by the environment. This can introduce various background noise unrelated to the user's behavioral characteristics, e.g., the magnetometer may be influenced by magnets. Therefore, we select two sensors, the accelerometer and gyroscope, because they have higher FS and furthermore, are the most common sensors built into current smartphones and smartwatches [38].

These two sensors also represent different information about the user's behavior: 1) the accelerometer records coarse-grained motion patterns of a user, such as how she walks [16]; and 2) the gyroscope records fine-grained motions of a user such as how she holds a smartphone [13]. Furthermore, these sensors do not need the user's permissions, making them useful for continuous background monitoring in implicit authentication scenarios.

C. What sensor features are best?

Using the raw sensor data streams from the selected sensors may not be as good as using statistical features derived from these raw sensor data streams. Hence, we segment the sensor data streams into a series of time windows, and compute statistics from both the time domain and the frequency domain for the sensor data values in a time window. The magnitude of sensor i's data stream in the k-th window is denoted $S_i(k)$. For example, the magnitude of an accelerometer data sample (t, x, y, z) is computed as $m = \sqrt{x^2 + y^2 + z^2}$. We implement the Discrete *Fourier* transform (DFT) [39] to obtain the frequency domain information. The frequency domain information is useful and is widely used in signal processing and data analysis, e.g., speech signals and images.

We compute the following statistical features derived from each of the raw sensor streams, in each time window:

- Mean: Average value of the sensor stream
- Var: Variance of the sensor stream
- Max: Maximum value of the sensor stream
- Min: Minimum value of the sensor stream
- Ran: Range of the sensor stream
- Peak: The amplitude of the main frequency of the sensor stream
- Peak_f: The main frequency of the sensor stream
- Peak2: The amplitude of the secondary frequency of the sensor stream
- Peak2_f: The secondary frequency of the sensor stream

We then test the performance of each feature and drop "bad" features. If a feature can be used to easily distinguish two users, we say the feature is a good feature. For a feature to distinguish two different persons, it is necessary for the two underlying distributions to be different. Hence, for each feature, we test whether this feature derived from different users is from the same distribution. If most pairs of them are from the same distribution, the feature is "bad" in distinguishing two persons and we drop it.

We use the Kolmogorov-Smirnov test (KS test) [40] to test if two data sets are significantly different. The KS test is a nonparametric statistical hypothesis test based on the maximum distance between the empirical cumulative distribution functions of the two data sets. The two hypotheses of a KS test are:

H_0*: the two data sets are from the same distribution*

H_1*: the two data sets are from different distributions.*

A KS test reports a p-value, i.e. the probability that obtaining the maximum distance is at least as large as the observed one when H_0 is assumed to be true. i.e., H_0 is accepted. If this p-value is smaller than α, usually set to 0.05, we will reject the H_0 hypothesis because events with small probabilities rarely happen (rejecting H_0 and accepting H_1 indicates a "good" feature for distinguishing users). For each feature, we calculate the p-value for data points for each pair of users and drop a feature if most of its p-values are higher than α.

Figure 3 shows the testing results for the features in both the smartphone and smartwatch. For each feature, the resulting p-values are drawn in a box plot. The bottom and the top lines of the box denote the lower quartile Q_1 and upper quartile Q_2, defined as the 25th and the 75th percentiles of the p-values. The middle bar denotes the median of the p-values. The y-axes in Figure 3 is in logarithmic scale. The red horizontal lines represent the significance level $\alpha = 0.05$. The better a feature is, the more of its box plot is below the red line. It denotes that more pairs are significantly different. From Figure 3, we find that the accPeak2_f and gyrPeak2_f in both the smartphone and the smartwatch are "bad" features, so we drop them.

Next, we try to drop redundant features, by computing the correlation between each pair of features. A strong correlation between a pair of features indicates that they are similar in describing a user's behavior pattern, so one of the features can be dropped. A weak correlation implies that the selected features reflect different behaviors of the user, so both features should be kept.

We calculated the Pearson's correlation coefficient between any pair of features. Then, for every pair of features, we took the average of all resulting correlation coefficients over all the users. Table III shows the resulting average correlation coefficients. The upper right triangle is the correlation between features in the smartphone, while the lower left triangle is the correlation between features in the smartwatch. We observe that Ran has very high correlation with Var in each sensor on both the smartphone and smartwatch. It means that Ran and Var have information redundancy. Also Ran has relatively high correlation with Max. Therefore, we drop Ran from our feature set.

D. Do multiple devices help?

We also study if using data from the same type of sensors (accelerometer and gyroscope), but from different devices is helpful for improving user authentication. Towards this end, we calculate the correlations between smartphone and smartwatch sensor data in Table IV. Since these features do not have strong correlation with each other, it implies that these same sensors on the two devices measure different aspects of a user's behavior, so we keep all these features.

Hence our feature vector for sensor i, in a given time window k, for the smartphone, SP, is

TABLE IV
CORRELATIONS BETWEEN SMARTPHONE AND SMARTWATCH. ROW LABELS ARE THE FEATURES FROM SMARTWATCH AND COLUMN LABELS ARE THE FEATURES FROM SMARTPHONE.

		Smartphone Accelerometer							Smartphone Gyroscope						
		Mean	Var	Max	Min	Peak	Peak_f	Peak2	Mean	Var	Max	Min	Peak	Peak_f	Peak2
Smartwatch Accelerometer	Mean	0.08	0.33	-0.23	0.20	0.26	0.10	0.42	0.27	-0.31	-0.10	0.03	0.13	-0.19	0.06
	Var	-0.29	0.23	0.09	-0.08	-0.21	0.27	-0.24	0.04	0.39	0.26	0.05	0.17	0.15	0.37
	Max	0.35	-0.05	-0.02	-0.34	-0.15	-0.33	0.20	-0.25	0.24	0.09	0.26	-0.32	0.23	-0.22
	Min	-0.24	0.29	-0.34	0.21	-0.37	0.39	0.05	0.30	0.04	-0.33	-0.32	-0.15	-0.23	-0.13
	Peak	-0.08	-0.11	0.40	0.08	-0.07	-0.33	-0.35	-0.17	0.21	0.24	-0.29	0.08	-0.28	0.21
	Peak_f	0.11	-0.21	0.03	-0.10	0.33	0.07	0.34	-0.22	-0.18	0.04	0.32	-0.07	-0.12	-0.31
	Peak2	-0.26	-0.16	-0.08	0.14	-0.32	-0.26	0.24	0.24	-0.24	0.41	0.15	-0.37	-0.12	-0.32
Smartwatch Gyroscope	Mean	0.02	0.13	-0.16	0.08	0.36	0.37	-0.26	-0.31	0.20	-0.31	0.33	0.37	-0.24	0.26
	Var	0.16	0.29	-0.33	-0.26	0.03	-0.30	-0.10	-0.26	0.03	0.05	0.02	-0.29	0.27	0.21
	Max	-0.12	-0.30	0.22	0.21	-0.14	-0.20	-0.03	0.10	0.12	0.05	0.31	0.30	0.32	-0.28
	Min	0.07	-0.22	-0.18	0.19	-0.29	0.30	0.11	0.15	0.06	0.29	-0.33	-0.11	-0.04	-0.13
	Peak	0.28	-0.21	-0.27	0.34	0.37	0.16	0.23	0.29	0.20	0.04	0.14	0.19	-0.10	-0.05
	Peak_f	-0.23	-0.06	-0.25	0.29	0.33	0.18	0.28	-0.16	0.25	-0.32	0.20	-0.04	-0.06	0.12
	Peak2	0.13	-0.07	0.21	-0.27	0.37	0.32	-0.11	0.38	-0.12	-0.22	0.06	0.04	0.33	0.11

$$SP_i(k) = [SP_i^t(k), SP_i^f(k)] \tag{1}$$

where t represents the time domain, f represents the frequency domain, and

$$\begin{aligned} SP_i^t(k) &= [mean(S_i(k)), var(S_i(k)), max(S_i(k)), min(S_i(k))] \\ SP_i^f(k) &= [peak(S_i(k)), freq(S_i(k)), peak2(S_i(k))] \end{aligned} \tag{2}$$

Therefore the feature vector for the smartphone is

$$SP(k) = [SP_{accerometer}(k), SP_{gyroscope}(k)] \tag{3}$$

Similarly, we have the the feature vector for the sensor data from the smartwatch, denoted $SW(k)$. Therefore, the authentication feature vector is

$$Authenticate(k) = [SP(k), SW(k)] \tag{4}$$

E. Can Context Detection help?

Since it seems intuitive that sensor measurements of motion may be different under different contexts, we now consider the minimum contexts that can improve the accuracy of user authentication. To be viable, we need very fast, user-agnostic context detection, since this must now precede user authentication, and we also want to keep real-time computation to an acceptable level. Hence, we try using the same feature vector in Eq. 3 for the smartphone only (no smartwatch) context detection. During the user enrollment phase, we feed these feature vectors from all users into the context detection model to train it. During the testing phase, we use this user-agnostic context detection model to detect the current user context.

1) Random Forest for context detection: We experimented with several machine learning algorithms for context detection, and chose the Random forest algorithm [41]. This is commonly used in data mining. It creates a model that predicts the value of a target variable based on several input variables.

Initially, we tried using four contexts: (1) The user uses the smartphone without moving around, e.g., while standing or sitting; (2) The user uses the smartphone while moving. No constraints are set for how the user moves; (3) The smartphone is stationary (e.g., on a table) while the user uses it; (4) The user uses the smartphone on a moving vehicle, e.g., train. However, we found that these four contexts can not be easily differentiated: contexts (3) and (4) are easily misclassified as context (1), since (1), (3) and (4) are all relatively stationary (e.g., when moving at a stable speed), compared to context (2). Therefore, we combined contexts (1), (3) and (4) into one stationary context, and left (2) as the moving context. The resulting confusion matrix in Table V showed a very high context detection accuracy of over 99% with these 2 simple contexts. The context detection time was also very short - less than 3 milliseconds.

TABLE V
CONFUSION MATRIX OF CONTEXT DETECTION RESULTS USING TWO SMARTPHONE SENSORS.

Confusion Matrix	Stationary	Moving
Stationary	99.1%	0.9%
Moving	0.6%	99.4%

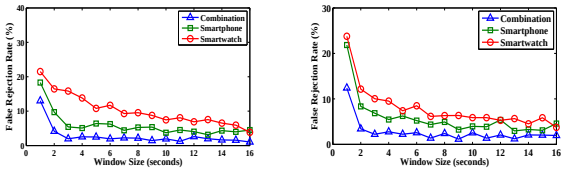
For these context training and testing experiments, we had users use their smartphones in fixed contexts under controlled lab conditions. Users were asked to use the smartphone and the smartwatch freely under each context for 20 minutes. They were told to stay in the current context until the experiment is finished. Note that such recording process is only needed for developing the context detection model and is not required for normal use in real-world scenarios. We use these data from the different users to train the context detection model in a user-agnostic manner. That is, when we perform context detection for a given user, we use a context detection model (i.e., classifier) that was trained with other users' data. This enables us to detect the context of the current user prior to authenticating her. For the Random Forest algorithm, we use 10-fold cross-validation to get the results in Table V.

F. User Authentication Algorithms

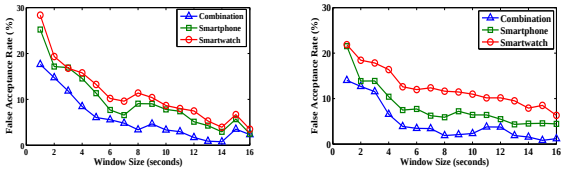
1) Features: We now ask whether such simple, fast and user-agnostic contexts (stationary versus moving) can significantly improve the accuracy of user authentication? If so, to what extent? For this, we did different experiments, where the

TABLE VI
AUTHENTICATION PERFORMANCE WITH DIFFERENT MACHINE LEARNING ALGORITHMS.

Method	FRR	FAR	Accuracy
KRR	0.9%	2.8%	98.1%
SVM	2.7%	2.5%	97.4%
Linear Regression	12.7%	14.6%	86.3%
Naive Bayes	10.8%	13.9%	87.6%



(a) Stationary (b) Moving



(c) Stationary (d) Moving

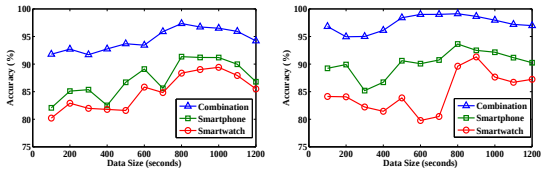
Fig. 4. FRR and FAR with different window sizes under two contexts. (a) and (b) are the FRRs under different contexts. (c) and (d) are the FARs under different contexts. Both the FRR and FAR become stable when the window size is larger than 6 seconds.

users could use their smartphones and smartwatches as they normally do in their daily lives, without any constraints on the contexts under which they used their devices. Users were invited to take our smartphone and smartwatch for one to two weeks, and use them under free-form, real-use conditions.

We evaluate the accuracy of user authentication when only the smartphone's sensor features from the accelerometer and gyroscope were used, and when both the smartphone and smartwatch's sensor features were used. The former had feature vectors with $7 \times 2 = 14$ elements, while the latter had feature vectors with $7 \times 2 \times 2 = 28$ elements.

2) Kernel Ridge Regression algorithm: Here we tried different machine learning algorithms, and found the Kernel Ridge Regression (KRR) machine learning algorithm to give the best results. Table VI shows user authentication results for a sample of state-of-the-art machine learning techniques: KRR, Support Vector Machines (SVM), linear regression, and naive Bayes. We see that KRR achieves the best accuracy. SVM also achieves high accuracy but the computational complexity is much higher than KRR (shown in Section V-H). Linear regression and naive Bayes have significantly lower accuracy compared to KRR and SVM.

Kernel ridge regressions (KRR) have been widely used for classification analysis [30], [42], [43], [44]. The advantage of KRR is that the computational complexity is much less than other machine learning methods, e.g., SVM. The goal of KRR is to learn a model that assigns the correct label to an unseen testing sample. This can be thought of as learning a function $f : X \rightarrow Y$ which maps each data x to a label y. The optimal



(a) Stationary (b) Moving

Fig. 5. Accuracy with different data sizes under the two contexts. We observe that the best accuracy happens when the data size is around 800. The accuracy decreases after the training set size is larger than 800 because a large training data set is likely to cause over-fitting in the machine learning algorithms.

classifier can be obtained analytically according to

$$\boldsymbol{w}^* = \text{argmin}_{\boldsymbol{w} \in \mathbb{R}^d} \rho \|\boldsymbol{w}\|^2 + \sum_{k=1}^{N} (\boldsymbol{w}^T \boldsymbol{x}_k - y_k)^2 \qquad (5)$$

where N is the data size and $\boldsymbol{x}_k^{M \times 1}$ represents the transpose of $Authenticate(k)$, the authentication feature vector, and M is the dimension of the authentication feature vector. Let $\boldsymbol{X}$ denote a $M \times N$ training data matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N]$. Let $\boldsymbol{y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_N]$. $\vec{\boldsymbol{\phi}}(\boldsymbol{x_i})$ denotes the kernel function, which maps the original data $\boldsymbol{x_i}$ into a higher-dimensional (J) space. In addition, we define $\boldsymbol{\Phi} = [\vec{\boldsymbol{\phi}}(\boldsymbol{x_1})\vec{\boldsymbol{\phi}}(\boldsymbol{x_2}) \cdots \vec{\boldsymbol{\phi}}(\boldsymbol{x_N})]$ and $\boldsymbol{K} = \boldsymbol{\Phi}^T \boldsymbol{\Phi}$. This objective function in Eq. 5 has an analytic optimal solution [30] where

$$\boldsymbol{w}^* = \boldsymbol{\Phi}[\boldsymbol{K} + \rho \boldsymbol{I_N}]^{-1} \boldsymbol{y} \qquad (6)$$

By utilizing certain matrix transformation properties, the computational complexity for computing the optimal $\boldsymbol{w}^*$ in Eq. 6 can be largely reduced from $O(N^{2.373})$ to $O(M^{2.373})$, which we will carefully discuss in Section V-H. This is a huge reduction since N=800 data points in our experiments, and M = 28 features in our authentication feature vector.

3) System Parameters: We need to decide on two important parameters in the system, the window size and the size of the dataset. We empirically derive the "optimal" values for these parameters.

Window Size.

The window size is an important system parameter, which determines the time that our system needs to perform an authentication, i.e., window size directly determines our system's authentication frequency.

For each context, we vary the window size from 1 second to 16 seconds. Given a window size and a detected context, for each target user, we utilize 10-fold cross-validation for training and testing. Here, we utilize the false reject rate (FRR) and false accept rate (FAR) as metrics to evaluate the authentication accuracy of our system. FRR is the fraction of the legitimate user's data that are misclassified as other users' data. FAR is the fraction of other users' data that are misclassified as the legitimate user's. For security protection, a large FAR is more harmful than a large FRR. However, a large FRR would degrade the usage convenience. Therefore, we investigate the influence of the window size on FRR and FAR, in choosing a proper window size.

TABLE VII
THE FRR,FAR AND ACCURACY UNDER TWO CONTEXTS WITH DIFFERENT DEVICES.

Context	Device	FRR	FAR	Accuracy
w/o context	Smartphone	15.4%	17.4%	83.6%
	Combination	7.3%	9.3%	91.7%
w/ context	Smartphone	5.1%	8.3%	93.3%
	Combination	0.9%	2.8%	98.1%

Figure 4 shows that the FRR and FAR for each context become stable when the window size is greater than 6 seconds. The smartphone has better (lower) FRR and FAR than the smartwatch. The combination of the smartphone and smartwatch has the lowest FRR and FAR, and achieves the best authentication performance than using each alone.

Data Size.

Another important system parameter is the size of the data set, which also affects the overall authentication accuracy because a larger training data set provides the system more information. According to our observations above, we set the window size as 6 seconds. We ranged the training set sizes, from 100 to 1200 and show the experimental results in Figure 5. We see that as the training set size increases, the accuracy first increases, approaching a maximum accuracy point, and then decreases. The maximum accuracy happens when the data size is around 800. The accuracy decreases after the training set size is larger than 800 because a large training data set is likely to cause over-fitting in the machine learning algorithms so that the constructed training model would introduce more errors than expected. Comparing the three lines in each figure, we also find that using more devices provides extra information that improves authentication accuracy.

4) User Authentication Evaluation with KRR: We now show the overall authentication performance of our system in Table VII by setting the window size as 6 seconds and the data size as 800 (from Section V-F3 results).

From Table VII, we have the following interesting observations: (1) **SmarterYou works well with just the smartphone, even without contexts**: by using only the smartphone without considering any context, our system can achieve authentication accuracy up to 83.6%. (2) **Auxiliary devices are helpful**: by combining sensor data from the smartwatch with the smartphone sensor data, the authentication performance increases significantly over that of the smartphone alone, reaching 91.7% accuracy, with better FRR and FAR. (3) **Context detection is beneficial for authentication**: the authentication accuracy is further improved, when we take the finer-grained context differences into consideration, reaching 93.3% accuracy with the smartphone alone, and 98.1% accuracy with the combination of smartphone and smartwatch data.

We also found that the overall time for implementing context detection followed by user authentication is less than 21 milliseconds. This is a fast user authentication testing time, with excellent authentication accuracy of 98%, making our system efficient and applicable in real world scenarios.

G. Masquerading attacks

Our third set of experiments was designed to analyze our system's performance in defending against some real world attacks (e.g., masquerading or mimicry attacks). We consider the worst case situation where we assume the attacker is able to monitor and record the victim's behavior. Thus the attacker can try his best to learn the victim's behavior. In these experiments, we asked each subject to be a malicious adversary whose goal was to mimic the victim user's behavior to the best of his/her ability. One user's data was recorded and his/her model was built as the legitimate user. The other users tried to mimic the legitimate user and cheat the system to let them be authenticated as the victim user. The victim user was recorded by a VCR. Subjects were asked to watch the video and mimic the behavior. Both the adversary and the legitimate user performed the same tasks, and the user's behavior is clearly visible to the adversary. Such an attack is repeated 20 times for each legitimate user and his/her 'adversaries'.

Recall that the goal of an attacker is to get access to the sensitive information stored in the smartphone, or in the cloud accessed through the smartphone. As we have shown in Figure 4 and Table VII, SmarterYou achieves very low FARs when attackers attempt to use the smartphone with their own behavioral patterns.

Now, we show that SmarterYou is even secure against the masquerading attacks where an adversary tries to mimic the user's behavior. Here, *'secure'* means that the attacker cannot cheat the system via performing these spoofing attacks and the system should detect these attacks in a short time. To evaluate this, we design a masquerading attack where the adversary not only knows the password but also observes and mimics the user's behavioral patterns. If the adversary succeeds in mimicking the user's behavioral pattern, then SmarterYou will misidentify the adversary as the legitimate user and he/she can thus use the victim user's smartphone.

In order to show the ability of SmarterYou to defend against these mimicry attacks, we counted the percentage of people (attackers) who were still using the smartphone without being de-authenticated by the system as the attack time progresses. Figure 6 shows the fraction of adversaries that are recognized as legitimate users by SmarterYou at time t, from which we can see how quickly SmarterYou can recognize an adversary and terminate his access to the smartphone. At $t = 0$, all the adversaries have access to the smartphone, but within 6s, only 10% of adversaries have access. That is, SmarterYou identified on average 90% of adversaries as unauthorized users within 6s. By $t = 18$s, SmarterYou identified all the adversaries. Therefore, SmarterYou performed well in recognizing the adversary who is launching the masquerading attack.

These experimental results also match with analysis from a theoretical point of view. We assume the FAR in each time window is p, then the chance that the attacker can escape from detection in n time windows is p^n. Based on our experimental results in Section V-F, our system can achieve 2.8% FAR in a time window of 6 seconds. Thus, within only three

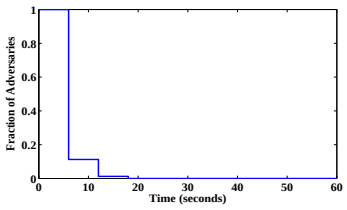


Fig. 6. Fraction of adversaries that have access to the legitimate user's smartphone at time t.

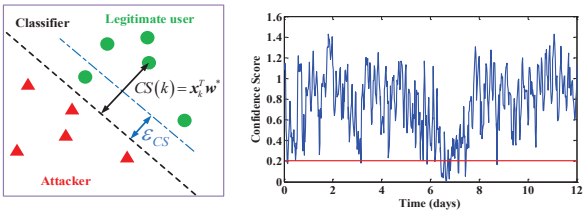


Fig. 7. The confidence score of a user with time. After around one week, the confidence score decreases below the threshold $\epsilon_{CS} = 0.2$ for a period of time. After automatic retraining, it increases back to normal values.

windows, the probability for the attacker escaping detection is $(2.8\%)^3 = 0.002\%$, which is very small. Therefore, our SmarterYou system shows good performance in defending against masquerading attacks.

H. Smartphone Overhead

We now evaluate the system overhead of SmarterYou on smartphones. Specifically, we analyze the computational complexity of our system, CPU and memory overhead, and the battery consumption it incurs on the smartphone.

1) Computational Complexity: The computational complexity of KRR in Section V-F2 is directly related to the data size according to Eq. 6. Here, we further show that the computational complexity can be largely reduced to be directly related to the feature size. (For readability, we put the detailed proof in the Appendix).

According to Eq. 6, the classifier is $\boldsymbol{w}^* = \boldsymbol{\Phi}[\boldsymbol{K} + \rho \boldsymbol{I_N}]^{-1}\boldsymbol{y}$.

Define $\boldsymbol{S} = \boldsymbol{\Phi}\boldsymbol{\Phi}^T$ ($\boldsymbol{\Phi} = [\vec{\phi}(\boldsymbol{x_1}), \vec{\phi}(\boldsymbol{x_2}), \cdots, \vec{\phi}(\boldsymbol{x_N})]$). By utilizing the matrix transformation method in [45], the optimal solution $\boldsymbol{w}^*$ in Eq. 6 is equivalent to

$$\boldsymbol{w}^* = [\boldsymbol{S} + \rho \boldsymbol{I_J}]^{-1}\Phi \boldsymbol{y} \tag{7}$$

The dominant computational complexity for $\boldsymbol{w}^*$ comes from taking the inversion of a matrix. Therefore, based on Eq. 6 and Eq. 7, the computational complexity is approximately $\min(O(N^{2.373}), O(J^{2.373}))$. If we utilize the identity kernel, the computational complexity can be reduced from $O(N^{2.373})$ to $O(M^{2.373})$ and is independent of the data size. Specifically, we construct 28-dimensional feature vectors (4 time-domain features and 3 frequency-domain features for each of two sensors, for each device).

Thus, our time complexity is reduced from $O((800 \times 9/10)^{2.373}) = O(720^{2.373})$ to only $O(28^{2.373})$. In our experiments, the average training time is 0.065 seconds and the average testing time is 18 milliseconds, which shows the effectiveness of our system applied in real-world scenarios.

2) CPU and Memory Overhead: The testing module of SmarterYou in a smartphone runs as threads inside the smartphone system process. We develop an application to monitor the average CPU and memory utilization of the phone and watch while running the SmarterYou app which continuously requests sensor data at a rate of 50 Hz on a Nexus 5 smartphone and a Moto 360 smartwatch. The CPU utilization is 5% on average and never exceeds 6%. The CPU utilization (and hence energy consumption) will scale with the sampling rate. The memory utilization is 3 MB on average. Thus, we believe that the overhead of SmarterYou is small enough to have negligible effect on overall smartphone performance.

TABLE VIII
THE POWER CONSUMPTION UNDER FOUR DIFFERENT SCENARIOS.

Scenario	Power Consumption
(1) Phone locked, SmarterYou off	2.8%
(2) Phone locked, SmarterYou on	4.9%
(3) Phone unlocked, SmarterYou off	5.2%
(4) Phone unlocked, SmarterYou on	7.6%

3) Battery Consumption: To measure the battery consumption, we consider the following four testing scenarios: (1) Phone is locked (i.e., not being used) and SmarterYou is off. (2) Phone is locked and SmarterYou keeps running. (3) Phone is under use and SmarterYou is off. (4) Phone is under use and SmarterYou is running. For scenarios (1) and (2), the test time is 12 hours each. We charge the smartphone battery to 100% and check the battery level after 12 hours. The average difference of the battery charged level from 100% is reported in Table VIII.

For scenarios (3) and (4), *the phone under use* means that the user keeps using the phone periodically. During the using time, the user keeps typing notes. The period of using and non-using is five minutes each, and the test time in total is 60 minutes.

Table VIII shows the result of our battery consumption tests, in terms of extra battery drain for SmarterYou. We find that in scenarios (1) and (2), the SmarterYou-on mode consumes 2.1% more battery power than the SmarterYou-off mode. We believe the extra cost in battery consumption caused by SmaterYou will not affect user experience in daily use. For scenarios (3) and (4), SmarterYou consumes 2.4% more battery power in one hour, which is also an acceptable cost for daily usage.

I. Retraining Authentication Models

The behavioral drift of the legitimate user must be considered. The user may change his/her behavioral pattern over weeks or months, which may cause more false alarms in implicit authentication. SmarterYou, therefore, will retrain the authentication models automatically and continuously based on the previous authentication performance. Here, we define the confidence score (CS) as $CS(k) = \boldsymbol{x}_k^T \boldsymbol{w}^*$ for the k-th authentication feature vector $\mathbf{x}_k^T$ as the distance between $\mathbf{x}_k^T$ and the corresponding authentication classifier $\boldsymbol{w}^*$.

As the authentication classifier $\boldsymbol{w}^*$ represents the classification boundary to distinguish the legitimate user and the adversaries, a lower confidence score (smaller distance between $\mathbf{x}_k^T$ and $\boldsymbol{w}^*$) represents a less confident authentication result (shown conceptually in the left figure of Figure 7). This suggests a change of user's behavioral pattern where retraining should be taken. For an authenticated user, we suggest that if the confidence score is lower than a certain threshold ϵ_{CS} for a period of time T, the system automatically retrains the authentication models.

In Figure 7 (right figure), we show the confidence score of the time-series authentication feature vectors for a user. We can see that the confidence score decreases slowly in the first week. At the end of the first week, the confidence score experiences a period of low values (lower than our threshold $\epsilon_{CS} = 0.2$ for a period), indicating that the user's behavior changes to some extent during this week. Therefore, it would be helpful if the system can automatically retrain the authentication models. Note that there are some earlier points lower than the threshold (0.2), but they do not occur for a long enough period to trigger the retraining. Also, it is hard for the attacker to trigger the retraining because the probability that the attacker continuously passes the authentication for a long period of time is low as described in Section V-G.

As our system recognizes user's behavior drift by checking the confidence score, it would then go back to the training module again and upload the legitimate user's authentication feature vectors to the training module until the new behavior (authentication model) is learned. Advanced approaches in machine unlearning [46] can be explored to update the authentication models asymptotically faster than retraining from scratch. After retraining the user's authentication models, we can see that the confidence score increases to normal values from Day 8.

As discussed earlier, an attacker who has taken over a legitimate user's smartphone must not be allowed to retrain the authentication model. Fortunately, the attacker can not trigger the retraining since the confidence score should be positive and last for a period of time. However, the attacker is likely to have negative confidence scores, which cannot last for sufficient time to trigger retraining, since he will be detected in less than 18 seconds by SmarterYou, according to Figure 6.

VI. Conclusions

We have proposed a new re-authentication system, SmarterYou, to improve the security of a smartphone, and of secret and sensitive data and code in the smartphone or in the cloud accessible through a smartphone. SmarterYou is an authentication system using multiple sensors built into a user's smartphone, supplemented by auxiliary information from a wearable device, e.g., smartwatch, with the same owner as the smartphone. Our system keeps monitoring the users' sensor data and continuously authenticates without any human cooperation. We first collect context features from the sensors' data in the smartphone (and the smartwatch if present) to detect the context of the current user. Based on the detected context and the authentication features in both the time and frequency domains, our system implements finer-grained authentication efficiently and stealthily.

We systematically evaluate design alternatives for each design parameter of such a sensor-based implicit authentication system. Based on our design choices, our evaluations demonstrate the advantage of combining the smartphone and the smartwatch and the enhancement in authentication accuracy with context detection and time-frequency information. SmarterYou can achieve authentication accuracy up to 98.1% (FRR 0.9% and FAR 2.8%) with negligible system overhead and less than 2.4% additional battery consumption. We believe this is the highest accuracy and lowest FAR reported by any sensor-based authentication method to date. We hope that the SmarterYou system and design techniques can help advance the field in implicit user authentication and re-authentication, for deployment in real-world scenarios.

References

[1] Y. Kim, T. Oh, and J. Kim, "Analyzing user awareness of privacy data leak in mobile applications," *Mobile Information Systems*, 2015.

[2] J. Achara, C. Castelluccia, J.-D. Lefruit, V. Roca, F. Baudot, and G. Delcroix, "Mobilitics: Analyzing privacy leaks in smartphones," *ERCIM Newsletter*, 2013.

[3] M. Qi, Y. Lu, J. Li, X. Li, and J. Kong, "User-specific iris authentication based on feature selection," in *CSSE*, 2008.

[4] K. Xi, J. Hu, and F. Han, "Mobile device access control: an improved correlation based face authentication scheme and its java me application," *Concurrency and Computation: Practice and Experience*, 2012.

[5] K. Niinuma, U. Park, and A. K. Jain, "Soft biometric traits for continuous user authentication," *IEEE TIFS*, 2010.

[6] ConsumerReports, "Keep your phone safe: How to protect yourself from wireless threats," *Consumer Reports, Tech.*, 2013.

[7] A. De Luca, A. Hang, F. Brudy, C. Lindner, and H. Hussmann, "Touch me once and i know it's you!: implicit authentication based on touch screen patterns," in *ACM CHI*, 2012.

[8] N. L. Clarke and S. M. Furnell, "Authenticating mobile phone users using keystroke analysis," *International Journal of Information Security*, vol. 6, no. 1, pp. 1–14, 2007.

[9] S. Buthpitiya, Y. Zhang, A. K. Dey, and M. Griss, "n-gram geo-trace modeling," in *Pervasive Computing*, 2011.

[10] O. Riva, C. Qin, K. Strauss, and D. Lymberopoulos, "Progressive authentication: Deciding when to authenticate on mobile phones." in *USENIX Security*, 2012.

[11] J. Zhu, P. Wu, X. Wang, and J. Zhang, "Sensec: Mobile security through passive sensing," in *ICNC*, 2013.

[12] J. Mäntyjärvi, M. Lindholm, E. Vildjiounaite, S.-M. Mäkelä, and H. Ailisto, "Identifying users of portable devices from gait pattern with accelerometers," in *ICASSP*, 2005.

[13] Z. Xu, K. Bai, and S. Zhu, "Taplogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors," in *conference on Security and Privacy in Wireless and Mobile Networks*, 2012.

[14] A. J. Aviv, K. L. Gibson, E. Mossop, M. Blaze, and J. M. Smith, "Smudge attacks on smartphone touch screens." *Woot*, 2010.

[15] M. Conti, I. Zachia-Zlatea, and B. Crispo, "Mind how you answer me!: transparently authenticating the user of a smartphone when answering or placing a call," in *CCS*, 2011.

[16] C. Nickel, T. Wirtl, and C. Busch, "Authentication of smartphone users based on the way they walk using k-nn algorithm," in *IIH-MSP*, 2012.

[17] M. Trojahn and F. Ortmeier, "Toward mobile authentication with keystroke dynamics on mobile phones and tablets," in *WAINA*, 2013.

[18] F. Okumura, A. Kubota, Y. Hatori, K. Matsuo, M. Hashimoto, and A. Koike, "A study on biometric authentication based on arm sweep action with acceleration sensor," in *ISPACS*, 2006.

[19] M. Frank, R. Biedert, E.-D. Ma, I. Martinovic, and D. Song, "Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication," *IEEE TIFS*, 2013.

[20] L. Li, X. Zhao, and G. Xue, "Unobservable re-authentication for smartphones," in *NDSS*, 2013.
[21] T. Feng, Z. Liu, K.-A. Kwon, W. Shi, B. Carbunar, Y. Jiang, and N. K. Nguyen, "Continuous mobile authentication using touchscreen gestures," in *Homeland Security, Conference on Technologies for*, 2012.
[22] H. Xu, Y. Zhou, and M. R. Lyu, "Towards continuous and passive authentication via touch biometrics: An experimental study on smartphones," in *Symposium On Usable Privacy and Security*, 2014.
[23] N. Zheng, K. Bai, H. Huang, and H. Wang, "You are how you touch: User verification on smartphones via tapping behaviors," in *Network Protocols, International Conference on*, 2014.
[24] H. G. Kayacık, M. Just, L. Baillie, D. Aspinall, and N. Micallef, "Data driven authentication: On the effectiveness of user behaviour modelling with mobile device sensors," *Mobile Security Technologies*, 2014.
[25] W.-H. Lee and R. B. Lee, "Multi-sensor authentication to improve smartphone security," in *ICISSP*, 2015.
[26] L. Yang, Y. Guo, X. Ding, J. Han, Y. Liu, C. Wang, and C. Hu, "Unlocking smart phone through handwaving biometrics," *IEEE Transactions on Mobile Computing*, 2015.
[27] L. Hong and A. Jain, "Integrating faces and fingerprints for personal identification," *IEEE TPAMI*, 1998.
[28] A. Serwadda and V. V. Phoha, "When kids' toys breach mobile phone security," in *CCS*, 2013.
[29] S. Mare, A. M. Markham, C. Cornelius, R. Peterson, and D. Kotz, "Zebra: Zero-effort bilateral recurring authentication," in *SP*, 2014.
[30] J. A. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, J. Vandewalle, J. Suykens, and T. Van Gestel, *Least squares support vector machines*. World Scientific, 2002.
[31] W.-H. Lee, X. Liu, Y. Shen, H. Jin, and R. Lee, "Secure pick up: Implicit authentication when you start using the smartphone," in *Symposium on Access Control Models and Technologies*, 2017.
[32] T. Y.-H. Chen, A. Sivaraman, S. Das, L. Ravindranath, and H. Balakrishnan, "Designing a context-sensitive context detection service for mobile devices," 2015.
[33] N. Kern, B. Schiele, and A. Schmidt, "Multi-sensor activity context detection for wearable computing," in *Ambient Intelligence*. Springer, 2003.
[34] ARM, "Security technology - building a secure system using trustzone technology," *ARM Technical White Paper*, 2009.
[35] F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar, "Innovative instructions and software model for isolated execution," in *International Workshop on Hardware and Architectural Support for Security and Privacy*, 2013.
[36] P. Wu, J. Zhu, and J. Y. Zhang, "Mobisens: A versatile mobile sensing platform for real-world applications," *Mobile Networks and Applications*, 2013.
[37] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. John Wiley & Sons, 2012.
[38] Google, "Android sensor manager," http://developer.android.com/guide/topics/sensors/sensors_overview.html.
[39] B. Boashash, *Time frequency analysis*. GPP, 2003.
[40] W. W. Daniel *et al.*, "Applied nonparametric statistics," 1990.
[41] T. K. Ho, "Random decision forests," in *ICDAR*. IEEE, 1995.
[42] S. An, W. Liu, and S. Venkatesh, "Face recognition using kernel ridge regression," in *CVPR*, 2007.
[43] W.-H. Lee and R. Lee, "Implicit authentication for smartphone security," in *Information Systems Security and Privacy*. Springer, 2015.
[44] ——, "Implicit sensor-based authentication of smartphone users with smartwatch," in *HASP 2016*, 2016.
[45] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 2012.
[46] Y. Cao and J. Yang, "Towards making systems forget with machine unlearning," in *Security and Privacy*, 2015.

VII. Appendix

A. Proof of Equivalence between Eq. 6 and Eq. 7

Eq. 6 is $\boldsymbol{w}^* = \boldsymbol{\Phi}[\boldsymbol{K} + \rho \boldsymbol{I_N}]^{-1}\boldsymbol{y}$, and Eq. 7 is $\boldsymbol{w}^* = [\boldsymbol{S} + \rho \boldsymbol{I_J}]^{-1}\Phi\boldsymbol{y}$. In order to prove that they are equivalent, we first prove $\boldsymbol{PB^T(BPB^T} + \rho\boldsymbol{R})^{-1} = (\rho\boldsymbol{P^{-1}} + \boldsymbol{B^TR^{-1}B})^{-1}\boldsymbol{B^TR^{-1}}$ as follows:

$$\begin{aligned} &\rho\boldsymbol{B^T} + \boldsymbol{B^TR^{-1}BPB^T} = \boldsymbol{B^TR^{-1}BPB^T} + \rho\boldsymbol{B^T} \\ &\Leftrightarrow (\rho\boldsymbol{P^{-1}} + \boldsymbol{B^TR^{-1}B})\boldsymbol{PB^T} = \boldsymbol{B^TR^{-1}}(\boldsymbol{BPB^T} + \rho\boldsymbol{R}) \\ &\Leftrightarrow \boldsymbol{PB^T}(\boldsymbol{BPB^T} + \rho\boldsymbol{R})^{-1} = (\rho\boldsymbol{P^{-1}} + \boldsymbol{B^TR^{-1}B})^{-1}\boldsymbol{B^TR^{-1}} \end{aligned} \tag{8}$$

Then we let $\boldsymbol{P} = \boldsymbol{I_J}$, $\boldsymbol{B} = \boldsymbol{\Phi}^T$ and $\boldsymbol{R} = \boldsymbol{I_N}$ in Eq. 8, we observe the left hand side of Eq. 8 is Eq. 6 and the right hand side of Eq. 8 is Eq. 7. Thus, we prove the equivalence between Eq. 6 and Eq. 7.