5-2019

User Behavior-Based Implicit Authentication

Yingyuan Yang
University of Tennessee, yyang57@vols.utk.edu

User Behavior-Based Implicit Authentication

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Yingyuan Yang

May 2018

To my wife Xueli Huang, for love and prayers...

To my mother Xiaowei Wang, for making all my accomplishments possible...

To my father Lujia Yang, for guidance and teaching...

To my daughter Caroline Yang, for love and hope...

Acknowledgments

I would like to thank Professor Jinyuan Sun, my advisor, for giving me the ability to accomplish this. Her unconditional support and insightful guidance were invaluable during the course of my doctorate study. I am also indebted to Professor Hairong Qi, Professor Max Schuchard, and Professor Lee D. Han for their time and patience to read my dissertation, to attend my defense, and to provide meticulous comments that tremendously improved the quality of this work.

I gratefully appreciate the financial support from NSF and Department of EECS of the University of Tennessee.

I also would like to thank my friends and classmates: Jiangnan Li, Eric Reinsmidt, Zhongbo Li, Joey Allen, Yue Tong, Xiangyu Niu, and Zhuo Yao for their tremendous help and support.

Abstract

In this work, we proposed dynamic retraining (RU), wind vane module (WVM), BubbleMap (BMap), and reinforcement authentication (RA) to improve the efficacy of implicit authentication (IA). Motivated by the great potential of implicit and seamless user authentication, we have built an implicit authentication system with adaptive sampling that automatically selects dynamic sets of activities for user behavior extraction. Various activities, such as user location, application usage, user motion, and battery usage have been popular choices to generate behaviors, the soft biometrics, for implicit authentication. Unlike password-based or hard biometric-based authentication, implicit authentication does not require explicit user action or expensive hardware. However, user behaviors can change unpredictably, which renders it more challenging to develop systems that depend on them. In addition to dynamic behavior extraction, the proposed implicit authentication system differs from the existing systems in terms of energy efficiency for battery-powered mobile devices. Since implicit authentication systems rely on machine learning, the expensive training process needs to be outsourced to the remote server. However, mobile devices may not always have reliable network connections to send real-time data to the server for training. In addition, IA systems are still at their infancy and exhibit many limitations, one of which is how to determine the best retraining frequency when updating the user behavior model. Another limitation is how to gracefully degrade user privilege when authentication fails to identify legitimate users (i.e., false negatives) for a practical IA system.

To address the retraining problem, we proposed an algorithm that utilizes Jensen-Shannon (JS)-dis(tance) to determine the optimal retraining frequency, which is discussed in Chapter 2. We overcame the limitation of traditional IA by proposing a W-layer, an overlay that provides a practical and energy-efficient solution for implicit authentication on mobile

devices. The W-layer is discussed in Chapter 3 and 4. In Chapter 5, a novel privilege-control mechanism, BubbleMap (BMap), is introduced to provide fine-grained privileges to users based on their behavioral scores. In the same chapter, we describe reinforcement authentication (RA) to achieve a more reliable authentication.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

As its name suggests, implicit authentication, or IA, is a technique that allows the smart device to recognize its owner by being acquainted with his/her behaviors. It is a technique that uses machine learning algorithms to learn user behavior through various sensors on the smart device and achieve user identification [104, 53]. Most of the current authentication techniques, e.g., password, pattern lock, fingerprint, and iris recognition, are explicit authentication, which requires user input. Compared to explicit authentication, IA is transparent to users during the usage, and it significantly increases the usability by reducing time users spent on login, in which users find it more annoying than lack of cellular coverage [4, 10, 107].

The rapid growth of mobile devices, especially smartphones, raises security concerns. One such concern is that traditional authentication methods, such as passwords or hard biometrics can be potentially circumvented [3]. A recent survey [4] showed that only 44% of smartphone owners configured PINs or passwords on their devices due to the inconvenience. Another survey [49] showed that 56% of participants mistyped a password at least one time out of every ten tries. The passwords can also be guessed and broken. Users need to constantly change the passwords to maintain their effectiveness, which further decreases the usability of the devices.

On the other hand, hard biometric-based authentication, such as face recognition, touch ID and iris scan require expensive hardware and explicit user interaction. Even though biometrics are much harder to be stolen than passwords, there have been works showing the feasibility of hard biometrics forgery [103]. In addition, the environmental variation may also cause the false identification of hard biometric-based authentication [109, 40].

The disadvantages of the traditional authentication methods have inspired researchers to develop new authentication methods that are transparent to users and incur a low cost. One promising method is implicit authentication (IA) based on user behaviors, which can be considered soft biometrics. Various activities, such as arm swing, walking, and contextual data (e.g., location) [97, 94, 29] are commonly used to derive soft biometrics. Compared with password-based and hard biometric-based authentication, IA does not require explicit user actions since it runs in the background and silently obtains user behaviors. Moreover, unlike hard biometrics, soft biometrics in IA only require basic hardware and sensors readily available on smartphones.

To complement primary authentication mechanisms such as PINs and passwords, many implicit authentication schemes have been proposed as secondary authentication mechanisms [23, 66, 92, 27, 32, 88, 62, 35, 19]. Among them, leveraging different features, Shi scheme [88], Multi-Sensor scheme [62], Gait scheme [35], and SilentSense scheme [19] are four different schemes that represent four research directions of the state-of-the-art implicit authentications [52]. In addition, current implicit authentication research tends to adopt all the available features to achieve better authentication accuracy [62, 105, 104]. To evaluate the performance of proposed methods, we implemented Shi scheme, Multi-Sensor scheme, Gait scheme, and SilentSense scheme. We also show the proposed methods can seamlessly cooperate with another framework such as [104, 105] to improve the system's performance.

Human behaviors are susceptible to change, e.g., a change of routine, which causes the existing implicit authentication (IA) mechanism to constantly fail to identify users. To deal with this problem, the state-of-the-art technique used in implicit authentication is to frequently retrain the model. However, it is difficult to decide an optimal retraining rate given the complex human behavior. In addition, the system needs to separate the behavior change of the legitimate user from malicious mimicry attack of the illegitimate user, which was an unsolved problem since the data samples of users often share common intersections. In other words, if the retraining frequency is too high, we could waste a lot of energy and computational resources in order to obtain a high authentication accuracy. If it is too low, the accuracy would be affected, which could cause high false positive and false negative rate in the authentication.

In addition to the retraining, existing authentication systems handle failed authentication attempts by locking the users out of their mobile devices. It is unsuitable for implicit authentication whose accuracy deterioration induces high false reject rate, rendering the implicit authentication system unusable. Furthermore, existing implicit authentication systems leverage computationally expensive machine learning algorithms, which can introduce a large authentication delay. It is difficult to improve the authentication accuracy of these systems without sacrificing authentication time. The improvement, on the other hand, should be resilient enough to absorb the impact of users' behavior change without reducing authentication transparency. In addition, the model used in the authentication needs to consider another fact such as mimicry attack, which significantly increases the system's design difficulty.

In the following sections, we will analyze some of the most important problems in IA research, and the corresponding techniques proposed to deal with them. We will also discuss some fundamental problems that currently have no effective solution. These problems could become new research directions in the future. Centered on the improvement of authentication accuracy, the existing research works in IA can be categorized into two classes, scheme (or structure) improvement and behavioral feature analysis. The following sections are arranged based on these two classes.

1.1 Basic Scheme

Since most of the existing IA schemes are developed based on the basic scheme, we discuss it first in this section. The corresponding improvements will be covered in the next section.

The basic IA scheme is shown in Fig. 1.1, which was first proposed in [22]. As shown in Fig. 1.1, the basic IA scheme has two phases - training and testing. In the training phase, past behavioral data is input as parameters in the training algorithm. The training result - i.e., a model with tuned parameters - is then returned for the testing purpose. In the testing phase, which usually happens in real time, recent behavioral data is input into the model, and a score is returned to either reject or to allow user access.

In the training phase, several different machine learning tools have been applied to analyze the users' behavioral data [44, 6, 30], e.g., SVM [62, 32], GMM [87] and statistic topic model [105]. The training phase usually takes more than a week to find the fine-tuned parameters for the model using a cluster of fast computers. The testing phase is more light-weight compared to the training phase, in which a complete testing cycle usually takes a few seconds in a computation limited device, such as the smartphone. Thus the current IA framework often offloads the training phase to the server or cloud. The user-end only contains a fine-tuned model returned from the training phase. If we assume the human behaviors do not deviate after the training phase, the structure may achieve a high accuracy during the authentication.

However, due to the behavior deviation, e.g., traveling, the accuracy of the fine-tuned model may also decay during the usage. Several more sophisticated structures are introduced to improve the basic scheme [53, 104, 54]. Several other works focus on the selection of machine learning algorithms and features, which may also improve the system's accuracy [94, 62, 32, 36, 37, 61, 81, 41].

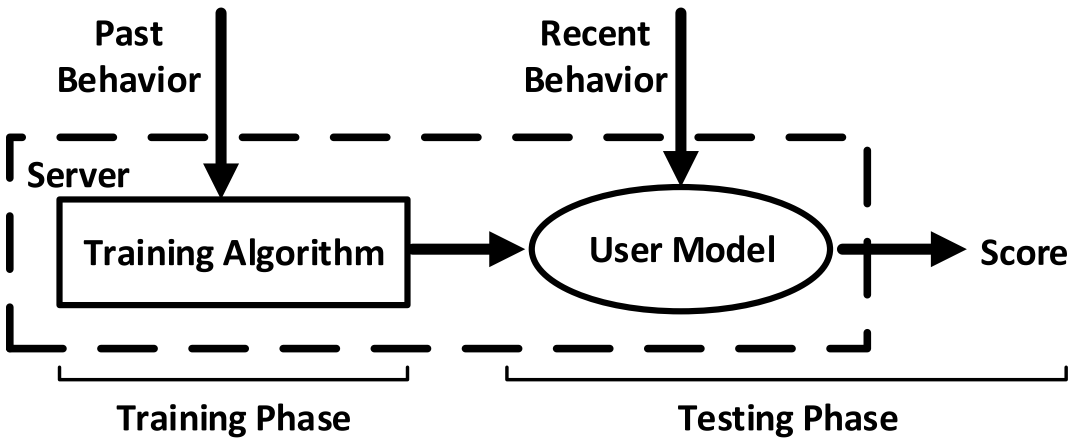


Figure 1.1: Overview

1.2 The Enhancement of Basic Scheme

Current IA research tends to improve the basic scheme as shown in the Fig. 1.1. The improved schemes are designed to solve different problems, which can be used in different scenarios in practice. It is possible to combine these schemes into one comprehensive structure that supports most of the requirements in real usage. However, we will not discuss the combination of different schemes, which is beyond the scope of the proposal thesis. Instead, we will discuss several important improvements of the basic scheme in this section.

Applications in most of the smart devices are running in their own environment called sandbox, which is supported by a virtual machine. An application running in its sandbox is insulated from the other applications. Instead of designing a system-level IA [53], Khan et al. tried to build a scheme that can be used at each application individually. Their scheme is built into the Android framework and provides some basic functions to the application developers. The function within the scheme is implemented in a library at the user level and can be used right away without the need to root any devices or be added to the OS, although it could become part of mobile platforms in the future. They also provide some basic machine learning algorithm, e.g., kNN and SVM, for developers to facilitate their design of IA scheme.

To reduce the impact of behavior deviation, one of the most common approaches is to retrain the model. After retraining the new model will contain the most-recent behaviors of the legitimate user. However, in practice, it is difficult for the IA system to decide the frequency of retraining since the current IA system does not have a user-feedback mechanism. The other problems regarding the retraining are new behavioral data selecting and noise filtering. Focusing on these problems, several research works introduce new techniques, which constantly detect the behavior deviation, filter out the noise and select the data in the retraining process [106, 39, 110, 18]. This research significantly enhanced the basic IA scheme, but the retraining problem in real usage is more sophisticated, which is still an open problem in IA.

The training phase in IA usually takes place in a remote server or the cloud. The communication between the server and local devices increases energy consumption and raises some security issues. In some conditions, the wireless connection may lose and reduce the accuracy of IA. The local IA with no server-end could overcome these issues, but it requires researchers to deal with the energy and computation issue caused by the smart device. A lightweight IA scheme has been introduced to authenticate a user in an energy-efficient way [104], in which the training phase can be migrated to a local device when the server is not available.

1.3 Behavioral Data Analysis

The machine learning algorithm is the core part of implicit authentication (IA). Due to the large diversity of users' data, it is hard to find one machine learning algorithm that is suitable for all the behavioral data sampled from various sensors. To best identify the user, different machine learning algorithms, e.g., SVM, kNN and GMM, have been selected to analyze various features pre-selected by researchers. In addition, the way of applying these algorithms also has a large impact on final identification accuracy.

Currently, most of the IA systems utilize Support Vector Machine (SVM) as the core machine learning algorithm, such as [36, 63, 19, 62, 18, 106, 53, 39, 9]. The SVM has been used to analyze the key stroke, gait, finger tip movement, and gait patterns. Some of the research achieved an impressive result, e.g., in [36]; they achieve a median equal error rate of 0% for intra-session authentication, 2%-3% for inter-session authentication and below 4% when the authentication test was carried out one week after the enrollment phase. In [19] they conduct extensive evaluations of their proposed approaches on the Android smartphone. In addition, they show that user identification accuracy is over 99%.

The other trend of current IA research focuses on the users' behavioral features selection or feature selection. A well-selected feature can achieve high accuracy during the authentication cycle [56, 52, 93, 27]. In [32], Fen et al. select a touch-related feature in uncontrolled environments and achieve over 90% accuracy in real-life naturalistic conditions within a small amount of computational overhead and 6% of battery usage. However, due to the large divergence of human behaviors, it is very hard to find one feature that separates all the users. The problem of feature selection becomes more complex if we consider human behavior deviation and mimicry attack in real usage as shown in [56, 52]. The problem of searching for a good feature in the authentication cycle is still an open problem in current IA research [56, 52].

1.4 Objectives and Scope

The major goal of this research is to perform a comprehensive evaluation of the deployability, security and usability-related issues of IA. Based on the findings of these evaluations, we also propose strategies to make IA more practical so that it can be adopted. We organize the remaining chapters as follows:

Chapter 2 mainly focuses on training and retraining problems in IA. With the rapid growth of the smart device market, associated security issues become more threatening and diverse than ever before. Due to the limitations of the traditional explicit authentication mechanisms (e.g., password-based, biometrics), researchers and the industry have been promoting implicit authentication (IA) that does not require explicit user action and potentially enhances user experience to further protect devices from misuse. IA typically leverages various types of behavioral data to deduce a user's behavior model for authentication purpose. However, IA systems are still at their infancy and exhibit many limitations, one of which is how to determine the best retraining frequency when updating the user behavior model. Another limitation is how to gracefully degrade user privilege, when authentication fails to identify legitimate users (i.e., false negatives) for a practical IA system. To address the first problem, we propose an algorithm that utilizes Jensen-Shannon (JS)-dis(tance) to determine the optimal retraining frequency. For the second problem, we introduce a dynamic privilege mechanism, again based on JS-dis(tance), to achieve multi-level fine-grained access control. Our simulation results show that the proposed techniques can successfully detect the degradation of accuracy of the user behavior model, as well as automatically determine and adjust to the best retraining frequency. It is also shown that the dynamic privilege-based access control reduces the impact of false negatives on legitimate users and enhances system reliability and user experience compared with the traditional lock-only method in case of authentication failure.

Chapter 3 describes a method of achieving high-efficiency front-end authentication. Motivated by the great potential of implicit and seamless user authentication, we attempt to build an efficient middle layer running on mobile devices to support implicit authentication (IA) systems with adaptive sampling. Various activities, such as user location, application usage, user motion, and battery usage have been popular choices to generate behaviors, the soft biometrics, for implicit authentication. Unlike password-based or hard biometric-based authentication, implicit authentication does not require explicit user action or expensive hardware. However, user behaviors can change unpredictably, which renders it more challenging to develop systems that depend on them. Various machine learning algorithms have been used to address this challenge. The expensive training process is usually outsourced to the remote server, but this can potentially increase the chance of data leakage. In addition, mobile devices may not always have reliable network connections to send real-time data to the server for training. Motivated by these limitations, we propose a W-layer, an overlay that provides an energy-efficient solution for real-time implicit authentication on mobile devices. The size of the data the system needs to collect at different times depends on the legitimacy of the user. This, in turn, affects how the sampling rate is adjusted, which can reduce energy consumption. To evaluate our method, we conducted several experiments on both synthetic and real datasets.

Chapter 4 proposes PersonaIA system to achieve dynamic feature selection in the process of identifying users. In this work, it solved the data overlapping problem and behavior changing problem using a statistical topic model in an innovative way to capture a unique feature set from user's behavior. More specifically, we proposed a real-time dynamic mapping between users and their feature sets utilizing the eigenvalue derived from their behavioral data. The mapping is unique in a specific moment for each individual. Since the feature set is dynamically chosen to best fit each user's behavior, the chance that two users' data will collide is significantly reduced. In addition, our research work has achieved high authentication accuracy with a small amount of energy input. The approach we proposed successfully solved one of the fundamental problems that had not been addressed for many years. It can be used in almost every implicit authentication system to enhance accuracy. Since the data overlapping problem is also pervasive in machine learning and associated areas as well, our approach can be applied in these areas to reduce the false negative and false positive in the testing phase.

Chapter 5 proposes BubbleMap (BMap) framework laying above various IA schemes to enhance their performance in the aspect of security and usability. Specifically, compared to original IA schemes, the improvements of BMap are: **1)** it enhances the system's authentication accuracy by increasing the chance of blocking illegitimate user, while reducing the chance of falsely rejecting legitimate user; **2)** it can be seamlessly applied to almost every implicit authentication scheme and significantly boosts their performance. To evaluate the proposed framework, we conducted a large-scale simulation using the data from 130 participants. In addition, we implemented BMap-based systems on four state-of-the-art IA schemes. In two years and eight months, we evaluated BMap on various conditions, and tracked the usage of 13 participants. In both large-scale simulation and long-term real test, BMap greatly enhances various IA schemes' performances with a small amount of time and energy input.

1.5 Challenges

Compared to hard biometrics such as the fingerprint, human behaviors are more susceptible to changes due to factors such as time, environment, mood, and age, which renders it more challenging to design IA systems based on behaviors. Because of the behavior change and noise produced by various sensors, users' data often overlaps with each other. If we filter out the overlapping data, the system may mistakenly block the legitimate users whenever their data falls in the overlapping area. If we keep the overlapping data, adversaries may take advantage of it to bypass the authentication. This is one of the fundamental problems in implicit authentication, which remained unsolved for more than seven years. Furthermore, the existing research work in implicit authentication utilizes a fixed feature set to achieve user authentication, which is not applicable in practice due to the behavior change. In my work, I solved data overlapping problem and behavior changing problem using the statistical topic model in an innovative way to capture a unique feature set from the user's behavior. More specifically, I proposed a real-time dynamic mapping between users and their feature sets utilizing the eigenvalue derived from their behavioral data. The mapping is unique among a specific moment for every individual. Since the feature set is dynamically chosen to best fit each user's behavior, the chance that two users' data collide is significantly reduced. In addition, my research work has been proven to be able to achieve high authentication accuracy with a small amount of energy input. The approach I proposed successfully solved one of the fundamental problems that had not been addressed for many years. It can be used in almost every implicit authentication system to enhance their accuracy. Since the overlapping problem is also pervasive in machine learning and associated areas as well, my approach can be applied in these areas to reduce the false negative and false positive in the testing phase.

The majority of existing research works rely heavily on machine learning models such as Support Vector Machine (SVM), k Nearest Neighbor (kNN) and Gaussian Mixture Model (GMM), where the expensive training process needs to be outsourced to the remote server. This design not only increases the communication burden between the client and server, but also potentially increases the chance of private data leakage. However, to achieve front-end learning in the energy limited smart device is difficult. On one hand, the system needs to efficiently handle real-time behavior matching, utilizing user's data that often contains noise. The authentication accuracy, on the other hand, must at least keep the same as or be higher than the original system that outsources training process. In my work, I successfully solved the front-end learning problem by devising a novel system that operates similarly to the mechanical wind vane to achieve user authentication. The proposed system has higher authentication accuracy than the state-of-the-art systems used in implicit authentication at that time; and most importantly, it does not require job outsourcing in the authentication process. While most research works focused on server-end implicit authentication that outsources computation to a remote server, I proposed a front-end implicit authentication system with better accuracy, efficiency and privacy-preserving capability. My work opened a new research area, which could become the main trend of implicit authentication in the future.

Chapter 2

Training and Retraining in Implicit Authentication

2.1 Introduction

As smart devices become the primary means of communication, more and more people rely heavily on them as the main way of Internet access [87]. On the other hand, smart devices store sensitive and private data including bank accounts, passwords, contacts, emails, and photos, while their security has not gained enough attention [101]. To protect smart devices from misuse, many authentication methods such as password, draw-a-secret and fingerprint recognition are employed in various smart device products from different companies [55]. These methods all require explicit user actions (e.g. entering a password, swiping finger), which can be inconvenient and cause users to bypass authentication. Recently, researchers and the industry (e.g., Samsung) became interested in implicit methods for authentication to enhance security and usability. In fact, security and usability are often conflicting goals in that users tend to disable or bypass the security system if it is not user-friendly.

Generally speaking, Implicit Authentication (IA) is a technique that allows the smart device to recognize its owner by being acquainted with his/her behaviors. It is a technique that uses machine learning algorithms to learn user behavior through various sensors on smart devices and achieve user identification [53]. User behavioral data, such as walking style, swipe speed, and location, are used to train user behavior models which are then used as the reference to match with users' current behavior. There are several advantages of IA compared with the traditional explicit authentication. First, behaviors are intrinsic to each person and are accumulated activities over a period of time, and thus cannot be forgotten or easily forged. One may often forget his/her passwords but rarely forgets his/her own behaviors [100]. Even though biometrics are much harder to be stolen than passwords, there have been works showing the feasibility of biometrics forgery [103]. Second, IA is much more user-friendly and requires no explicit user action, which leads to enhanced security against vulnerabilities caused by human factors (e.g., user disabling security features, using weak passwords that are easier to memorize). A recent survey shows that only 44% of smartphone owners configure PINs or passcode on their devices [4]. People find password entering more annoying than lack of cellular coverage, small screen size, and poor voice quality. A recent bypass flaw of Samsung smartphones reveals that vulnerabilities introduced by human factors are potentially more dangerous and easier to be overlooked, even if biometrics-based authentication systems such as fingerprinting and facial recognition are used in place of a password [3] .

On the other hand, IA has its own limitations, one of which being that it is difficult to find behaviors that uniquely identify a user, unlike biometrics. Machine learning is most widely used to tackle this difficulty [17, 80], and there are many research works dealing with how to select suitable machine learning algorithms for various activity types such as those obtained from touch [27, 93], accelerometer [94], location [34], etc., as well as how to provide a general framework for IA [87, 22, 53]. In this research work, on the other hand, we focus on two critical and difficult problems that affect the practical deployment of IA systems: model retraining and handling authentication failure that have not been treated sufficiently in the literature. Retraining is needed since the machine learning accuracy[1]is affected by the quality of the training data (i.e., user behavioral data) as time evolves. As more users join and remain in the system, we are more likely to obtain better training data and capture long-term changing behaviors of users, and hence need to retrain the learned user behavior model by refreshing its parameters at appropriate times to reflect such changes. Authentication failure in this work is referred to as the failure to authenticate legitimate users and block access to the smart device and apps. This can occur when the user's behavior changes, e.g., traveling to a strange place. Existing solutions feature a binary decision-making [53, 111] or similar [63, 18] mechanism that either allows access to or locks the device and some apps at once, which can result in annoyance and the subsequent bypass or removal of the IA system. To the best of our knowledge, we are the first to provide satisfactory solutions to the retraining and authentication failure problems in IA. Our solutions are generally applicable regardless of the machine learning algorithms being used and will lay the foundation for realistic IA systems.

Devising suitable solutions for retraining and authentication failure is challenging due to the following reasons. Finding the optimal frequency of retraining is important but difficult. If the retraining frequency is too high, we could waste a lot of energy and computational resources in order to obtain high accuracy. If it is too low, the accuracy would be affected which could cause high false positive and false negative rate in the authentication. In addition, it is difficult to achieve intelligent retraining, where the retraining frequency is different for different users at different times and spaces. To balance between the performance of the machine learning models and energy consumption, we propose an entropy-based measurement to determine the best frequency for model retraining. The state of the art research uses timeline-based retraining [96, 20]. This method takes advantage of empirical data to determine the best retraining frequency, and uses this time as predefined measurement for future retrainings. The time cycles between retrainings remain the same. However, the change in each user's behavior in IA systems is different and unpredictable. Even the same person could change behavior at any time in an unforeseeable manner. For this reason, the timeline-based retraining may not be effective for IA. For the authentication failure problem, it is highly difficult to balance between false positives (allowing illegitimate users' access to the device) and false negatives (denying legitimate users' access) or offer great user experience when we only have binary options (lock or unlock the device and apps). We therefore propose a new access control mechanism based on dynamic privilege that works by dividing the privilege system into several levels, where each level corresponds to some category of function units or apps with similar sensitivity or security requirement.

[1]Accuracy here is defined as the proportion of correct authentication results (i.e., true positives and true negatives).

For example: the highest level includes mobile banking apps and contact book; the next level includes social apps and device ID, and so on. Instead of locking the device directly when authentication fails, the dynamic privilege mechanism can temporarily assign a reasonable privilege level to the current user based on the JS-dis in between the testing data and the training data. Thus, legitimate users will be able to continue using apps such as Facebook and Maps, but will be temporarily locked out of highly sensitive apps. Access to highly sensitive apps will be automatically regained as more data about the user is collected, without explicit user action. The user can also rely on a backup plan, e.g. entering password, to regain the access. Our simulation results suggest that the dynamic privilege mechanism can largely reduce unpleasant user experience.

2.2 Related Work

This research work is most related to implicit authentication (IA)- mechanisms based on user behavior [100], especially those implemented on smart devices [93, 94, 27]. On the contrary, instead of proposing a new IA mechanism, we address two practical issues inherent in all IA systems that have been largely ignored in the literature. Our work is thus complementary and parallel to these existing related works.

In general, IA relies on the behavioral biometrics that are considered as soft biometrics as opposed to hard biometrics such as facial recognition and iris scan. Specifically, in [93], Sun et al. propose a multi-touch system to authenticate user based on the motion of different fingers. De Luca et al. [27] use touch screen pattern as main attribute to identify different persons. From another angle, Tamviruzzaman et al. [94] propose a multiple-behavior authentication technique that uses both location and gait pattern as soft biometrics to identify user.

An app-centric approach has been introduced in [53] to simplify the IA development. Most of these works focus on one time training without further retraining. In [71], Monrose et al. present the problem of retraining in the authentication, where the machine learning model is retrained once a new user is introduced. Sheng et al.[86] introduce a technique that can retrain part of the system when a sufficient sample has been collected. In [96], Thomas et al. use timeline-based retraining to achieve url spam filtering. In practice, most of the machine learning methods use timeline-based retraining. For this reason, the flexibility of these methods is very limited. The difference between these works and our work is that we use entropy to indirectly measure the accuracy changes in the IA mechanism to further decide the retraining frequency. Our method is dynamically adaptable in that it automatically selects the best retraining frequency for each person which can be varying.

In addition, IA systems implemented on smart devices has been a popular topic recently [65, 53, 111, 63, 18]. However, these systems handle authentication failures by simply locking the device which can be annoying and unacceptable if the failures are caused by false negatives. To overcome this problem, we propose a fine-grained access control system that again leverages entropy to define privilege levels. This degrades user privilege gracefully when authentication fails and greatly enhances user experience for IA systems that are prone to false negatives.

2.3 Preliminary

In this section we provide some background information on entropy, Kullback-Leibler (KL) divergence and Jensen-Shannon (JS) distance, which we use to develop the retraining and dynamic privilege mechanisms.

2.3.1 Entropy or Timeline

Entropy, as it relates to dynamical systems, is the rate of information production [78]. In machine learning, the differences in between entropies are used to measure the similarity in between the testing data and the training data [44].

In this work, we leverage entropy to measure the behavior change of a user for IA, since it is more suitable than the timeline-based method. The behavioral change pattern of each person varies from each other. One may change behavior frequently but others may not. It is also possible that the change varies from time to time for the same person. For example, a person's behaviors can change more rapidly and differently when he/she is traveling in a strange place. Hence, the timeline-based method is not a good choice for retraining. In contrast, each time the behavior changes, it is also accompanied by an entropy fluctuation, as indicated in our evaluation results. By observing these fluctuations, we can determine the best retraining frequency.

2.3.2 KL Divergence and JS-Distance

The common way to measure the entropy difference in between two states is by calculating the KL divergence [57] between them. The KL divergence in between two discrete random variables X and Y is defined as:

$$D_{KL}(X||Y) = \sum_{n=1}^{N} p(X=n) \log \frac{p(X=n)}{p(Y=n)}. \tag{2.1}$$

If the distributions X and Y are equal, the KL divergence is equal to zero.

However, the KL divergence is not a proper distance measure because it is not symmetric [44]. Thus, we use a smoothed and symmetric extension, JS-dis for measuring the similarity. Using (5.6), we can further define the JS-dis as:

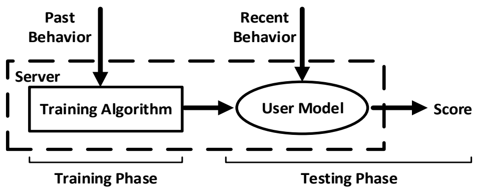
$$D_{JS}(X||Y) = \frac{1}{2}[D_{KL}(X||M) + D_{KL}(Y||M)], \tag{2.2}$$

with the averaged variable $M = \frac{1}{2}(X+Y)$.

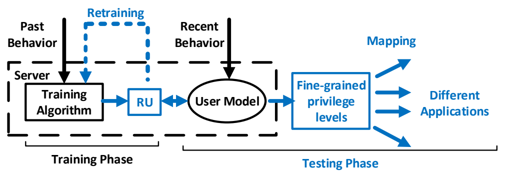
2.4 The Proposed Implicit Authentication Framework and Adversary Model

The basic IA framework is shown in Fig. 2.1 (a), which was first proposed in [22]. We augment it with two key functionalities, retraining and dynamic privilege, to build practical IA systems.

As shown in Fig. 2.1 (a), the basic IA has two phases - training and testing. In the training phase, past behavioral data is input as parameters to the training algorithm. The training result - i.e., a model with tuned parameters - is then returned for testing purpose.



(a) Basic IA framework



(b) Dynamic IA framework

Figure 2.1: IA Frameworks

In the testing phase, which usually happens in real time, recent behavioral data is input into the model and a score is returned to either reject or allow user access.

To cope with retraining and authentication failure, we propose a dynamic IA framework - as shown in Fig. 2.1 (b) - obtaining best retraining frequency and fine-grained privilege control. Compared with the basic IA framework, we introduce a retraining unit (RU) to monitor the behavior changes in real time and automatically decide when to retrain the model. To achieve fine-grained privilege control, we further divide the testing score into different levels, which correspond to different apps (clustered by their sensitivities). Instead of locking the device, our mechanism tends to only lock some sensitive apps based on the testing result.

Adversary Model

We are mainly concerned with adversary who steals the smartphone from a legitimate user, and uses it for accessing sensitive apps and user data. Due to the properties of basic IA, the accuracy of identifying the adversary is proportional to the data collected by the device [87]. Collecting data consumes time which will give the adversary higher chance to gain longer access to the device.

Furthermore, we consider more powerful adversary with the following capabilities.

- The adversary can imitate the legitimate user by observing the user whenever possible, but cannot follow the user all the time.
- The adversary has knowledge of the user's past behavioral data, e.g., by copying the behavioral data stored in the device learning database.

2.5 Intelligent Retraining

To achieve long-term high accuracy, user behavior data must be continually sent to the server to retrain the behavior model and flush the expired parameters. In addition, to achieve low energy consumption we need to find the best retraining frequency. In this section, we will discuss how to design such intelligent retraining.

The idea behind our intelligent retraining is to measure the similarity in between the testing sample and the training samples by using JS-dis. If the distance is larger than the legitimate threshold, it indicates changes in user behavior and the behavior model needs to be retrained.

2.5.1 How to Retrain

To measure the difference in between two individual samples in the training and testing dataset, using JS-dis in Eq. (5.8) we have:

$$D_{JS}(E||R) = \frac{1}{2}[D_{KL}(E||M) + D_{KL}(R||M)]. \tag{2.3}$$

E in Eq. (5.5) indicates a sample in the t(e)sting dataset, and R indicates a different sample in the t(r)aining dataset. M is defined as $M = \frac{1}{2}(E + R)$.

Since there may be noise or error message in the testing data, we need to further measure the average JS-dis in between the training samples and the testing sample in each testing by:

$$\overline{D_{JS}^{(n)}} = \frac{\sum_{k=1}^{K} D_{JS}(E^{(n)}||R_k)}{K}, \tag{2.4}$$

where K is the number of training samples (we call it stride), and (n) denotes the nth testing sample. Since we only need to consider the most recent training data, it is not necessary to include all the training samples. In this work, we use K to indicate these recent data.

For consistency reason, we also cluster K testing samples into one set, which indicates a behavioral pattern of the current user.

After defining the average JS-dis, we can further calculate the standard deviations for the elements in each stride as:

$$s^{(n)} = (\frac{1}{K}\sum_{k=1}^{K}(D_{JS}(E^{(n)}||R_k) - \overline{D_{JS}^{(n)}})^2)^{\frac{1}{2}}. \tag{2.5}$$

In the evaluation section, we will show that the average accuracy can be reflected by the standard deviation of the JS-dis in Eq. (5.4).

2.5.2 When to Retrain

Algorithm 1: Retraining Algorithm

```
   Input: Current Data, Retraining Parameter, Stride
   Output: boolean Retraining_Decision
1  initialize CD:=Current Data, RP:=Retraining Parameter;
2  initialize Retraining_Decision:=false ;
3  CD_JS_dis[]=JS_Dis(TS[],CD) ;
   /* TS[] stores all the previous samples in training set */
4  CD_JS_Dis_ave=Average(CD_JS_dis[]) ;
5  Dis[].add(CD_JS_Dis_ave);
   /* Add the average distance of the current sample to the distance array */
6  if (CD.index mod Stride)==0 then
       /* Completed a stride */
7      std=StandardDeviation(Dis[]);
8      if (std≥RP) then
9          Retraining_Decision=true;
10     else
11         Retraining_Decision=false;
12     Dis[].clear;
13 else
14     Retraining_Decision=false;
15 return Retraining_Decision;
```

To determine the best retraining frequency, we define a threshold ε which represents the acceptable distance such that the accuracy within ε is enough high. If the most recent distance is larger than ε, we should consider retraining.

The detailed retraining algorithm is described in Algorithm 1. Current Data indicates the current testing sample , which is a distribution of different features. Retraining Parameter is the threshold ε. The lower the Retraining Parameter, the higher the accuracy (if the accuracy has not reached the upper bound.) CD_JS_dis[] contains the array of distance in between the current sample and **all** the previous samples in the training set. Dis[] is used to calculate the standard deviations in between CD_JS_Dis_ave values. Finally, if the Retraining_Decision is true, it will ask for retraining.

2.5.3 Retraining Process

Fig. 2.2 shows the process of selecting the retraining frequency based on the testing sample data. Since we already known how to calculate the JS-dis in between the training samples and the current testing sample, we can further take their average value ($\overline{D_{JSc}^{(n)}}$) and mark it as one output of the current stride. In Fig. 2.2, the average value of these JS-distances is drawn as a deep blue stripe. There are more than one $\overline{D_{JSc}^{(n)}}$ in one stride. In this work, the training data could come from the original training or the previous retraining.

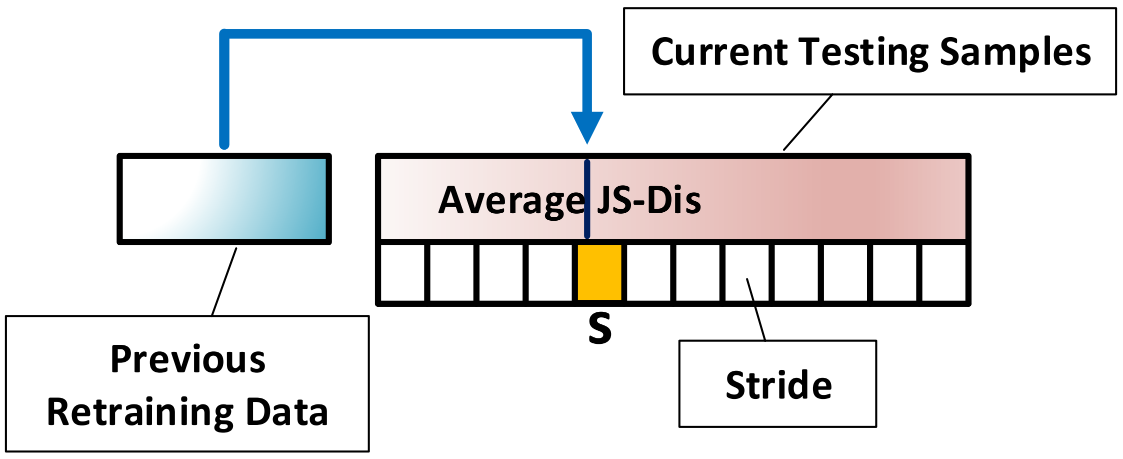


Figure 2.2: Retraining Process

We further divide these average values into different stride. Then, we can calculate the standard deviation for each of these strides' data. In Fig. 2.2, the current sample's standard deviation value is marked as "s". The final step is to compare the standard deviation with the predefined ε. For different implementations, we could choose different values of ε. If $\varepsilon < s$, due to the behavioral pattern change, the accuracy of the user behavior model will drop significantly, and we should retrain the model after this stride.

2.6 Dynamic Privilege-Based Access Control

The basic idea of dynamic privilege is to divide the testing score into fine-grained levels, and each level corresponds to some specific apps. By comparing the current user score with the predefined levels, our mechanism can assign a suitable privilege (by allowing some apps while disabling others) for this user, achieving a more practical access control. To successfully design the dynamic privilege mechanism, we need to answer questions such as how to define each privilege level, what is the entropy distance between each level, how to assign a reasonable privilege to the user based on the entropy distance in between the test data and training data, and how to reassign the privilege if the user's behavior is back to normal? We will discuss these problems in detail and present solutions.

The dynamic privilege mechanism is realized by keeping a multi-level privilege table. In this table, the apps are divided into different categories based on their sensitivity and given different privilege levels. For example, highly sensitive apps such as mobile banking and contact book will be given the highest level. Lower levels will be given to email, maps, games, etc. To categorize each app, one can use the default setting or configure the setting manually. This process depends on the specific implementation on different devices.

2.6.1 Defining the Privilege Levels

The most important step of dynamic privilege-based access control is how to define different privilege levels. We leverage empirical data to find the average JS-dis in between all the previous TP (True Positive) samples and FN (False Negative) samples for each person in the dataset. We first sort these average JS-dis(s) based on their values, and we divide them equally to form different clusters. For each user defined level, we define the rule of it utilizing the average values in the corresponding cluster. For example, after we filter out some noise clusters, the rule of the first level is defined as the average value in the first cluster, and using the same technique, we can define the other levels. These levels can be assigned to apps based on the sensitivity of these apps. Since the empirical data come from the training set, we need to keep the training set "fresh" enough to maintain good performance of the dynamic privilege mechanism. To keep it fresh, we need to retrain the model and keep refreshing the training dataset, which we have discussed in the previous section 2.5.1.

2.6.2 Mapping to the Privilege Levels

After defining the value for each level, we can calculate the average JS-dis $\overline{D_{JS}^{(n)}}$ in between the current testing sample and each training sample, and further decide the appropriate privilege for the user at this time.

Algorithm 2: Mapping Algorithm

```
   Input: Current Data, Current Level, Privilege Table[]
   Output: New_Level
   initialize CD:=Current Data, CL:=Current Level, PT[]:= Privilege Table[];
   initialize New_Level:=null ;
1  for (each training sample i in TS[]) do
       /* calculate the JS-dis in between each previous sample and current testing
          sample                                                                  */
2      JS_dis[i]=JSdis(TS[i],CD);
3      i++;
4  JS_ave=Average(JS_dis[]);
5  if JS_ave>PT[].Level(CL) then
6      New_Level=CL++;
7  else
8      New_Level=CL;
       /* Retrain if necessary                                                    */
9  TS[].add(CD);
10 return New_Level;
```

The mapping procedure is shown in Algorithm 2. We first find the average JS-dis between the current testing sample (CD) and all the previous samples (TS[]), and then calculate the average value of these average JS-distances (JS_ave). Using the defined distance rule for each level in privilege table (Privilege Table[]), we compare JS_ave with the rule associated with the level. If JS_ave is larger, it indicates that the current testing sample is beyond the tolerance distance to the previous i samples, and we will reduce the user privilege to the lower level. However, if JS_ave is smaller, the current testing sample is still very closed to the previous i samples, and thus we will keep the user privilege level unchanged.

Fig. 2.3 shows how the dynamic privilege access control works. The blue stripe in the average JS-dis array indicates the average distance ($\overline{D^{(n)}_{JSc}}$) in between the current testing sample and the previous retraining data (same as Fig. 2.2). The privilege table, which stores the predefined privilege rules based on the training data, is a component that should reside in the authentication module. The lower the number, the higher the privilege level, e.g., L2 is a higher level than L3. To update user privilege, the dynamic privilege mechanism first compares $\overline{D^{(n)}_{JSc}}$ with the predefined rule in the **current** privilege level of the user. For example, if the current level is L2, the mechanism will compare $\overline{D^{(n)}_{JSc}}$ with the L2 rule, and will lower the current privilege if $\overline{D^{(n)}_{JSc}}$ is larger and keep the current privilege unchanged otherwise.

It is possible that $\overline{D^{(n)}_{JSc}}$ is much smaller than the value stored for the current level in the privilege table. It means that the difference in between the current user and the legitimate user is smaller than the given level and indicates that the user privilege should be raised. In this case, the dynamic privilege mechanism will begin to retrain the model and elevate the current user privilege to a higher level if it is not already the highest. For example, if the current privilege is L3, but $\overline{D^{(n)}_{JSc}}$ is much smaller than the L3 rule for a while (e.g. 1 hour), the mechanism will send a retraining signal to the sampling app which will upload more recent samples to the remote server for retraining. At the same time, the mechanism will elevate the current user to L2.

2.7 Discussion

Although the accuracy of IA can increase with more advanced technology used in smart devices, the retraining and authentication failure problems still hinder realistic deployment

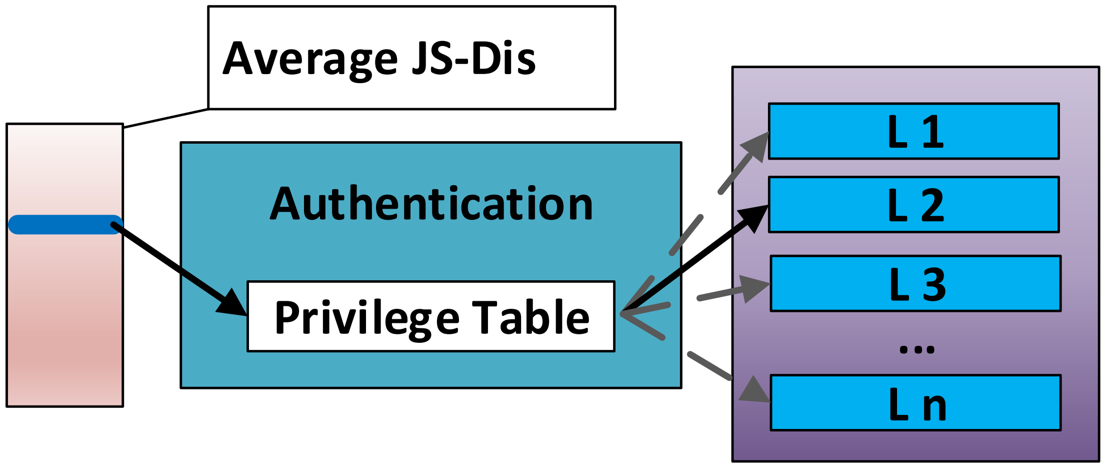


Figure 2.3: Dynamic Privilege Overview

of IA systems. How and when to retrain the user behavior model and what to do when the legitimate user fails the authentication remain unsolved. To address the retraining problem, we proposed a technique using JS-distance to determine the best retraining frequency. For authentication failure, we introduced the dynamic privilege mechanism with finer privilege levels. Compared with the predefined privilege rule, we can decide which level should be assigned to the user based on his/her current behavior. Compared with the lock-only mechanism in the existing related work, the dynamic privilege-based access control can largely reduce unpleasant user experience by only locking part of the device.

In the future, we will implement the retraining and dynamic privilege algorithms on mobile devices to evaluate their efficacy and efficiency. We will also incorporate user feedback to enhance the performance of retraining. From the feedback, system can deduce the false negative and false positive and choose a suitable retraining rate accordingly. In addition, we plan to study the impact of false positives (allowing illegitimate users to access phone contents) that may be induced by our fine-grained dynamic privilege mechanism.

Chapter 3

High Efficiency Front-End Authentication

3.1 Introduction

The rapid growth of mobile devices, especially smartphones, raises security concerns. One such concern is that traditional authentication methods, such as password or hard biometrics, can be potentially circumvented [3]. A recent survey [4] showed that only 44% of smartphone owners configured PINs or passcodes on their devices. Another survey [49] showed that 56% of participants mistyped a password at least one time out of every ten tries, and concluded that users consider password entry on smart devices more annoying than the lack of coverage, small screen size, or poor voice quality. Hard biometrics-based authentication, such as face recognition, touch ID and iris scan require expensive hardware and explicit user interaction. The disadvantages of the traditional authentication methods have inspired researchers to develop new authentication methods that are transparent to users and incur low cost. One promising method is implicit authentication (IA) based on user behaviors which can be considered soft biometrics. Various activities, such as arm swing, walking, and contextual data (e.g., location) [97, 94, 29] are commonly used to derive soft biometrics. Compared with password-based and hard biometric-based authentication, IA does not require explicit user actions since it runs in the background and silently obtains user behaviors. Moreover, unlike hard biometrics, soft biometrics in IA only require basic hardware and sensors readily available in smartphones. Nevertheless, behaviors are more susceptible to changes due to factors such as time, environment, mood, and age, which renders it more challenging to design IA systems based on behaviors.

In this research work, we build an overlay, called W-layer, that runs independently within the device and can be integrated seamlessly with various machine learning algorithms. To ensure practical deployment for popular battery-powered devices such as smartphones, pads and smart watches, our system is made energy-efficient by leveraging lightweight computation and adaptive sampling. The uniqueness of W-layer is two-fold. First, W-layer provides a client-side-lightweight IA solution that can replace or assist in the IA solution that relies on complex machine learning. The majority of existing research work [62, 93, 53, 87] relies heavily on machine learning models such as Support Vector Machine (SVM), k Nearest Neighbor (kNN) and Gaussian Mixture Model (GMM), where the expensive training process needs be outsourced to the remote server. This design not only increases the communication burden between the client and server, but also potentially increases the chance of private data leakage. Unlike the framework in [53, 54], W-layer is energy-efficient in that only lightweight computation is involved in behavior matching and adaptive sampling is employed to keep the sensing overhead at bay. The resulting solution is therefore practical even on low-end mobile devices. It will become apparent in our later discussion that W-layer is a self-contained IA solution that can replace the machine-learning-based solutions. To use W-layer as an assisting technology, machine learning can be run in the server to obtain a reference behavior model for each user. This reference model will be used by W-layer to guide activity data collection and real-time behavior matching. In this case, machine learning will run much less frequently (assuming behavior change happens less often than authentication) and less real-time communication will take place. Second, W-layer addresses the challenging problem of distinguishing between legitimate users' behavior deviation and illegitimate users' behaviors which affects the accuracy and practicality of the system.

In addition, to successfully develop W-layer, we need to solve challenges related to system design, which are associated with designing the system architecture of W-layer and developing it into a practical system. Specifically, we need to devise a real-time behavior matching algorithm which is the core component of and suitable for IA. Furthermore, to ensure practical deployment and user acceptance, this algorithm should improve system reliability by reducing the false negative and false positive rates.

3.2 Related Work

Our work leverages user behavior for IA. The majority of the recent research work, e.g., [62, 93, 53, 87, 41, 24], relies on machine learning techniques such as SVM, kNN and GMM. The expensive training process of these techniques is outsourced to remote server as discussed in [49, 22]. The main difference between the existing work and our proposed work is four-fold. First, our method utilizes Jensen-Shannon (JS) divergence to identify the user. It can achieve IA and significantly reduce energy consumption due to training and server-client communication. Furthermore, uploading the private data, e.g., GPS and touch, potentially increases the chance of data leakage, but our approach implements IA independently in the device and significantly improves personal data privacy. Therefore, our approach represents a new way of implementing IA. In addition, existing work features a fixed or static set of activities from which behaviors are derived, and our work features dynamic sets (represented by the size of stride discussed in Section 3.5). Because of the fixed or static nature, existing techniques are limited in one way or another whereas our approach is flexible and versatile. For example, quite a few papers study the touch behavior [63, 18, 19, 27, 93, 70, 85] which is specific to this generation smartphones. Their techniques may not work for smart watches and glasses, or even the next generation smartphones (touch may not be the main way of operating future smartphones). Finally, some existing work allows the user to adjust the sampling rate, e.g., [62]. But their method essentially still belongs to static IA because the sampling rate is not automatically adjusted by the system.

3.3 Preliminary

In this section, we introduce tokenization, the method we use to represent activities from which behavior is deduced. JS divergence is also discussed which we use for similarity comparison.

3.3.1 Tokenization

Given a dataset containing various raw sensor data, tokenization is the task of chopping the data into pieces (tokens). At the same time tags are added to ambiguous tokens, and certain characters in the stop-word list are discarded. For example, in here, a GPS reading will be tokenized into the following: (loc69x, loc1333y), and a light reading will be tokenized into: (l98). We use the prefix to identify each sensor (Therefore loc = location and l = is light). The number attached to the label is the actual reading from the sensor. After tokenizing we are now left with a set of tokens. Each token is truncated according to its value(s). For example, the light meter reading l30.41 and l30.58 would belong to the same token "l30". The location reading of "11th Ave, City_A" is split into two distinct tokens "City_A" and "11th" by filtering out words according to the stop-word list, e.g., Ave.

3.3.2 Jensen-Shannon Divergence

The common way of measuring similarity in between two distributions is by calculating the Kullback-Liebler (KL) divergence, which is defined as:

$$D_{KL}(\mathbf{X}||\mathbf{Y}) = \sum_{n=1}^{N} p(X = n) \log \frac{p(X = n)}{p(Y = n)}, \tag{3.1}$$

where $\mathbf{X}$ and $\mathbf{Y}$ are two distributions.

However, the KL divergence is not well bounded and it is an asymmetrical measurement, which is not suitable in building our system. To overcome these drawbacks we adopt JS divergence [64], which is defined as:

$$D_{JS}(\mathbf{X}||\mathbf{Y}) = \frac{1}{2}[D_{KL}(\mathbf{X}||\mathbf{M}) + D_{KL}(\mathbf{Y}||\mathbf{M})], \tag{3.2}$$

where $\mathbf{M} = \frac{1}{2}(\mathbf{X} + \mathbf{Y})$. Particularly, for a distribution $\mathbf{X}$, $D_{JS}(\mathbf{X}||\mathbf{X}) = 0$; otherwise, $D_{JS}(\mathbf{X}||\mathbf{Y})$ will shift to 1 depending on how large of the difference between $\mathbf{X}$ and $\mathbf{Y}$, for example given $P_{\mathbf{X}} = [p_{X_1}, p_{X_2}] = [1, 0], P_{\mathbf{Y}} = [p_{Y_1}, p_{Y_2}] = [0, 1]$, $D_{JS}(\mathbf{X}||\mathbf{Y}) = 1$.

3.4 System Architecture and Adversary Model

3.4.1 System Architecture

With knowledge briefly described in the previous sections, we are ready to build W-layer. The system architecture, including the main modules and functional blocks, interfaces, and data flows, is shown in Fig. 3.1.

As shown in Fig. 3.1, sensor monitors are used for collecting raw behaviors such as $location = (loc69x + -20, loc1333y + -20)$ and $touch\_screen = Scroll$. These data are stored in the system's Cache DB component (1), which is implemented using SQLite. The Cache DB module uses a pre-designed Interface for receiving data and facilitating the implementation of new sensors and data sources. Another module on the client side is the Behavior Matching Module implemented as a background service, which we have named the Wind Vane Module (WVM). The WVM is the main mechanism driving the novel dynamic adaptability of the system and is in charge of the real-time-client-side authentication. The WVM captures user's real-time behaviors (3) and compares it with the legitimate user's historical behaviors captured and stored in local storage (hard drive for example). After comparison, a matching result will be returned that determines if the current user is a legitimate user. The Behavior Matching Module also controls the sampling frequency and fine-grained data uploading mechanism (2).

From another aspect data flows in the system can be explained as interactions between different layers shown in the left part of Fig. 3.1, while W-layer can be viewed as a vessel to exchange the information between the Machine Learning Model and the WVM. In W-layer, the communication is enhanced by data serialization, in which data exported from W-layer is serialized to a single string (different sensors will be attached with different labels).

The Cache DB, shown in Fig. 3.1, temporarily stores the sampling data. Data held in the Cache DB is periodically exported to the local machine, where the legitimate data is combined to form the historical dataset. The smartphone's internal storage is much faster than the external storage, but is also very limited in size. For a successful implementation of an IA system, authentication needs to happen in real-time. It is imperative that the identification unit is implemented using local storage.

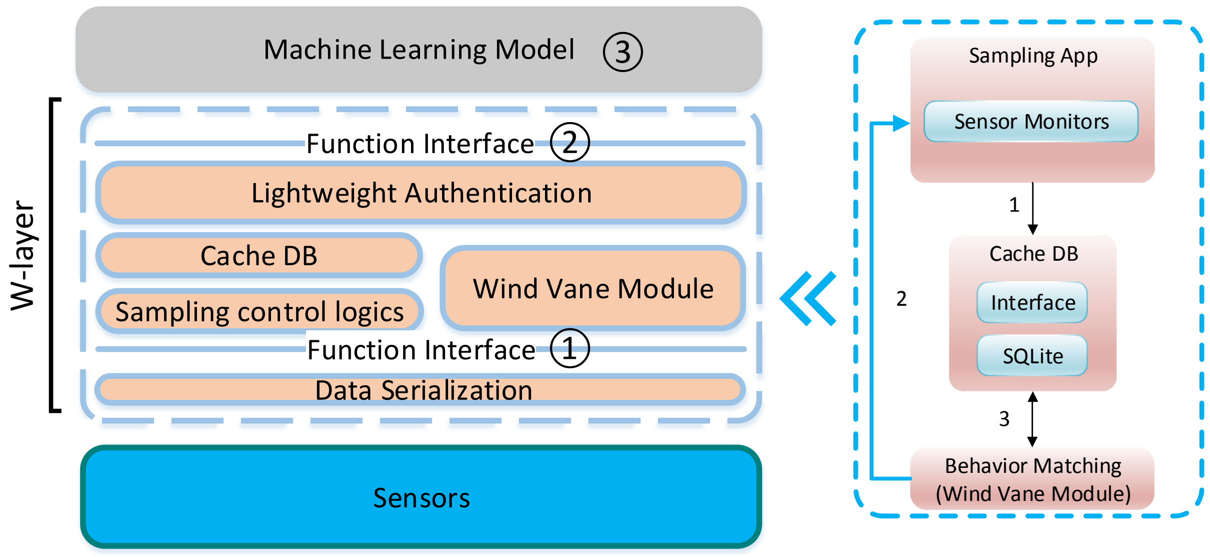


Figure 3.1: The system architecture.

Although our method is designed naturally to support local IA, it can also cooperate with other machine learning techniques through Function Interface (① and ②) which provides raw data and fine-grained data respectively during the uploading.

3.4.2 Adversary Model

We assume two types of adversary. One is a regular adversary who is unaware of our IA system, i.e., who does not know or is unable to mimic the legitimate user's behaviors. We also consider a powerful adversary, who is able to emulate the behaviors of the legitimate user. An example of a regular adversary is a random person who finds the lost device and attempts to use it. An example of a powerful adversary is a close friend, co-worker, or family member who tends to share a lot of similar activities.

3.5 Wind Vane Algorithm

There are two components of IA systems that are energy intensive: sampling and data storage. Regularly fetching users' behaviors and grabbing unnecessary data, such as features that are useless in distinguishing users, cause excess sampling and storage, which reduces energy efficiency. W-layer tackles this problem by first having an adaptive sampling rate.

When we suspect the legitimacy of a user, we increase the sampling frequency. On the other hand, we decrease the sampling frequency if the legitimacy is confirmed. In addition, W-layer selects the most distinguishing features among people for authentication. These features change over time which can be captured by W-layer. Finally, once we have sent a minimum amount of data to the Behavior Matching Module, we use JS divergence [64], which is a widely-used and energy-efficient method for comparing two distributions.

In this section, we will discuss several different IA modules that can achieve fine-grained sampling while being lightweight enough to run on the client side. Each module builds upon and improves over the previous one. Moreover, fine-grained sampling can be used to enhance the performance of other machine learning techniques, e.g., SVM, PLDA [76] and neural networks. We begin the section by introducing the Basic Module and Stride Module, which are widely used in current IA systems. After carefully examining the these modules, we find that they are static in nature and may not adapt to the changing behaviors well. We then introduce the Dynamic Stride Module, an adaptive alternative. Although this module can handle dynamically changing behaviors, the problem of behavior deviation is left unsolved. Behavior deviation occurs when the legitimate user's behavior deviates too much from his/her historically "normal" behaviors which could cause mis-authentication. Finally, we conclude with a complete module called Wind Vane, which successfully extracts the most representative dynamic behaviors of the legitimate user, and overcomes all the aforementioned problems.

3.5.1 Basic Module

In general, to achieve IA, a fine-tuned module returned by machine learning technique is used to identify the legitimate user based on the sampling data stored in the device. We call this method the Basic Module. Most current IA mechanisms fall into this category and the next (Stride Module discussed in Section 3.5.2) [53, 49, 62, 60].

We can easily modify the Basic Module to achieve dynamic sampling and energy-efficient authentication. As shown in Fig. 3.2 (a), the first step is tokenizing raw data sampled from various sensors. We then count the number of occurrences for each type of tokens and

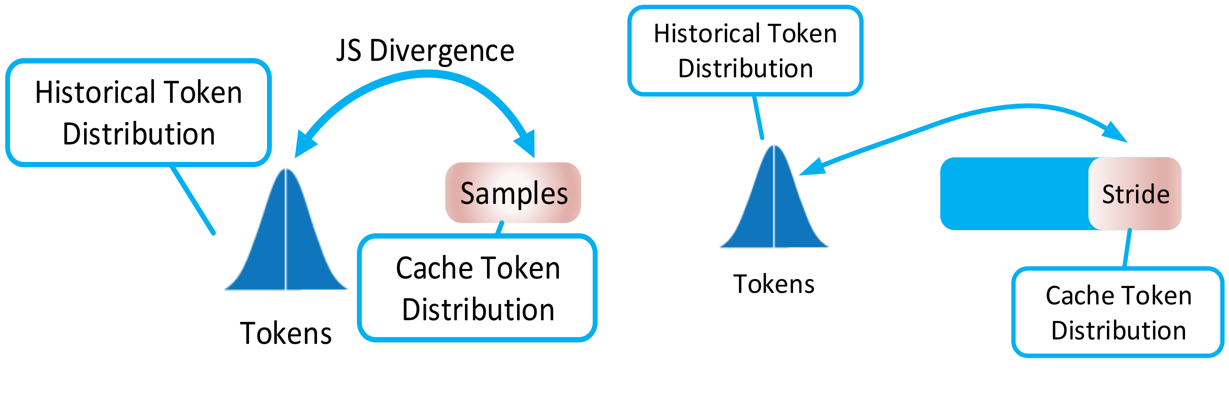


Figure 3.2: Implicit authentication modules. (a) Basic Module and (b) Stride Module

generate the distribution for this sample. The distribution can span several sampling cycles where the tokens are counted accumulatively.

DEFINITION 1. *Let* $\mathbb{S}^{(\phi)} = \{s_1^{(\phi)}, s_2^{(\phi)}, ..., s_{i\in I}^{(\phi)}; 1 \leq \phi \leq \Phi\}$ *be a behavioral sample space containing* i *distinct tokens sampled from various sensors in time* ϕ*, where* Φ *is the maximum time range and* I *is index set. The frequency of each token is* $P(X_i) = \frac{\sum_\phi n_i^{(\phi)}}{\sum_\phi N^{(\phi)}}$*, where* $n_i^{(\phi)}$ *is number of times each token appearing in time* ϕ*, and* $N^{(\phi)} = \sum_{i=1}^{I} n_i^{(\phi)}$*. The* ***accumulative behavioral pattern (ABP)*** *is defined as:* $P(\boldsymbol{X}) = \{P(X_1), P(X_2), ..., P(X_I)\}$.

Once the tokenization, trimming and accumulation have been completed, we can perform authentication (or behavior matching) using JS divergence. Assuming we have a constant sampling rate t, D_{JS} represents the JS divergence between the tokens' frequencies in the Cache DB (can contain either legitimate or illegitimate data) and in the historical data (must be legitimate). The time gap between two samples is determined by $T = \frac{t}{D_{JS}^{\gamma}}(t > 0, 0 \leq D_{JS} \leq 1)$, where γ represents the weight parameter for D_{JS}, and t is a predefined constant sampling time. Large JS divergence indicates a higher possibility that the current user is an adversary.

By comparing between the distributions of the historical data and the current cache data (which contains several samples as shown in Fig. 3.2 (a)), we can determine the legitimacy of the current user. Small D_{JS} indicates that the current user is more likely the legitimate user. In contrast, large D_{JS} indicates the user is an adversary. Based on this result, we can either speed up or slow down the sampling rate by controlling the gap between the sampling cycles using $T = \frac{t}{D_{JS}^{\gamma}}$. Generally speaking, representing humans' behaviors requires more than one sample. Thus, it may not be enough if we only choose one of the most recent samples to verify the current user. In the Basic Module, the ABP is represented by choosing as many samples as possible from the Cache DB.

The Basic Module can dynamically change the sampling rate to suit different situations. For different users, the sampling rate, which is related to D_{JS}, should be different. Compared to the traditional method of a constant sampling rate, this module is more flexible for devices with a small memory.

The Basic Module achieves dynamic sampling with several issues. To calculate D_{JS}, we must traverse the whole cache dataset. If the cache size is large, the time consumption would be overwhelmingly high. The other problem is that the cache may still hold old data from the previous user, and this data may have an adverse effect on D_{JS}. If the previous user is the legitimate user and the current user is illegitimate, the system will keep the low sampling rate since the cache contains a small proportion of samples from the current user. To overcome these problems, we introduce the Stride Module, which is an extension of the Basic Module.

3.5.2 Stride Module

The underlying mechanism of the Stride Module is to select a subset of the cache data for authentication purpose. Similar to the Basic Module, by comparing between the current user's ABP derived from the subset of the cache data and the historical ABP, we can determine the legitimacy of the user and adjust the sampling rate accordingly.

The Stride Module is shown in Fig. 3.2 (b). We first define a suitable "stride size," which is a subset of the cache data. We divide the whole cache dataset into different chunks, the size of which is equivalent to the stride size and each of them represents a typical (non-repetitive) behavioral pattern that minimizes classification error. Instead of comparing the whole cache data with a legitimate user's ABP, the Stride Module only examines a small portion of the data in the cache. Suppose we have a stride with a size of K. Then using the same calculation in the Basic Module, we can derive the JS divergence between the two distributions using $D_{JS} = D_{JS}(X_K||Y)$. Splitting the cache results in a significant improvement in speed compared to the Basic Module. For example, if we have a cache that stored 5000 samples (one day's data samples) and we set the stride to a value of 50 samples, the Stride Module can run up to 100 times faster than the Basic Module. This mechanism is useful when the cache size is large (usually modern smartphones can easily hold 10^4 to 10^5 samples from all sensors). Furthermore, since the user's ABP can be represented by a limited number of samples, the Stride Module can be used to identify the behavioral patterns of current users while filtering out noise from the previous users and correctly determining the sampling rate.

In the Stride Module, we need to find a good stride size to represent the current user. Such a stride allows W-layer to separate the current user from the previous users with high accuracy, and has been proven to be effective in detecting powerful adversaries [60]. However, the amount of samples needed to create a behavioral pattern varies among people. A predefined stride size is not ideal. For example, suppose we only need 12 samples to represent the behaviors of Alice. If Alice is an adversary, we can separate her from a legitimate user Bob using only 12 samples. However, this stride size is only suitable for Alice, and will change over time. Hence, a fixed stride size is not feasible even for the same person, and we need to consider a more sophisticated module. In the following subsections, we will focus on how to choose the ideal stride size for each person at different times.

3.5.3 Dynamic Stride Module

Before diving into the Dynamic Stride Module, we need to discuss some facts about ABP. Assuming we have a legitimate user Y and an adversary X, their behavioral patterns are represented by $P(Y)$ and $P(X)$, respectively. We further assume that adversary X has stolen Y's smartphone. If we calculate the JS divergence between two behavioral patterns from the same person, we will get a value close to 0 because the behavioral patterns should be extremely similar, e.g., $D_{JS}(X||X) = D_{JS}(Y||Y) = 0$. However, if Y's behavioral pattern is compared with X's behavioral pattern, the JS divergence should be a positive real number, which is less or equal to 1. Ideally, it would be close to 1, $D_{JS}(X||Y) = D_{JS}(Y||X) = \delta > 0, \delta \leq 1$. This is because a value closer to 1 indicates a stark difference between the two behavioral patterns. Thus, assume the cache DB consists of data from both X and Y, and this data is called M, we will have $0 < D_{JS}(Y||M) < \delta$. If M contains a larger proportion of the legitimate user's data, $D_{JS}(Y||M)$ will be close to 0. It will be close to 1 if M contains enough adversary's data. Furthermore, once we confirm that the current user is an illegitimate user, we wish to increase his/her JS divergence as much as possible by adjusting the stride size. However, if it turns out that the current user is the legitimate user, we wish to minimize the JS divergence. The corresponding stride size is called the optimal stride size.

DEFINITION 2. *Suppose we have historical ABP - $\boldsymbol{Y}$, which contains the historical behavioral pattern of a legitimate user, and the current ABP - $\boldsymbol{X}$. For the index set J, the* ***best behavioral pattern for the adversary*** *$\boldsymbol{X}_A$ is defined as: $arg \max_{j \in J} D_{JS}(\boldsymbol{X}_j || \boldsymbol{Y})$; the* ***best behavioral pattern for legitimate user*** *$\boldsymbol{X}_L$ is defined as: $arg \min_{j \in J} D_{JS}(\boldsymbol{X}_j || \boldsymbol{Y})$; the corresponding number of samples in j is called the* ***optimal stride size****, denoted by $|\boldsymbol{X}_j| = C'$.*

Let us first discuss how to initialize the module to select the optimal stride size for a *legitimate user*. We assume that the first user is legitimate, and the current stride size is set to C. We want to find the optimal stride with a size of C' that produces the smallest D_{JS}, which should be close to 0. To find the stride size C', we enlarge or shrink the current stride n times and choose the stride with the smallest D_{JS}. We keep repeating this process during the initialization phase until the difference between the current $D_{JS}^{(j)}$ and the previous $D_{JS}^{(j-1)}$ is very small, i.e., $D_{JS}^{(j)} - D_{JS}^{(j-1)} < \varepsilon$. We use the stride size $|\mathbf{X}_j|$ as the optimal stride size in the initialization step, and let $C' = |\mathbf{X}_j|$.

Now we discuss how to select the best behavioral pattern for the adversary during the authentication. Suppose we begin with a large stride value $|\mathbf{X}_{j-1}|$, which could be potentially equal to the entire cache, i.e., $|\mathbf{X}_{j-1}| = \hat{C}$. We have an ABP with JS divergence $D_{JS}^{(j-1)}$, and later we use an n-times smaller stride $|\mathbf{X}_j| = \frac{\hat{C}}{n}$ which gives us $D_{JS}^{(j)}$. If $D_{JS}^{(j-1)} < D_{JS}^{(j)}$, we know that the current stride S_n contains behavioral patterns that may not belong to the legitimate user. Therefore, we need to further shrink the stride until we reach some point $\mathbb{J}$ where $D_{JS}^{(\mathbb{J}-1)} < D_{JS}^{(\mathbb{J})} \geq D_{JS}^{(\mathbb{J}+1)}$, at which time we know the stride in point $\mathbb{J}$ should be the optimal stride. We should speed up the sampling rate accordingly since the current user has high chance to be an adversary.

Fig. 3.3 depicts the Dynamic Stride Module when assuming the current user is an adversary. In the figure, for each previous stride, the number of samples is monotonically decreasing, i.e., $|\mathbf{X}_1| > ... > |\mathbf{X}_{j-1}| > |\mathbf{X}_j|$. On the contrary, the corresponding JS divergence is monotonically increasing, i.e., $D_{JS}^{(1)} < ... < D_{JS}^{(j-1)} < D_{JS}^{(j)}$. After $D_{JS}^{(j)}$ is reached, we have already shrunk the stride to be n^{j-1} times less than the original stride. In Fig. 3.3, since $D_{JS}^{(j-1)} < D_{JS}^{(j)} \geq D_{JS}^{(j+1)}$, and $D_{JS}^{(j)}$ is the maximum JS divergence for the current user, we choose $|\mathbf{X}_j|$ as the optimal stride size. If we set $n = 2$, we can further simplify the problem to a binary search problem.

This module not only supports dynamic sampling but also automatically determines the stride size. However, in this module, we made several assumptions. The first assumption is that the smartphone has been stolen. This assumption may be unreasonable if the legitimate user experiences behavior deviation, e.g., travel. A better module is needed if this assumption is relaxed. Also, we assumed that the starting stride is large enough to cover all the behavioral patterns of the current user, and the size of the stride could be equal to the total cache size. This design is not the most efficient, and it serves only to gain some fundamental understanding of finding the best behavioral pattern. In the following subsection, we will present the Wind Vane Module that is adaptive enough to address all these issues.

3.5.4 Wind Vane Module

In our previous discussion, the legitimate user's behavior deviations may be detrimental to the correct execution of IA. Intuitively, deviated behaviors may be temporary or bear more similarity to the legitimate user's historical behavioral patterns than illegitimate users. To capture this intuition, once our system observes that similar historical behavioral patterns occur again, it increases its confidence about the legitimacy of the user. Translating this intuition to system design, our system needs to re-observe that the JS divergence monotonically decreases to confirm the behavior deviation. Thus, we need to build a module that can not only find the optimal stride size but also detect whether the behavioral change is due to the legitimate user's behavior deviation.

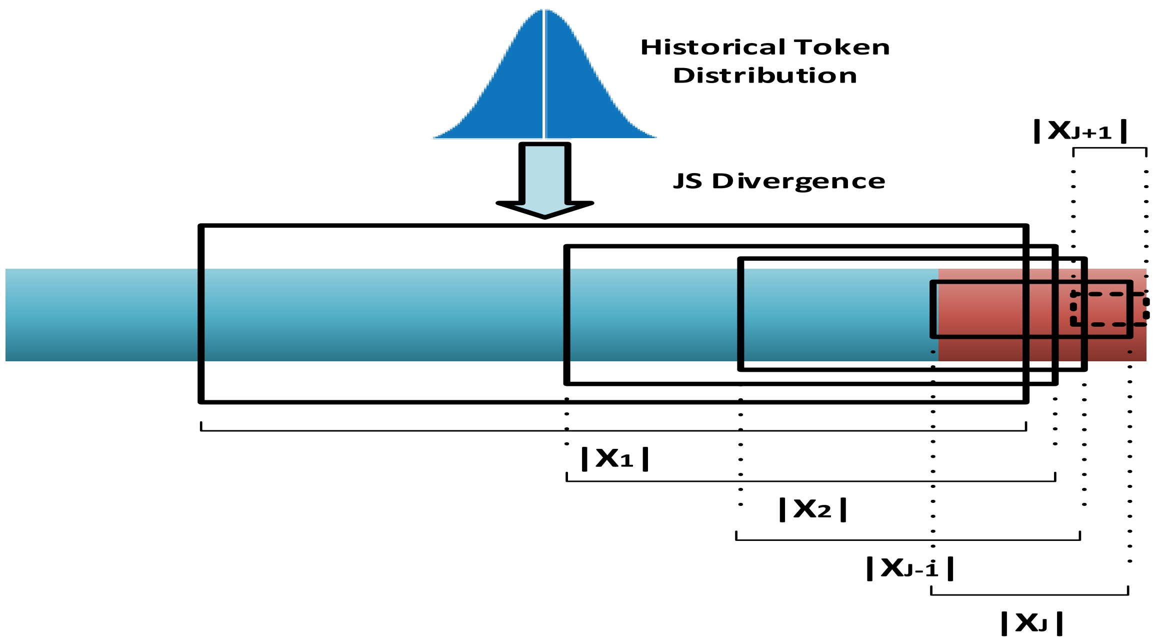


Figure 3.3: Dynamic Stride Module.

We name this module Wind Vane, which is inspired by the module's similarity to the mechanical wind vane used to measure the direction of the wind. Instead of measuring the wind, the proposed WVM measures the change of direction of one's behaviors and also observes if the behavior change is due to an adversary or a legitimate user's behavior deviation.

The Wind Vane Module (WVM) has the following metrics:

*1. Pointer: p, indicating the behavior (wind) direction, either from (**L**)egitimate user to (**A**)dversary or (**A**)dversary to (**L**)egitimate user.*

2. Wind strength: D_{JS}, equal to the JS divergence.

3. Wind strength threshold: δ. The wind strength (D_{JS}) must be larger than δ to rotate the wind vane.

4. Duration time: m, behavior (wind) duration.

The difference between the Dynamic Stride Module and WVM is that the Dynamic Stride Module only shrinks the stride size to find the best behavioral pattern. The WVM first observes the JS divergence changes, and then decides to either enlarge the stride size or shrink it according to whether the adversary or the legitimate user is observed.

Suppose we take a series of continuous strides with equal size, $|\mathbf{X}_1| = |\mathbf{X}_2| = ... = |\mathbf{X}_{n-1}| = |\mathbf{X}_n|$. The corresponding JS divergence for each sample is $D_{JS}^{(1)}, D_{JS}^{(2)}, ..., D_{JS}^{(n-1)}, D_{JS}^{(n)}$. If at some stride $j \leq n$, the system observes that $D_{JS}^{(j)} < D_{JS}^{(j+1)} < ... < D_{JS}^{(j+m)}$, it will change the direction of the wind vane from $\mathbf{L}$ to $\mathbf{A}$, and p will turn to point to the adversary (from the legitimate user). Since we are now concerned with an adversary, the system will start using a faster sampling rate. Also, the WVM will adjust the stride size to indicate the behavioral patterns of the adversary (maximizing D_{JS}). In contrast, if at some stride $j \leq n$, the system observes that $D_{JS}^{(j)} > D_{JS}^{(j)} > ... > D_{JS}^{(j+m)}$, it will mark the direction from $\mathbf{A}$ to $\mathbf{L}$ (behavior deviation), and p will turn to point to the legitimate user (from the adversary). In this direction, the system learns that the current user is most likely the legitimate user, and hence will reduce the sampling rate and adjust the stride size to indicate the behavioral patterns of the legitimate user (minimizing D_{JS}). In practice, the JS divergence may not always be monotonically increasing or decreasing due to irregular behaviors or noise. For this reason, we compare the current D_{JS} with the threshold δ . For example, since we assumed that the first user is legitimate and predefined δ to be 0.5, D_{JS} should be smaller than δ. If the system observes in a later sample that the current D_{JS} is greater than δ and this trend continuously repeats for m times, the WVM will consider the current user to be an adversary and rotate p to point to the adversary.

Fig. 3.4 describes how the WVM works internally. Unlike a mechanical wind vane, which can point in any direction, the WVM only points in two directions. The direction change of the WVM only happens in the following two cases. *Case 1 (illegitimate usage)*: after initializing the stride size of a legitimate user's behavioral pattern, the WVM will monitor the JS divergence in each sample. If it finds that the JS divergence is larger than the threshold δ and this trend occurs for m samples continuously, the pointer p is set to $\mathbf{A}$. Under this condition, the WVM will re-initialize the stride size to one with a larger JS divergence. The corresponding sampling rate will also be increased. *Case 2 (behavior deviation)*: due to the behavior deviation, the WVM may point to $\mathbf{A}$ temporarily but rotate the pointer p back to $\mathbf{L}$ if the system re-observes that the JS divergence decreases monotonically. When this happens, the WVM will try to find a stride size that minimizes the JS divergence and decreases the sampling rate accordingly.

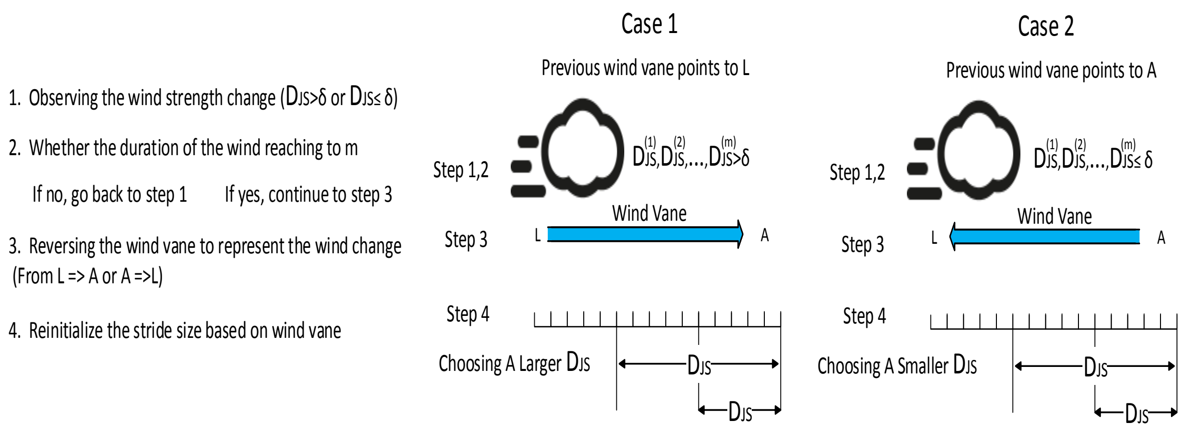

1. Observing the wind strength change ($D_{JS}>\delta$ or $D_{JS}\le\delta$)
2. Whether the duration of the wind reaching to m
If no, go back to step 1
If yes, continue to step 3
3. Reversing the wind vane to represent the wind change (From L => A or A =>L)
4. Reinitialize the stride size based on wind vane
Case 1
Previous wind vane points to L
Step 1,2
$D_{JS}^{(1)},D_{JS}^{(2)},...,D_{JS}^{(m)}>\delta$
Wind Vane
Step 3
L
A
Step 4
Choosing A Larger D_{JS}
D_{JS}
D_{JS}
Case 2
Previous wind vane points to A
Step 1,2
$D_{JS}^{(1)},D_{JS}^{(2)},...,D_{JS}^{(m)}\le\delta$
Wind Vane
Step 3
L
A
Step 4
Choosing A Smaller D_{JS}
D_{JS}
D_{JS}

Figure 3.4: Wind Vane Module.

Compared with the previous modules, the WVM can achieve: **(i)** less computational overhead; **(ii)** less power consumption; and **(iii)** user behavioral pattern derivation with adaptive noise filtration.

3.6 Parameter Selection

To facilitate the implementation of the WVM, we analyzed how to select the parameters, e.g., m, δ and γ. The results are discussed in this section.

3.6.1 Data Collection

The dataset we used in the simulation is the MIT Friends and Family Dataset [5], which contains 130 participants and a total of 9 types (GPS, accelerometer, SMS, app installation, battery usage, call logs, app usage, blue-tooth devices log, Wi-Fi access points) recorded over 5 months. Among all the 132,960,052 samples in the dataset, 10% of them were used in our experiments. Although the data columns related to user ID, e.g., phone number, were hashed and never saved in clear text, we still deleted them from the dataset. The data was tokenized, where the top 30 most frequently used tokens were filtered out.

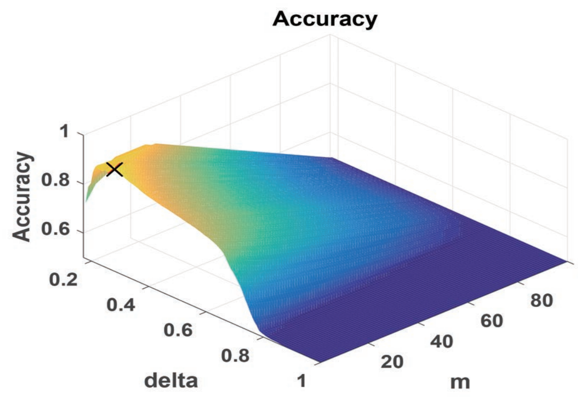
To simulate the real usage, we prepared two datasets, which correspond to the normal usage (by the legitimate user) and abnormal usage (e.g., device captured by the adversary). The first dataset, referred to as the legitimate dataset, contains data sampled only from the legitimate user. The second dataset, refered to as the synthetic dataset, was created utilizing the *splicing* approach [87] as follows. We randomly select a user to be the legitimate user, and another user to be the adversary. We splice a portion of the legitimate user's segment with a portion of the adversary's segment. The point at which the two segments are concatenated is called the *splicing moment*. In practice, the *splicing moment* can be regarded as the time of device capture.

3.6.2 Data Analysis

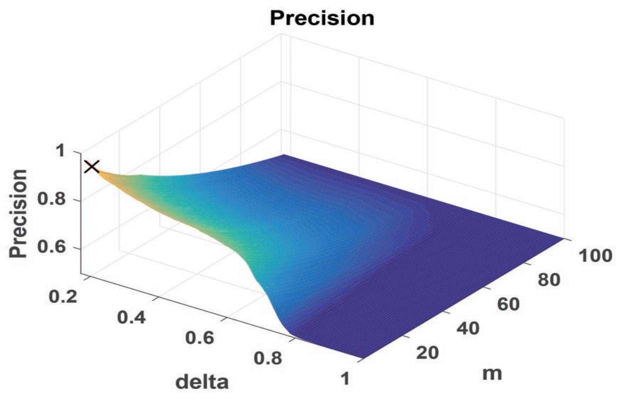
We randomly selected 40 people in the synthetic dataset, and each person's data contains more than 0.12-million samples over 5 months. We further chose the values for δ and m that minimize the classification error. The average results are shown in Fig. 3.5, in which m was defined in the range between 1 and 100. Since we selected the stride size that minimizes the JS divergence, δ shown in the figure is between the minimum JS divergence and 1. We calculated the accuracy ($\frac{TN+TP}{TP+FP+TN+FN}$) and precision [1]($\frac{TP}{TP+FP}$) for each pair of δ and m, and chose the pair that produces the best results. As shown in Fig. 3.5, the δ and m pair was marked with a red "X", and the corresponding accuracy and precision are 92.53% and 97.57%, respectively. In the test, we found that the optimal value of δ is very consistent across the dataset, scattered between 0.2 to 0.4. However, the optimal value of m fluctuates between 1 to 20. Another observation is that the best results in accuracy and precision are not always consistent, which is clearly shown in Fig. 3.5.

Intuitively, γ represents the system's belief on the effectiveness of JS divergence as the similarity metric. Together with the JS divergence, it adaptively controls the time gaps between two samples by $\frac{t}{D_{JS}^{\gamma}}$. When γ is large, the system has strong belief on the JS divergence and WVM. To avoid bias, we choose γ by averaging over the historical testing results, in each of which a mis-authentication should decrease γ until $\gamma = 0$ and a correct authentication will increase γ until γ reaches the maximum, e.g., 1.5, in the synthetic dataset, equivalent to roughly 5-minute sampling gaps when $D_{JS} = 0.1$.

[1]where TP (true positive) means that the system correctly passes legitimate users, TN (true negative) means that the system correctly blocks illegitimate users, FP (false positive) means that the system incorrectly passes illegitimate users, and FN (false negative) means that the system incorrectly blocks legitimate users.



(a)



(b)

Figure 3.5: Parameter selection. (a) Accuracy, (b) Precision.

3.7 Discussion

We introduced W-layer and the WVM to achieve lightweight IA, while also supporting the traditional server-based IA. W-layer challenges the traditional design of IA by incorporating several components and modules on the mobile devices to achieve energy and computational efficiency, lightweight authentication, dynamic sampling, and easy integration. In addition, we devise the WVM to further reduce the energy consumption and improve the accuracy of authentication. To evaluate our methods, we are conducting experiments on both the synthetic dataset and real dataset.

Chapter 4

PersonaIA

4.1 Introduction

In this research work, we design and build an implicit authentication system, *PersonaIA*, that automatically selects dynamic sets of features for user behavior extraction. To ensure practical deployment, we integrate lightweight computation and adaptive sampling into the system, making it energy-efficient for popular battery-powered devices such as smartphones, pads, and smart watches. There are several key features of *PersonaIA* that are desirable for a variety of applications, not limited to authentication. The ability to match a user's current behaviors with behavior history and differentiate them from other users' makes *PersonaIA* suitable for authentication, authorization, access control, and targeted advertising. *PersonaIA* can also detect a user's behavior deviation from the past which is most attractive to health-related applications such as assisted living, patient monitoring and fitness tracking. Moreover, it is possible to fine-tune the system to discover similar behaviors of a group of people that distinguish them from other groups, which supports emerging applications including membership applications such as building access and family-based sharing, as well as group-based recommendation applications such as group purchasing. Therefore, *PersonaIA* will be useful in a wide array of domains including cybersecurity, healthcare and fitness, education, online and mobile social networks, and e-commerce.

The uniqueness of *PersonaIA* is three-fold. First, *PersonaIA* uses dynamic sets of activities (called tokens in this thesis). The majority of existing research work relies on activities that are often fixed or static in type and does not select through the extracted behaviors to find the most desirable ones for different applications [63, 18, 19, 27, 36, 7, 88]. We select sets of available activities that characterize a person where the sets can be dynamically changing due to human behavior dynamics. Therefore, in addition to extracting the dominating behaviors (which are often common behaviors) of a person as in the existing work, our behavior modeling technique has the capability of discovering the most unique ones on the fly. Second, the challenges and complexity of *PersonaIA* extends far beyond behavior modeling. Although modeling human behavior is an important component in this work, it is not sufficient for the success of *PersonaIA* which also relies on properties such as energy efficiency, adaptability, and reliability. In other words, *PersonaIA* also faces challenges from system design and implementation. This requires the modeling technique is applied in a non-straightforward manner and combined seamlessly with other techniques we propose. For example, unlike the framework in [53, 54], *PersonaIA* is energy-efficient in that the client side employs computation outsourcing and adaptive sampling to keep the overhead at bay. Third, *PersonaIA* deals with the difficult problem of distinguishing between legitimate users' behavior deviation and illegitimate users' behavior which affects the accuracy and practicality of the system.

To successfully develop *PersonaIA*, we need to solve challenges related to modeling and system design. Modeling-related challenges are associated with finding a suitable machine learning technique for selecting and extracting dynamically changing user behaviors. We will need to first identify the requirements for the desirable behaviors in *PersonaIA* and their implication on system design, and then find an appropriate modeling technique and adjusting it for feature selection and behavior extraction. System-related challenges are associated with designing the system architecture of *PersonaIA*, and developing it into a reliable and lightweight system. We will need to avoid expensive computations and unreliable communications on the energy-constrained client side, and design a suitable behavior matching algorithm, the core of authentication, that improves system reliability by reducing the false negative and false positive rates. In this research work, modeling and system related challenges are mainly overcome by topic models and the novel Wind Vane Module (WVM).

We first present a high-level overview of our key idea: modeling, discovering, and selecting user behaviors, with an example, before diving into technical details. Ideally, the behaviors *PersonaIA* relies on should satisfy two requirements: *a) likely to occur for a person* and *b) not likely repeated by others*. The former ensures the behaviors can be observed when needed by the application, while the latter guarantees the unique identification of the person. Like popular machine learning algorithms such as SVM and Gaussian mixture model (GMM), topic models can discover behaviors satisfying *a)* in a relatively straightforward manner. Nonetheless, more efforts should be made to discover behaviors for Requirement *b)*, especially with behavior dynamics, which will be the focus of this work and addressed mainly by topic models. We use the following example to demonstrate the two requirements. The behaviors derived from "location" (departing and arriving stations), "time", and "speed of travel" are *likely to occur* for Caltrain commuters in the Bay Area. Nevertheless, these behaviors are not sufficiently distinctive to tell one Caltrain commuter from another, and we need to leverage other types of behaviors not related to commuting, e.g., playing Angry Bird on Caltrain, to satisfy Requirement *b)*. There are two behavior extraction phases in this project: the *reference extraction phase* that occurs at the server and the *real-time extraction phase* that takes place at the client. The server uses topic models to generate reference behaviors for each person. The reference behaviors guide the selection of desirable user activities for personalized authentication (i.e., behavior matching in the *real-time extraction phase*). The *reference extraction phase* can serve as feature selection for other machine learning techniques (not limited to topic models) to improve their behavior extraction results.

4.2 Related Work

Extracting user behavior through use of topic modeling is a key aspect of our work. Although initially designed for natural-language documents [17, 80], topic models have been applied to image, video, genetics, human activities and software debugging [31, 74, 47, 11]. There are some recent studies on modeling individual or group behaviors [47, 98, 59, 29, 102, 46]. Some of them are very limited in scope, e.g., only one person is considered in [47] and only location data is considered in [29]. Some merely showed that topic models could be the right tool for user behavior modeling without considering the system's aspects [46, 102]. These works all differ from our research in one or more of the following ways: *i)* Our research focuses on discovering distinctive user behaviors suitable for implicit fingerprinting based on dynamic activity types and data; *ii)* Applying topic models straightforwardly does not solve our problem. The success of *PersonaIA* relies on many other key mechanisms and techniques, e.g., WVM, re-training, dynamic privilege; and *iv)* Our research facilitates system reusability and extensibility by the design of a modular and generic architecture, supporting further research and development.

As well, our work relies on leveraging user behavior for IA. The majority of recent research work focuses on using behavioral biometrics such as touch, motion, shake, and arm swing, to design authentication applications for smartphones [63, 18, 19, 27, 93, 41, 49, 24]. The main difference between their work and our proposed research is two-fold. Their work is mostly concerned with using specific behavior(s) for authentication while ours is centered around finding out which behaviors most uniquely identify a person and using these behaviors for authentication. Their work features a fixed or static set of activities from which behaviors are derived, and our work features dynamic sets. Because of the fixed or static nature, their techniques are limited in one way or another whereas our approach is flexible and versatile. For example, quite a few papers study the touch behavior [63, 18, 19, 27, 93, 70, 85] which is specific to this generation smartphones. Their techniques may not work for smart watches and glasses, or even the next generation of smartphones, e.g., touch may not be the main way of operating future smartphones. In addition to these works, an IA framework for Android has been developed in [53, 54] allowing researchers to improve their authentication schemes. However, the framework implements the machine learning module in smartphones which can be impractical. The complex topic models in our system crashed on the smartphone in our experiment, which is the reason we propose a different approach by shifting the training and testing of topic models to the server. Some works allow the user to adjust sampling rate, e.g., [62], but essentially their methods, unlike ours, belong to static IA where the sampling rate is not dynamically controlled by the system. In addition to the differences mentioned above, these works differ from our research in one or several of the following ways: *i)* We explore a relatively new learning technique of dynamically selecting sets and sampling rate that are shown to be advantageous; *ii)* The WVM assists in energy-efficient behavior matching and improves the authentication success rate; and *iii)* We propose a finer-grained authentication mechanism which further enhances system reliability and user experience.

4.3 Background Knowledge

In this section, we introduce some background information on Jensen-Shannon (JS) divergence, partially labeled Dirichlet allocation (PLDA) and associated techniques.

4.3.1 Topic Models

Tokenization: Given a dataset containing various raw sensor data, tokenization is the task of chopping the data into pieces (tokens). At the same time tags are added to ambiguous tokens, and certain characters in the stop-word list are discarded. For example, in here, a GPS reading will be tokenized into the following: (loc69x, loc1333y), and a light reading will be tokenized into: (l98). We use the prefix to identify each sensor (Therefore loc = location and l = is light). The number attached to the label is the actual reading from the sensor. After tokenizing we are now left with a set of tokens. Each token is truncated according to its value(s). For example, the light meter reading l30.41 and l30.58 would belong to the same token "l30". The location reading of "Girard Ave, Philadelphia" is split into two distinct tokens "Philadelphia" and "Girard" by filtering out words according to the stop-word list, e.g., Ave.

Topic modeling: The topic modeling is a technique that uncovers the hidden thematic structure of document collections which in this work is the tokenized dataset. The details of topic modeling are shown in Table 4.1. A topic is a distribution of tokens with probabilities, e.g., in Table 4.1 (b), topic 01 contains loc69x, loc1333y, l98, ... with probability of 0.003, 0.003, 0.02, ... respectively. The topic modeling, from another aspect, is a process of feature selection in which features are represented by various tokens. The features/tokens in the topic of highest probability represent the most related behaviors that co-occur only with a certain user, e.g., in Table 4.1 (c), Topic 03 and its corresponding tokens are selected to represent behaviors of User 01. As shown in Table 4.1 (c), *behavior* used in topic modeling

Table 4.1: Topic modeling.

(a)

User	Topic distribution				
	01	02	03	...	N
01*	0.01	0.03	0.59	...	0.01
...	...	...	...	...	...
23	0.54	0.02	0.01	...	0.04

(b)

Topic	Token distribution				
	loc69x	*loc1333y*	*l98*	...	*vol407*
01	0.003	0.003	0.02	...	0.001
...	...	...	...	...	...
N	0.002	0.002	0.01	...	0.001

(c)

User	Topic†	Behaviors‡
01	03	$p_{vol401} = 0.4, p_{vol400} = 0.3, \dots$
02	20	$p_{batt94} = 0.6, p_{vol4070} = 0.4, \dots$
...	...	...
23	01	$p_{l98} = 0.1, p_{loc69x} = 0.05, \dots$

*Label "User XX" uniquely identifies each user. Topic† indicates the topic of the highest probability. Behaviors‡ are represented by token distributions, e.g., $p_{l98} = 0.1, p_{loc69x} = 0.05, \dots$.

is represented by a token distribution, which serves as a fingerprint in *PersonaIA*. Many statistical topic models are developed from the latent Dirichlet allocation (LDA) topic model [17], which has the total probability: $p(\varphi_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D}; \alpha, \beta)$

$$= \prod_{i=1}^{K} p(\varphi_i; \beta) \prod_{d=1}^{D} p(\theta_d; \alpha) \prod_{n=1}^{N} p(z_{d,n}|\theta_d) p(w_{d,n}|\varphi_{1:K}, z_{d,n}),$$

where φ_k indicates a distribution over each token in kth topic. θ_d indicates the topic proportions for the dth dataset. $\theta_{d,k}$ is the topic proportion for topic k in dataset d. $z_{d,n}$ indicates the topic assignment for the nth token in dataset d and $w_{d,n}$ is the nth token in dataset d. α is hyper parameter which controls the sparsity of a topic that is assigned to each dataset. β is similar to α which controls sparsity of per-dataset token distribution.

Partially labeled Dirichlet allocation (PLDA) is an extension of LDA topic modeling, which utilizes token frequency in the dataset to deduce a proper topic distribution for each label. As shown in Table 4.1 (a), different labels uniquely identify each user, and for each label PLDA [76] will generate a probability distribution of all predefined topics, e.g., user with label "User 01" contains Topic 01 to Topic N where the topic identifier N is predefined with probability p_N. As training is completed, a fine-tuned topic distribution is returned as a reference model, where the topic of highest probability is selected as the most probable topic to identify such user, as shown in Table 4.1 (c). Essentially, in the testing phase the real-time data captured by various sensors will be evaluated by the reference model returned by the training phase. A legitimate user is identified if the highest probability topic in the real-time data matches the most probable topic of the legitimate user, otherwise the current user will be identified as illegitimate. Note that for each topic there is an associated token distribution, as shown in Table 4.1 (b). The matching process is a comparison of two distributions, one from a reference behavior model and the other from real-time usage, utilizing Jensen-Shannon (JS) divergence. The relationship between PLDA and the WVM is two-fold. First, tuning various parameters/topics in PLDA requires term-frequency/token-frequency prepared by the WVM. Second, the WVM is responsible for authenticating the user before training is completed, while after PLDA returns a fine-tuned model the WVM will base authentication on the model. Further description of PLDA topic modeling and associated data processing phases will be discussed in Section 4.5.

4.3.2 Behaviors in IA

The key to appropriately apply topic models to behavior discovery is to draw analogies between our daily lives and text documents. Our daily lives can be considered as unlabeled text documents in topic models. Each document (daily activities) is a mixture of topics (behaviors), and each topic is a mixture of words (activities). Activities serve as the “vocabulary” of our daily lives whose hidden meanings are reflected by behaviors. Specifically, given the daily lives of different people, we can adjust the model to prefer distinctive behaviors over similar behaviors. The distinctive behaviors and their dominating activities will guide the activity selection, as well as sensor selection to reduce unnecessary sensing and energy consumption. The selected sets of activities are referred to as *dynamic sets*, because the activities that define a user’s implicit fingerprint are ever changing and depend on various environmental variables, such as the user’s current location. Using topic models for behavior extraction is not a new technique, and has been shown to have excellent results in previous works [47, 29, 98]. We define *distinctive behaviors* as the most desirable behaviors for individual fingerprinting. For the behavior to be considered distinctive, it must satisfy the following, *a) the behavior must frequently occur for a user* but *b) the behavior must infrequently occur for others*. The former ensures that a distinctive behavior can be observed while the latter guarantees the unique identification of an individual user. Discovering behaviors that satisfy *(a)* is relatively straightforward. However, satisfying requirement *(b)* when using dynamic sets of distinctive behaviors is a challenging task, and is the focus of this work. In addition, for each topic there exists a token distribution, and only the first 10% of tokens of highest probability are selected as main features for user identification. The *frequent behaviors* is defined as the subdataset that contains that 10% of tokens. To differentiate real human behaviors and *behaviors* used in topic modeling, we call the latter *behaviors* as *behavior pattern(s)*, but *behavior(s)* is also used in the thesis if it does not lead to ambiguity.

4.3.3 Jensen-Shannon Divergence

The common way of measuring similarity in between two distributions is by calculating the Kullback-Liebler (KL) divergence, which is defined as:

$$D_{KL}(\mathbf{X}||\mathbf{Y}) = \sum_{n=1}^{N} p(X=n) \log \frac{p(X=n)}{p(Y=n)}, \tag{4.1}$$

where $\mathbf{X}$ and $\mathbf{Y}$ are two distributions.

However, the KL divergence is not well bounded and it is an asymmetrical measurement, which is not suitable in building our system. To overcome these drawbacks we adopt Jensen-Shannon (JS) divergence [64], which is defined as:

$$D_{JS}(\mathbf{X}||\mathbf{Y}) = \frac{1}{2}[D_{KL}(\mathbf{X}||\mathbf{M}) + D_{KL}(\mathbf{Y}||\mathbf{M})], \tag{4.2}$$

where $\mathbf{M} = \frac{1}{2}(\mathbf{X} + \mathbf{Y})$. Particularly, for a distribution $\mathbf{X}$, $D_{JS}(\mathbf{X}||\mathbf{X}) = 0$; otherwise, $D_{JS}(\mathbf{X}||\mathbf{Y})$ will shift to 1 depending on how large of the difference between $\mathbf{X}$ and $\mathbf{Y}$, for example given $P_{\mathbf{X}} = [p_{X_1}, p_{X_2}] = [1,0], P_{\mathbf{Y}} = [p_{Y_1}, p_{Y_2}] = [0,1]$, $D_{JS}(\mathbf{X}||\mathbf{Y}) = 1$.

4.4 System Architecture and Adversary Model

4.4.1 System Architecture

With knowledge briefly described in the previous sections, we are ready to build *PersonaIA*. The system architecture, including the main modules and functional blocks, interfaces, and data flows, is shown in Fig. 4.1. *PersonaIA* can be separated into two main parts: client side and server side.

On the client side of *PersonaIA* sensor monitors are used for collecting raw behaviors such as $location = (loc69x + -20, loc1333y + -20)$ and $touch\_screen = Scroll$. These data are stored in *PersonaIA*'s Cache DB component (1), which is implemented using SQLite. The Cache DB module uses a pre-designed Interface for receiving data and facilitating the implementation of new sensors and data sources. The final module on the client side is the Behavior Matching Module, which we have named Wind Vane Module (WVM). The WVM is the main mechanism driving the novel dynamic adaptability of *PersonaIA* and is in charge of the real-time client-side authentication. The WVM captures user's real-time behaviors (8) and compares it with the legitimate user's historical behaviors captured and stored in local storage (hard drive for example). After comparison, a matching result will be returned that determines if the current user is a legitimate user. The Behavior Matching Module also controls the sampling frequency and fine-grained data uploading mechanism (9).

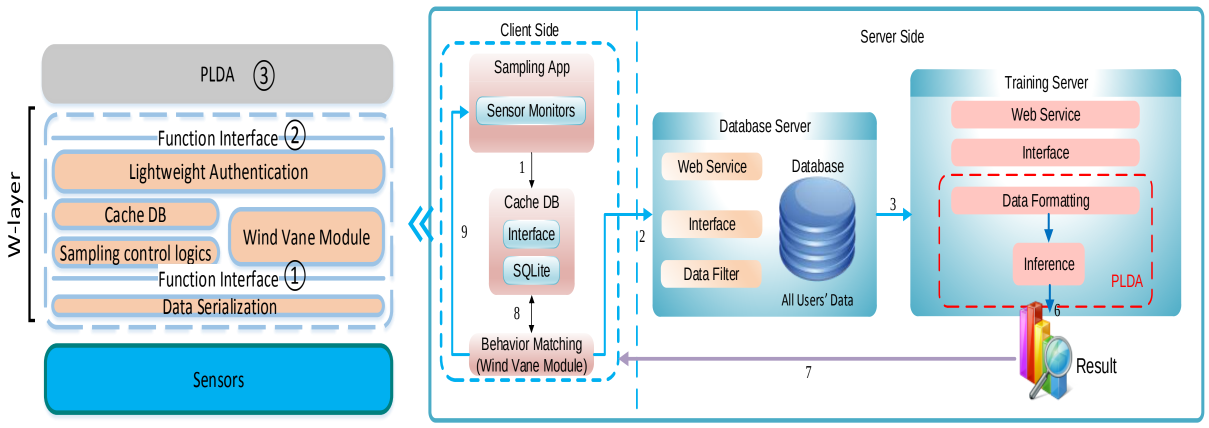


Figure 4.1: The system architecture of *PersonaIA*.

Raw data captured by sensors often contain noise and special characters, e.g., %, comma, dot, etc. We have built up a stop-word list to automatically preclude the most irrelevant characters and for some frequently used characters/tokens, e.g., android, app, etc. We also delete them from the Cache DB. After filtering out the noise the Behavior Matching Module will upload the fine-grained user's data to the database server through web services for further processing (2).

Finally, the Training Server, containing a Web Service, will receive each user's data (3) through the Interface and use this data to return an accurate authentication result (6) to the Client Side (7). The Training Server adopts PLDA (③) to generate a topic distribution for each user and finds out the most probable topic (main topic) for each user. Before the data is sent to PLDA for training, it is formatted to fit the requirements of PLDA. Using PLDA, the server side returns the corresponding behavior model to the client side, which measures the difference between the real-time behavior and the historical behavior model. The comparison between the two is used to adjust the WVM settings. The Interfaces are introduced to enhance the compatibility of system.

From another aspect data flows in *PersonaIA* can be explained as interactions between different layers shown in the left part of Fig. 4.1, while the W-layer can be viewed as a vessel to exchange the information between the remote server and the WVM. In the W-layer, the communication is enhanced by data serialization, in which data exported from the W-layer is serialized to a single string (different sensors will be attached to different labels). After the server receives the string, the server de-serializes it based on the respective label. The W-layer is also designed to send the raw data (①) and the fine-grained data (②) to the server through two interfaces. The server trains the model using the data provided by W-layer, and returns the result back to client. This result is used for more accurate authentication and adjusting WVM settings. In addition, the adaptiveness of *PersonaIA* is reflected in the following ways: *1)* the sampling frequency for each authentication, controlled by the WVM, is dynamically adjusted in which the legitimate user will have lower sampling frequency than illegitimate users; and *2)* for each authentication, the amount of sensor data used for behavior matching, is dynamically adjusted to reflect the behaviors of current user. The adaptiveness of *PersonaIA* is also related to the dynamic property of the behavior set.

The Cache DB, shown in Fig. 4.1, temporarily stores the sampling data. The data held in the Cache DB either uploads to the remote server or exports to a local machine for further processing. The smartphone's internal storage is much faster than the external storage, but is also very limited in size. In order for a successful implementation of an IA system, the authentication needs to happen in real-time. Therefore, it's integral that the identification unit is implemented using local storage. Also, the W-layer exports the entire dataset inside the Cache DB when the server requests data. This design severely reduces the transmission overhead and also increases the overall system robustness. For example, when the smartphone is offline the Cache DB will temporarily hold the most recent data, and send it to the server once the user is back online.

To support different frameworks, W-layer provides two interfaces that can be easily modified for remotely uploading and locally exporting data. We employ the outsourcing solution to migrate costly computations to the remote server. However, due to several limitations of this method, our system is also designed to support local IA using the WVM. By changing the exporting function within the "ExportFileTask" class, our system can achieve either remote or local exporting. The sampling control logics and lightweight authentication in the W-layer will be discussed in Section 3.5.

4.4.2 Adversary Model

We assume there are two different types of adversaries. One is the adversary that is unaware of the existence of *PersonaIA*, such as a person that finds a lost phone. We also consider a more powerful adversary, one who is aware that *PersonaIA* is on the phone or knows the behaviors of the legitimate user. An example of a regular adversary is a random person who finds the lost device and attempts to use it. An example of a more powerful adversary is a close friend, co-worker, or family member. Since this person is close to the person, they have a higher potential of emulating the legitimate user's behaviors.

4.5 Server-Side User Behavior Models

Through Function Interfaces ① and ② shown in Fig. 4.1, sensor data is uploaded to server, and the uploading process is controlled by the WVM. The topic modeling phase is shown in Table 4.1. After tokenization we calculate the Accumulative Behavioral Pattern (ABP) in the dataset selected by the WVM and use Gibbs sampling to approximate the topic and word distributions in the PLDA topic model [44, 76]. The token with high frequency, appearing only with a specific user, has a large chance to precipitate a major component of a user's most probable topic through Gibbs sampling. As shown in Table 4.1 (b) and (c), a topic contains tokens returned from the WVM. Intuitively, if a token, e.g., $batt94$ - battery reading 94%, appears frequently in dataset, it will have higher probability in that topic than the other infrequent tokens, e.g., $vol4000$ - battery voltage reading 4000 millivolt. Since PLDA uses Dirichlet distribution, for each user it generates a complete topic distribution containing all topics initially defined, as shown in Table 4.1 (a) and (b). The information about human behaviors is transformed to a set of token and topic distributions and the highest probability topic is selected as the most probable topic for identifying the user.

In this example, a dataset labeled by a specific user ID, e.g., User 01, is sent to PLDA for training. After training, usually done by using Gibbs sampling, a distribution of topics, ranged from Topic 01 to Topic N with probabilities p_1 to p_N, is returned as a fine-tuned model. In this example, the third topic, Topic 03, has the highest probability in the topic distribution of User 01 and is selected as the most probable topic to identify this user. The token distribution of this topic reflects the user's behaviors during the usage, as shown in Table 4.1 (c). It is not necessary for the token to be human readable since the WVM uses JS divergence in comparing distributions regardless of grammar and sentence structure.

4.6 Energy-Efficient Client Side

One of the core issues on the client side is making an energy-efficient method to verify authentication. If an IA application uses too much energy, it will not be feasible to distribute it on a large scale, e.g., many users. The client side's energy consumption is mainly due to the Behavior Matching Module. Therefore, behavior matching plays a key role in battery consumption. In addition to energy efficiency, we must consider other key issues that hinder realistic deployment on a large scale. This includes how to: initialize the system when there are no *reference behaviors model* from the server, handle behavior mismatches, and perform necessary interactions between the client and server.

Our solution to these problems is a novel Behavior Matching Module called Wind Vane Module (WVM). Wind Vane is an efficient algorithm that serves as a generic behavior matching mechanism to support the decision making of different applications, e.g., authentication, advertisement, education, etc. WVM is implemented on the client side of *PersonaIA*. Outsourcing the most expensive training part to the server side allows significant improvement to the energy efficiency of our Behavior Matching Module. The sensing and communication costs incurred by WVM are also low, since the sensing and communication frequency is reduced when the system is in a normal state (i.e., no unauthorized user is detected). Specifically, the *reference behaviors* from the server are updated by re-training much less frequently and do not need to be sent to the client in real-time in the normal state.

Chapter 5

BubbleMap and Reinforcement Authentication

5.1 Introduction

Rich behavioral data gathered by various sensors embedded in smart devices facilitates the implicit authentication (IA) of users based on their behaviors [105, 87, 62]. In general, IA systems authenticate users by matching their real-time behavior to their historical behavior. Real-time behavior is obtained from one or more sensors whose data can uniquely characterize the user and distinguish them from other users, at the time of authentication. Similarly, historical behavior is obtained from the same sensors in the past and updated after new data is collected. IA schemes typically run in the background and stream data at an appropriate frequency to ensure that data is sufficiently collected, and the battery consumption is reasonable. IA is a promising method of authentication since it generally does not require any form of explicit user actions as in password-, biometrics-, or token-based explicit authentication methods.

As with any other practical security system, IA systems need to strike a good balance between security and usability. On the one hand, we need the system to cope with the legitimate user's behavior deviation and noise [105], e.g., a change of routine, and not falsely rejecting the user (*usability*). On the other hand, the system needs to differentiate between a legitimate user's behavior deviation and illegitimate users' behaviors to prevent falsely allowing the adversaries to access the system (*security*). Compared to explicit authentication, IA more easily induces and is more susceptible to false negatives (falsely rejecting a legitimate user) and false positives (falsely accepting adversaries), due to the complexity of human behaviors and limitations of the machine learning algorithms used for extracting the user behavior model. It decreases the authentication accuracy IA systems are able to achieve and hinders the systems' wide deployment. In addition, since smart devices are a popular platform IA systems run on, energy consumption is an important consideration factor.

In this thesis, we propose BubbleMap (BMap), a framework to dynamically adjust and map users' privileges for accessing smart devices. This framework is independent of the machine learning algorithm used to implement an IA scheme and the features used, and can be adopted by any existing IA system [62, 93, 19, 32, 88, 42, 85, 21, 94, 93] as a plug-in to improve authentication accuracy as well as balancing between security and usability. We introduce intermediate privilege levels to the two-level (full access or no access) systems used by the existing IA schemes in the *Initial Mapping* step of BMap. An ideal IA scheme should always map the legitimate user to the top level (full access), and illegitimate users to the bottom level (no access). Nevertheless, when the legitimate user's behavior changes, it becomes harder to distinguish it from illegitimate users' behaviors. The intermediate privilege levels are incorporated in BMap to buffer the impact of behavior deviation so that the legitimate user still has access to some of the content (e.g., apps with lower security requirements), and therefore usability of the system is improved. It enhances security of the system as well in that an adversary will not likely be mapped to the top level immediately due to the buffering. However, the ultimate goal of BMap is to help the IA system quickly reach a definitive conclusion by mapping the legitimate user to the top level and illegitimate users to the bottom level. This is achieved by the remaining steps of BMap, *Privilege Movement* and *Bubble Expansion* in which questions including where to move the current user's privilege, and how fast and how much to move it are addressed to reflect users' dynamically changing behavior. As shown in our experiments, the time users spend in the intermediate levels is only 0.4% of their total usage time, and the authentication accuracy of the tested IA systems enhanced with BMap is universally improved. Main contributions of this work include:

• We design and develop BMap, and apply it to various state-of-the-art IA systems to boost their performance in terms of authentication accuracy, security, and usability. It is a plug-and-play that bridges the gap between IA research and the deployment of IA in practical systems.

• We evaluate the performances of the existing IA systems enhanced with BMap using large-scale comprehensive simulations and real-world experiments conducted over two years and eight months. We give quantitative results on the increase of authentication accuracy and analysis on improved security and usability. In addition, the energy consumption incurred by BMap is shown to be small.

5.2 Related Work

The majority of the existing implicit authentication schemes [62, 93, 19, 32, 88, 85, 21] focus on finding suitable behavioral features such as touch, typing, and other motions that uniquely identify users. The amount of data gathered by various sensors directly affects the accuracy of implicit authentication systems [62, 105, 87]. By increasing the time spent in collecting users' behavior data, the accuracy of implicit authentication can be improved [62, 87] with the cost of usability. In this work, we proposed the BubbleMap (BMap) framework to improve the authentication, accuracy, and usability of the original schemes at the same time. Dynamically adjusting privilege structure and absorbing the impact of various noises, BMap adds another layer of protection to implicit authentication systems, and is generally suitable for various schemes such as [87, 88, 62, 35, 77, 19, 15, 23, 26, 38, 85, 92, 95, 110].

To complement primary authentication mechanisms such as PIN and passlocks, various implicit authentication schemes have been proposed as secondary authentication mechanisms [23, 66, 92, 27, 32, 88, 62, 35, 19, 48, 67, 72, 79]. Among them, leveraging different features, Shi scheme [88], Multi-Sensor scheme [62], Gait scheme [35], and SilentSense scheme [19] are four different schemes that represent four research directions of state-of-the-art implicit authentications [52, 84]. In addition, current implicit authentication research tends to adopt all the available features to achieve a better authentication accuracy [62, 105, 104]. To evaluate the performance of BMap, we implemented Shi scheme, Multi-Sensor scheme, Gait scheme, and SilentSense scheme. We also show BMap can seamlessly cooperate with another framework such as [104, 105] to improve the system's performance.

BMap utilizes privilege control to dynamically adjust users' privilege. Privilege control mechanism has been widely used in different areas to enhance systems' security[82, 33, 108, 43]. Analyzed users' data, Eiji Hayashi et al [43] suggest to use multi-level authentication to improve the accuracy and usability of biometric-based authentication systems such as implicit authentication. However, due to the high complexity of human behaviors, the implementation of multi-level authentication in implicit authentication has not been seen. Implicit authentication mainly utilizes biometric behavior such as touch, motion, shake, and armswing, to identify users [70, 41, 24, 18, 39, 94, 63, 53, 51]. Since users' behaviors have large divergence and contain various noises [105, 106, 19], directly applying multi-level authentication to implicit authentication systems is not feasible. To this end, we analyzed the functionality of implicit authentication, mathematically modeled the privilege changing process in implicit authentication, and bridged a fine-grained privilege control to implicit authentication systems using BMap. In order to adopt sophisticated human behaviors, we upgraded the traditional fixed-level privilege control [82, 33, 108, 43, 43, 13, 25, 89] to support any number of privilege levels.

To deal with the behavior and sensor noises, most of the existing implicit authentication schemes use simple approaches such as resampling [62], averaging the results [19], or no approach at all [63, 53, 51]. Such noises will degrade system performance in terms of authentication accuracy. The problem will be exacerbated as the size of the behavior data grows. We applied a Kalman filter [99] to correct behavior deviation and filter out sensor noise during the authentication. We showed that a Kalman filter is naturally suitable for implicit authentication and can be implemented in practice to further improve authentication accuracy while reducing the system's latency.

5.3 Preliminary

In this section, we provide background information on machine learning classifiers, kernel density estimator, and Kalman filter. Among the machine learning algorithms tested, only the output of SVM needs to be converted to probabilistic values for BMap to handle, which will be discussed in Section 5.4.2. Kernel density estimator will be used to estimate the occurrence frequency of a particular behavior score in Section 5.4.4. Kalman filter filters out behavior data and sensor reading noises and will be discussed in Section 5.4.4.

5.3.1 Machine Learning Classifiers

SVM is the most widely adopted technique in IA systems [36, 63, 19, 62, 18, 106, 53, 39, 9]. Given a training dataset sampled from a group of people, SVM outputs a hyperplane located in a high dimensional space to cluster the data into two classes, the legitimate class and the illegitimate class. During authentication, new data sampled from the current user is verified according to its position in the hyperplane. The user is deemed legitimate if the new data falls in the legitimate class. In the proposed BMap, we need to calculate the distance between the hyperplane and the testing result in a high dimensional space which renders it difficult with SVM's traditional output. Instead, we leverage the probability output calculated by fitting a sigmoid function, $\frac{1}{1+exp(Af_i+B)}$, to the margins of the SVM [75], where A and B are the parameters to estimate and f_i denotes the margins of the SVM output. The probability output of SVM is referred to as behavior score in this work. Behavior scores represent a user's behavior in the numeric form and are used by the system to deduce a user's legitimacy. Other classifiers such as Gaussian mixture model (GMM) and statistical topic model can be directly used in BMap without additional transformation, since they output probability results.

5.3.2 Kernel Density Estimator

Kernel density estimator [83, Measure, 16] serves as a tool to analyze the usage pattern of the IA system, e.g., legitimate and illegitimate usages in a given time interval, by estimating how often a given behavior score occurs. This is necessary in distinguishing between the legitimate user's deviated behaviors and illegitimate users' behaviors. Kernel density estimator divides the interval into small bins with length h, in each of which it calculates the number of behavior scores that fall into the bin. A distribution of the behavior scores is obtained by placing a Gaussian over each score and then adding up the contributions over the whole dataset. The kernel density model is $p(x) = \frac{1}{N}\sum_{n=1}^{1}\frac{1}{(2\pi h^2)^{D/2}}exp-\frac{||x-x_n||^2}{2h^2}$, where D denotes $D-dimensional\ space$, N is the total number of behavior scores, x_n is an individual behavior score and x denotes the center of each bin. The kernel density estimator is used in *Bubble Expansion* to adjust the distance of the movement $+\mu$ and $-\mu$.

5.3.3 Kalman Filter

Kalman filter [50] is employed in BMap to filter out sensor noise and help correct behavior deviation. The two types of noises it assumes, process noise and observation noise, can be used to model behavior deviation (or behavior noise) and sensor noise, respectively, making it an excellent tool for noise filtering in IA systems. In addition, Kalman filter is loop carried which means it automatically filters out noises at the time of authentication, rather than needing more data to perform the filtering as in the existing literature [19, 36, 63, 7, 28, 61, 97]. This property greatly reduces the system's latency.

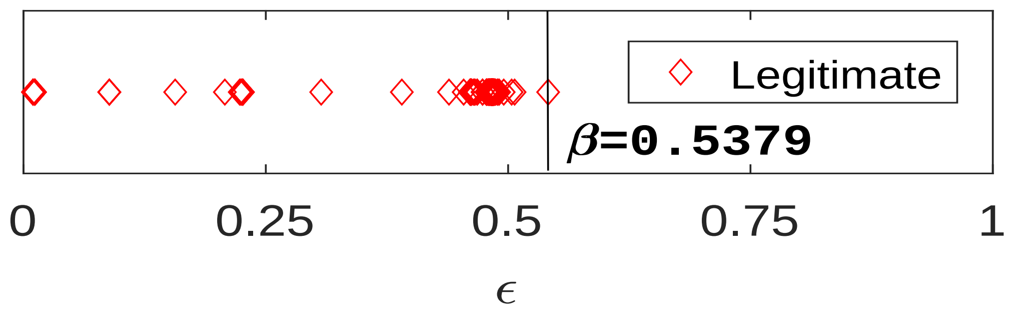
5.4 The Proposed BMap Framework

We first provide an overview of the BMap framework and then elaborate on the detailed design.

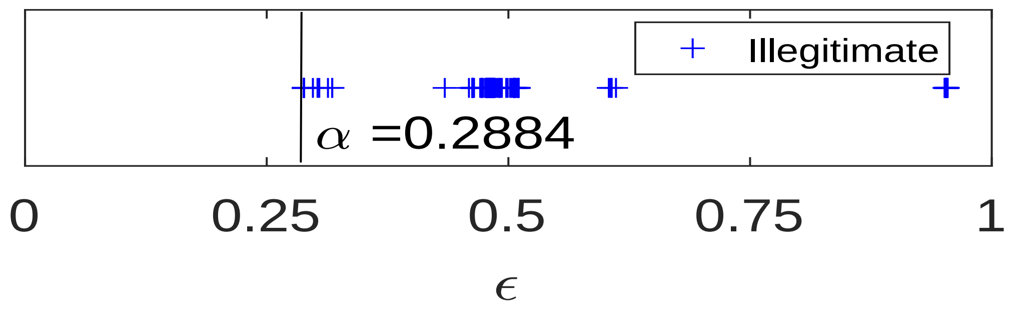
5.4.1 System Overview

Generally speaking, existing IA schemes authenticate users by deriving a behavior score ϵ using data gathered in a period of time, called time window (or authentication cycle) which is a design-specific parameter. The score ϵ is then compared with a threshold, e.g., 0.5, and if the threshold is exceeded, the system concludes that the current user is illegitimate and locks the device. When legitimate and illegitimate users have vastly different behaviors, existing IA schemes can achieve high authentication accuracy [1]. However, based on our

[1]The accuracy of the identification is calculated by $ACC = \frac{TR+TA}{TR+TA+FR+FA}$, where the true accept (TA) denotes a legitimate user's data sample has been correctly identified, otherwise denoted by false accept (FA), and the false reject (FR) denotes a legitimate user's data sample has been incorrectly identified to be illegitimate users' data sample, otherwise denoted by true accept (TR).



(a)



(b)

Figure 5.1: Behavior scores of (a) a legitimate user, (b) an illegitimate user.

preliminary simulations using the Friends and Family dataset [5, 8], more than 70% of users' behavior data overlaps and cannot be separated by simply setting a threshold. As a simple example, we randomly selected two participants from the dataset, one as the legitimate user and the other as the illegitimate user, and converted the system's output to probabilistic behavior scores. The time window is set to 15 seconds. As shown in Fig. 5.1 (a) and (b), the legitimate and illegitimate users both have a large proportion of behavior scores located around the threshold 0.5 which makes them inseparable. The behavior overlapping problem can be exacerbated by mimicry attacks where the adversary imitates the legitimate user's behaviors [56]. BMap attempts to improve authentication accuracy even in the presence of this problem, by using the proposed *Initial Mapping* , *Privilege Movement* and *Bubble Expansion*, which will be discussed in what follows.

On a high level, in *Initial Mapping*, three bubbles, legitimate bubble, slack bubble, and illegitimate bubble are created by defining their boundary values α and β based on the legitimate user's historical behavior data. Each bubble contains different privilege levels representing access rights to apps of different security levels. *Privilege Movement* then determines where to move the user's privilege level based on his current behavior. *Bubble Expansion* is used to fine tune the bubble sizes defined in *Initial Mapping* and how far the privilege level should be moved in *Privilege Movement*. It reduces the impact of noisy data and behavior deviation and therefore improves authentication accuracy. Specifically, the first step in BMap is to map multiple privilege levels to the three bubbles using *Initial Mapping*. We add a few intermediate privilege levels to the two-level (full access or no access) systems used by the existing IA schemes. Apps are categorized based on their security requirements and mapped to privilege levels. For instance, in a system with n ($n \geq 3$) privilege levels R_1 through R_n, apps can be mapped to the levels as shown in Fig. 5.2. Apps with the highest security requirements such as banking, e-commerce, health and fitness, credit score, and password manager are mapped to the highest privilege level R_1. Apps with lower security requirements such as social media, texting, games, and utility apps are mapped to lower levels such as R_2, R_3, R_{n-1}. R_n is the lowest privilege level which corresponds to locking the device no access. The legitimate bubble, slack bubble, and illegitimate bubble contain the top level R_1, the intermediate levels R_2 to R_{n-1}, and the bottom level R_n, respectively. Note that defining the privilege levels, the security requirements for the apps, and their correspondence is system and user dependent. It is relevant but not a focus of this work. Interested readers are referred to [82, 33, 108, 43] for more details. After obtaining the privilege levels, the system needs to map the user to a specific level R_c based on the user's current behavior at the time of authentication. The level R_c is called the user's current level as shown in Fig. 5.2. This is performed in the second step of BMap, *Privilege Movement*. Once in this level, the user has access to all the apps corresponding to R_c and the levels below R_c, but not the levels above R_c. Moreover, overlapping behaviors are effectively separated in this step. Finally, *Bubble Expansion* is used to dynamically adjust the privilege boundaries as more behavior data becomes available and to filter out behavior and sensor noises.

Adversaries in BMap BMap defends against password guessing attacks and behavior mimicry attacks (adversary imitating the legitimate user's behavior). In password guessing attacks, we assume that it takes the adversary a few tries (exceeding the limit) to guess and input the password correctly. Likewise, it requires some time for the adversary to fully mimic the legitimate user's behavior[56]. We do not consider an adversary who can enter

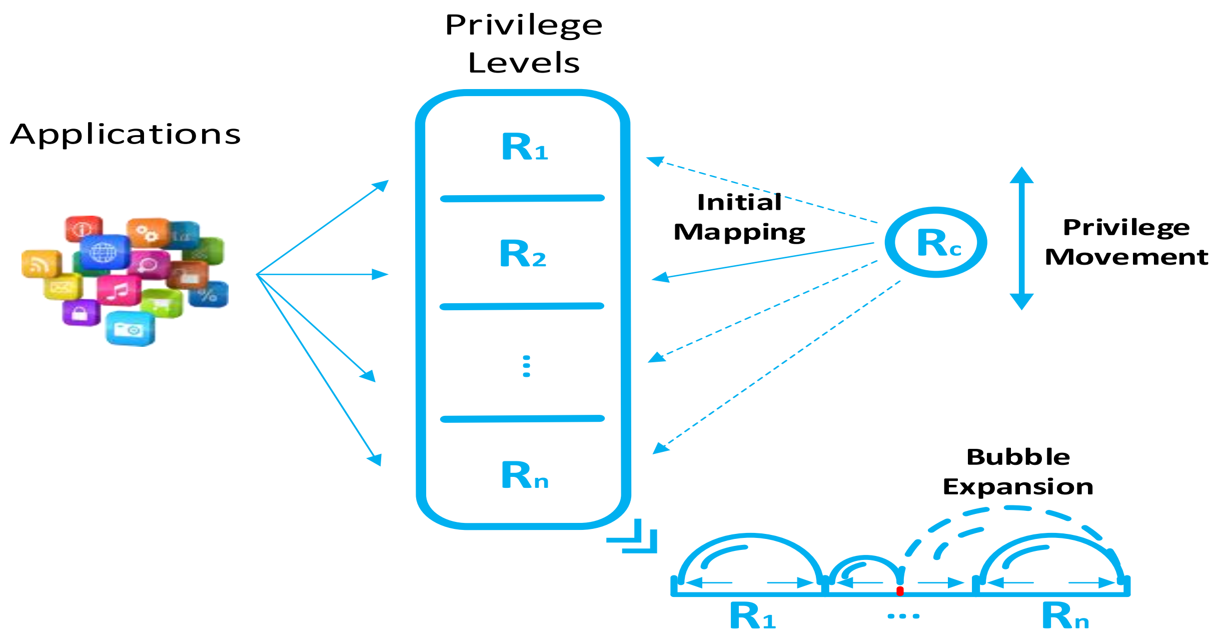


Figure 5.2: The BMap Framework.

the password correctly within the trial limit because this is a general problem common to all existing authentication systems using password. Without loss of generality, we assume the use of a backup authentication mechanism to count for when the IA fails, similar to how a passcode is required (with limited tries) to unlock an iPhone X when Face ID fails. A password guessing adversary may eventually correctly guess and input the passcode. But due to *Bubble Expansion*, every wrong password input accelerates the expansion of the illegitimate bubble causing the system to quickly map the adversary to the bottom level, thus reducing false positives. Since mimicry attacks also require launch time [56], BMap defends against them in a similar manner. BMap also reduces false negatives by quickly mapping the legitimate user's privilege back to the top level once a correct passcode is entered within reasonable tries. We use the terms illegitimate user and adversary interchangeably.

The passcode used to unlock the device is not to be confused with the PIN number which the user can enter to re-initialize the system and retrain the machine learning model based on new behavior data[106]. This can be a useful feature when the system keeps failing the legitimate user's authentication and needs calibration.

In this work, we use a subset of all available features in the Friends and Family dataset, i.e., GPS, accelerometer, touch, SMS, app installation, battery usage, call logs, app usage, bluetooth devices log, and Wi-Fi access points to evaluate the existing IA schemes. For example, we use the accelerometer data for Gait [77] and the touch and accelerometer data for SilentSense [19]. In order to give a fair comparison, we applied BMap to four state-of-the-art IA schemes, Shi et al. [88], Multi-sensor [62], Gait [77], and SilentSense [19], by strictly following the feature selection and parameter tuning process in these schemes.

5.4.2 Initial Mapping

We mainly discuss applying BMap to SVM-based IA schemes [62, 77, 19]. For the other IA schemes [49, 19, 105, 88], since their output is already a probabilistic behavior score, BMap can be directly applied.

DEFINITION 1. *Let behavior score $\epsilon \in [0, 1]$ denote the probabilistic output of an SVM approximated by a two-parameter sigmoid function $\frac{1}{1+exp(Af_i+B)}$. In a specific training set* [2]*, we further divide the interval [0, 1] into n sub-intervals, called bubbles, denoted by $D_n \subset [0, 1]$. The legitimate bubble is the largest sub-interval that contains only true accept (TA) behavior scores. The illegitimate bubble is the largest sub-interval that contains only true reject (TR) behavior scores. The slack bubble is the sub-interval in between the legitimate bubble and illegitimate bubble.*

The *Initial Mapping* mechanism is illustrated in Fig. 5.3. The system first initializes the value of parameters α and β by fitting the sigmoid function to the SVM output trained by data sampled from legitimate and illegitimate users. Note that the distance between α and β can be very small, e.g., 0.01, but they never collide. The legitimate, slack, and illegitimate bubbles are blown based on these two parameters, in which only the legitimate bubble can explode. Assuming the system has n privilege levels, in each authentication cycle as new data is collected, the SVM takes the data as input and outputs a new behavior score indicating the system's authentication decision. If the new score falls in the legitimate

[2] A training set is a dataset that contains various users' historical behavioral data.

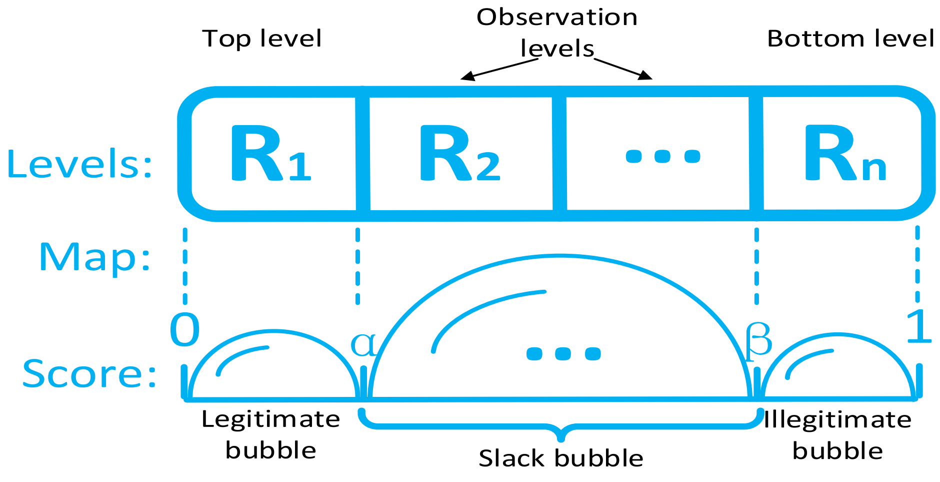


Figure 5.3: Initial mapping.

bubble, the system will move the user's current privilege level R_c to R_1 (if $R_c \neq R_1$) which grants the user full access. If the new score falls in the illegitimate bubble, the system will lock the device. If the new score falls in the slack bubble, the system will map R_c to one of the observation levels R_2, R_3, ..., R_{n-1}, where the user has only limited access.

As shown in Fig. 5.1 (a) and (b), the legitimate and illegitimate bubbles are $[0,\ \alpha]$ and $[\beta,\ 1]$, respectively. The slack bubble is located in $[\alpha,\ \beta]$, which contains ambiguous behavior scores that could come from either the legitimate user or illegitimate users and need separation. In a given dataset, we can easily find α and β by searching for the largest and smallest behavior score ϵ derived from the legitimate user's and illegitimate users' training data, respectively. In *Initial Mapping*, we first assume that α and β are fixed and focus on the mapping of the current privilege level R_c to one of the observation levels in the slack bubble. We then release this assumption in Section 5.4.4 when we complete our discussion with the possible movement of the bubble boundaries. Compared to the existing implicit authentication schemes, *Initial Mapping* in BMap focuses on both security and usability. Since the system only grants full access to the user who is most likely to be legitimate, security is enhanced. When the likelihood declines, instead of completely locking the user out, the system maps the user to an observation level that grants lower access rights. It enhances usability if the user is legitimate while limiting the security breach if the user is illegitimate. Nevertheless, *Initial Mapping* only handles failed authentications in a more gradual way by adding the slack bubble and observation levels. It does not fundamentally ameliorate the false reject (FR) and false accept (FA) performance, which will be the focus of *Privilege Movement* and *Bubble Expansion*.

5.4.3 Privilege Movement

In *Initial Mapping*, the current privilege R_c is mapped to one of the defined privilege levels $[R_1, R_2, ..., R_n]$ when a new behavior score becomes available at the time of authentication and remains in that level until more data comes in. Such a mapping mechanism does not fundamentally improve the FR and FA performance since the system still needs a way to confirm the user's legitimacy once her behavior score is mapped to the uncertain observation level. Recall that the system's goal is to eventually grant the user full access if she is legitimate and lock her out otherwise. The slack bubble is just a buffer for a smoother transition. We introduce *Privilege Movement* in the mapping of R_c, where R_c is moved up (towards R_1) or down (towards R_n) gradually out of the slack bubble. We assume that the implicit authentication scheme gives high authentication accuracy, i.e., the legitimate and illegitimate users' behavior scores fall into their corresponding bubbles rather than the slack bubble, when the scheme is newly trained.

We summarize *Privilege Movement* mechanism in Fig. 5.4. The system keeps track of the user's behaviors and once it observes a behavior score that falls into the slack bubble, it searches through the previous scores to find a more definitive answer. If there were scores in the legitimate bubble, the system leans towards regarding the user as legitimate and moves R_c upward with distance $-\mu_l$ at the end of the current authentication cycle. This process is repeated until R_c reaches R_1. Similarly, if there were scores in the illegitimate bubble, the system leans towards regarding the user as illegitimate and moves R_c downward with distance $+\mu_a$ at the end of the current authentication cycle. This process is repeated until R_c reaches R_n. If R_c falls in between privilege levels, the user is assumed access privilege of the lower level. The movement distances $-\mu_l$ and $+\mu_a$ are design parameters that can be constants or variables. For the discussion in this subsection, we let $\mu_l = l/2$ and $\mu_a = l$

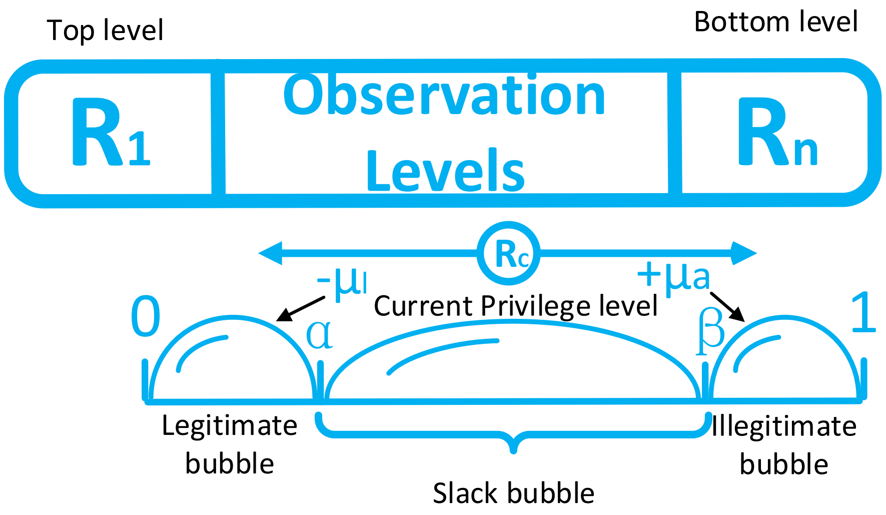


Figure 5.4: Privilege movement.

where l is the fixed distance between privilege levels. The system is thus less tolerable and more restrictive when there is evidence that the current user is illegitimate. It is also more conservative in giving the user higher access privilege when the user's legitimacy was confirmed in the past but is currently in doubt. Such design is to enhance security while not sacrificing usability. Moreover, the FR and FA performance is improved since the system always tries to move R_c out of the slack bubble based on evidence. *Privilege Movement* mechanism has $O(1)$ time complexity, which renders the system's latency the same as the implicit authentication schemes without BMap. In the next subsection, we discuss making μ_l and μ_a variables to improve authentication accuracy.

As an example, the behavior score distribution for the legitimate user and illegitimate user is shown in Fig. 5.5 (a) and (b), respectively, using the aforementioned simulation with two participants (Fig. 5.1 in Section 5.4.1). The scores are grouped into five one-hour time slots, where each time slot contains multiple time windows. In each time slot, there are behavior scores belonging to the legitimate/illegitimate bubble that co-occur with scores belonging to the slack bubble. The scores that belong to the legitimate/illegitimate bubble are used as evidence and guidance to move the scores in the slack bubble. When behavior deviation happens, *Initial Mapping* may map the legitimate user to the observation level

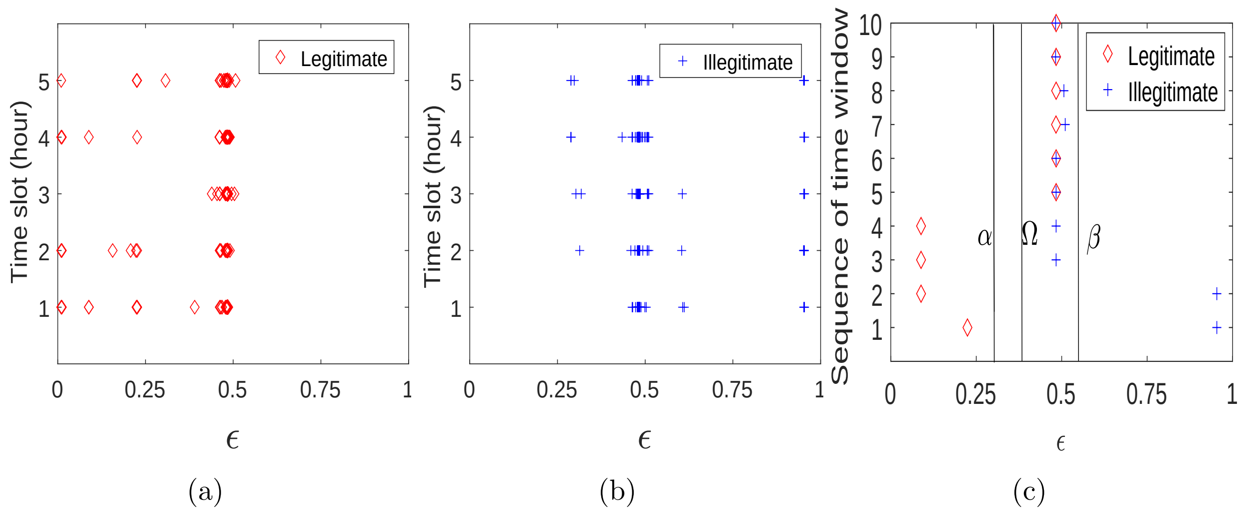


Figure 5.5: Behavior scores of (a) legitimate user in a 5-hour period, (b) illegitimate users in a 5-hour period, and (c) both users in 10 time windows.

and still cause false rejects which are corrected with *Privilege Movement*. The same is true for false accepts. In addition, we randomly selected a time slot from Fig. 5.5 (a) and (b), and magnified it in Fig. 5.5 (c) where the threshold Ω is predefined to best separate the two users. For the ease of presentation, we assume that there is only one observation level and three privilege levels in total. In the first time window, the legitimate user's behavior score falls in the legitimate bubble (shown in the figure) but her R_c has not reached R_1 (not shown in the figure). The system therefore moves R_c upward for $l/2$. In the second through fourth time windows, the score falls in the legitimate bubble again but R_c has reached R_1. So R_c remains in R_1. In the fifth through tenth time windows, R_c falls in the slack bubble. Since the system observed four behavior scores in the legitimate bubble, R_c remains in R_1. If the system observed scores in the illegitimate bubble instead, R_c would have been moved towards R_n. The illegitimate user in Fig. 5.5 (c) follows a similar Privilege Movement process. Using the dataset [5], we were able to observe the co-occurrence of legitimate/illegitimate-bubble behavior scores and slack-bubble behavior scores for the same user in a reasonably short period of time (2-3 minutes), in all of the two-participant simulations we conducted.

The effectiveness of *Privilege Movement* is highly dependent on the size of the legitimate and illegitimate bubbles. If α and β are fixed, they may become less indicative as more behavior data from either the legitimate user or illegitimate users become available. This problem will be addressed in the *Bubble Expansion* mechanism where the size of the bubbles is dynamically adjusted to reflect the behavior change and improve the authentication accuracy.

5.4.4 Bubble Expansion

We now introduce *Bubble Expansion*, in which the bubble boundaries α and β are updated. In practice, due to behavior deviation and sensor noise, the initial setting of α and β may become inaccurate. If behavior scores from the legitimate user keep falling in the slack bubble, it may indicate that the legitimate bubble is too small and more air is needed to reduce false rejects. Similarly, the illegitimate bubble may need to be expanded to reduce false accepts. Authentication accuracy is improved as a result. As shown in Fig. 5.6, the original legitimate and illegitimate bubbles are $[0,\ \alpha]$ and $[\beta,\ 1]$, respectively. The new bubbles become $[0,\ \alpha']$ and $[\beta',\ 1]$ after expansion. In addition, the system's latency is reduced since less *Privilege Movement* is needed and the system can make decisions more quickly.

In a given dataset, it is straightforward to find out whether the behavior scores that keep falling in the slack bubble belong to the legitimate user. In reality however, it is difficult for the system to know in which case the second-factor authentication (password input for our discussion) is needed to provide feedback, as previously mentioned. We assume that the legitimate user will input the correct password and illegitimate users will input incorrect passwords at the beginning of usage. Although illegitimate users can guess passwords, after several unsuccessful tries the chance that illegitimate users are locked out is increased exponentially due to illegitimate bubble expansion. Similarly, an attacker can also mimic legitimate users' behavior, but it also requires time [56]. Due to illegitimate bubble expansion, compared to original schemes, attackers will have a larger chance of being blocked before they fully mimic legitimate users' behavior. For this reason, the true reject rate and the system's security are increased. Correspondingly, due to legitimate bubble expansion, legitimate users who input correct passwords at the beginning of usage will have a larger chance of being mapped to the top privilege level. For this reason, the true accept rate and the system's usability are increased.

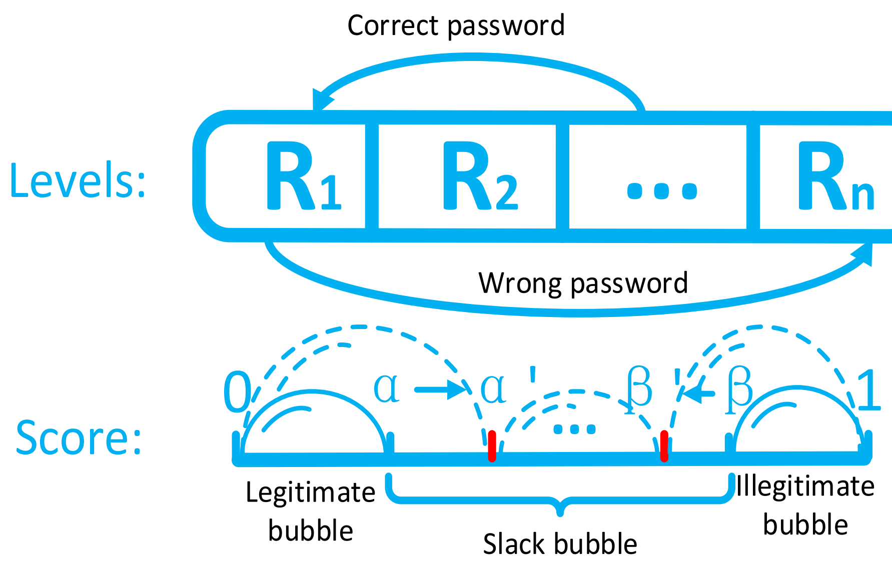


Figure 5.6: Bubble expansion.

We model *Bubble Expansion* by applying physical laws that describe the motion of bodies under the influence of a system of forces. Specifically, the expansion S in time t is defined as:

$$S = \frac{1}{2}(a - \hat{a})t^2 + v_0 t, \tag{5.1}$$

where a denotes the acceleration of the expansion, t denotes the number of time windows or authentication cycles, v_0 denotes the initial velocity of the expansion, and $\hat{a}$ is the resistance that slows down or stops the expansion. Every time the user inputs the correct password and the behavior score is outside of the legitimate bubble, more air will be blown into legitimate bubble, and expand it to contain the behavior score where the expansion is proportional to the distance between the behavior score and legitimate bubble $(\epsilon - \alpha)$.

The acceleration of the expansion a is defined as:

$$a = \frac{R_d * \varepsilon}{W_1} + W_2 + \delta, \tag{5.2}$$

where $W_1 = \frac{\sum_i n_l^{(i)} + n_a^{(i)}}{\sum_i N^{(i)}}$ is a balancing parameter that controls the expansion, W_2 is a constant representing the initial acceleration, $\sum_i n_l^{(i)}$ is the number of times the user inputs the correct password when her score is in the slack bubble, $\sum_i n_a^{(i)}$ is the number of times the user inputs a wrong password when her score is in the slack bubble, $\sum_i N^{(i)}$ is the total number of authentication cycles, R_d is the distance between R_c and R_1, $\varepsilon = \epsilon - \alpha$ is the distance between the behavior score and legitimate bubble, and δ is the mixture of behavior noise and sensor noise.

The expansion of the legitimate bubble may result in the inclusion of illegitimate users' behavior scores that originally fall in the slack bubble. To reduce such false accepts, we introduce the resistance $\hat{a}$ that constrains the expansion:

$$\hat{a} = a(\int_0^\alpha p(\varepsilon_a) d\varepsilon_a + \theta), \tag{5.3}$$

where θ is a constant that prevents α from surpassing β, $\int_0^\alpha p(\varepsilon_a) d\varepsilon_a$ denotes the probability that the legitimate bubble contains behavior scores derived from illegitimate users in the training set, and ε_a denotes the behavior score derived from illegitimate users' data in the training set. $\int_0^\alpha p(\varepsilon_a) d\varepsilon_a$ is estimated using kernel density estimator.

Substituting (5.8) into (5.1) and assuming $t = 1$, we have

$$S = \frac{1}{2} a(1 - \int_0^\alpha p(\varepsilon_a) d\varepsilon_a - \theta) + v_0, \tag{5.4}$$

where we let $V = 1 - \int_0^\alpha p(\varepsilon_a) d\varepsilon_a - \theta$, called fluid viscosity, control when the expansion stops.

Substituting 5.6 into 5.4, we have

$$S = \frac{1}{2}(\frac{R_d * \varepsilon}{W_1} + W_2)V + v_0 + \Delta, \tag{5.5}$$

where $\Delta = \frac{V * \delta}{2}$ is estimated and eliminated using a Kalman filter.

In each authentication cycle, if the user inputs the correct password, the predicted state estimate $x_{k|k-1}$ which controls the expansion of the legitimate bubble is defined as: $x_{k|k-1} =$

$F_k x_{k-1|k-1} + B_k u_k$, where $F_k = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix}$, $B_k = \begin{bmatrix} \frac{t^2}{2} \\ t \end{bmatrix}$ and $u_k = (\frac{R_d * \varepsilon_a}{W_1} + W_2)V$. The predicted estimate covariance $P_{k|k-1}$ is defined as: $P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k$, where the process noise covariance is $Q_k = \begin{bmatrix} \frac{t^4}{4} & \frac{t^3}{2} \\ \frac{t^3}{2} & t^2 \end{bmatrix} * \sigma_a^2$ with σ_a being the magnitude of the process noise (behavior noise). The innovation covariance is $S_k = H_k P_{k|k-1} H_k^T + R_k$, where $H_k = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and R_k is the covariance of the observation noise (sensor noise). Kalman gain is calculated as: $K_k = P_{k|k-1} H_k^T S_k^{-1}$. Since a Kalman filter is loop carried, we update the state estimate and associated covariance at the end of each authentication cycle as: $x_{k|k} = x_{k|k-1} + K_k(z_k - H_k x_{k|k-1})$, and $P_{k|k} = (I - K_k H_k) P_{k|k-1}$. We calculate the expansion as $P_{k|k} H_k$ and need to rescale it before applying it to real systems.

If the user inputs a wrong password, we let $u_k = \frac{\varepsilon_l}{R_d * W_1} + W_2$, and a similar process happens for the expansion of the illegitimate bubble. Furthermore, to defend password guessing, if users continuously input wrong password m times, the legitimate bubble will explode until users re-blow it by passing the hidden factor authentication discussed in Section 5.4.1. Instead, the illegitimate bubble will keep expanding and finally cause the legitimate bubble to shrink. Note that the bubble boundaries α and β never collide, since the slack bubble could become very small, e.g., with length of 0.01, but it never explodes.

In addition to causing false accepts, the expansion of the legitimate bubble also affects *Privilege Movement*, or more specifically, the distance of the movement $-\mu_l$ and $+\mu_a$. Now that the bubble boundaries α and β are dynamically adjustable, the distance of the movement needs to be adjusted accordingly. We let $-\mu_l = -\mu_l \frac{\int_0^\alpha p(\varepsilon_l) d\varepsilon_l}{\int_0^\alpha p(\varepsilon_a) d\varepsilon_a}$ and $+\mu_a = +\mu_a \frac{\int_\beta^1 p(\varepsilon_a) d\varepsilon_a}{\int_\beta^1 p(\varepsilon_l) d\varepsilon_l}$, where ε_l and ε_a denote the behavior scores derived from the legitimate user's and illegitimate users' data in the training set, respectively, $\int_0^\alpha p(\varepsilon_l) d\varepsilon_l$ and $\int_0^\alpha p(\varepsilon_a) d\varepsilon_a$ denote the probabilities that the legitimate bubble contains behavior scores derived from the legitimate user's and illegitimate users' data in the training set, respectively, and $\int_\beta^1 p(\varepsilon_l) d\varepsilon_l$ and $\int_\beta^1 p(\varepsilon_a) d\varepsilon_a$ denote the probabilities that the illegitimate bubble contains behavior scores derived from the legitimate user's and illegitimate users' data in the training set, respectively.

If the ratio $\frac{\int_0^\alpha p(\varepsilon_l)d\varepsilon_l}{\int_0^\alpha p(\varepsilon_a)d\varepsilon_a}$ is large, it indicates that the legitimate user's behavior scores still dominate the legitimate bubble, and the distance of *Privilege Movement* is appropriate. Otherwise, the distance needs to be adjusted.

5.5 The Proposed RA Framework

This section begins by introducing reinforcement authentication. It then shows the impact of behavior deviation. Finally, it describes the *human-centered feature selection* and *behavioral data alignment*.

5.5.1 Framework Overview

The overview of reinforcement authentication is shown in Fig. 5.7. In the beginning, users' behavioral data sampled by various sensors is temporarily stored in the smart device, which will package the data and send them to remote servers through a secured channel for further processing. In the remote server data derived from different users are labeled using a unique id. The server then uses a personal feature set to train machine learning models, e.g., SVM, GMM, topic model, and kNN. After training a fine-tuned model is returned. The smart device can use the model to authenticate users.

The feature selection of reinforcement authentication (RA) has two-fold. The first fold feature selection is identical to implicit authentication, which selects a set of features that best separate a group of users. In this feature set, the accuracy differences between various features might have large diversity. Among them, some of the features might have similar accuracy, e.g., ± 0.1% accuracy difference, but the others might not. To construct the feature list, RA utilizes the features that have similar accuracy, while other features are denoted as system reserved features. As shown in Fig. 5.7, the second fold (human-centered) feature selection is performed by users. In *human-centered feature selection*, the system sends feature list to users through a secured channel. Later on, users reply to the system by choosing the features that best represent their routine behavior. The system then combines the replied features and reserved features as a new feature set to identify users. As mentioned earlier, to successfully identify a user, only a small portion of total features is

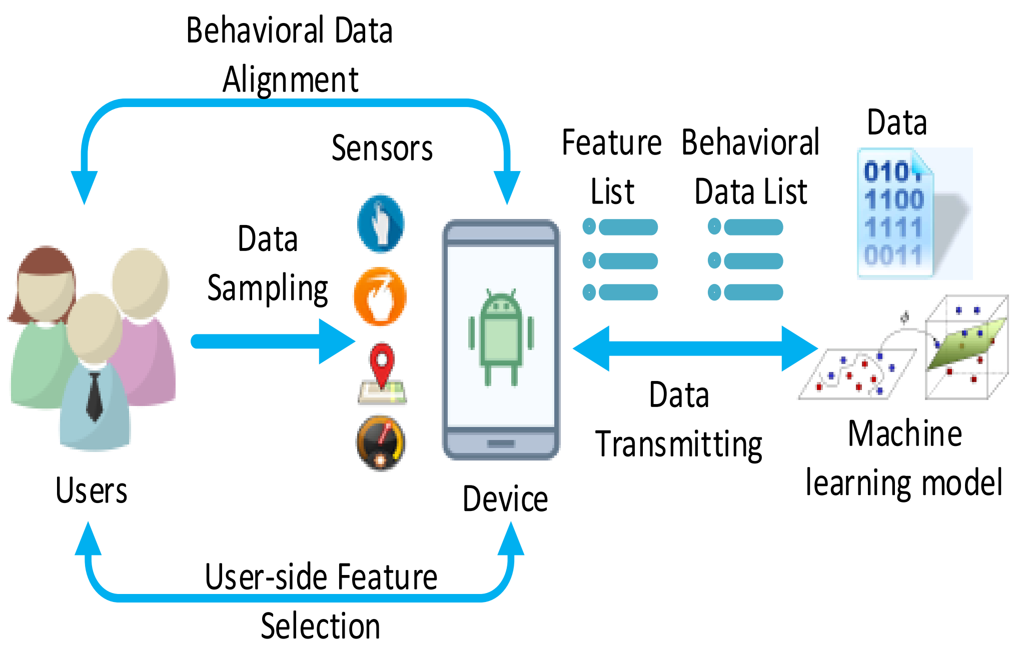
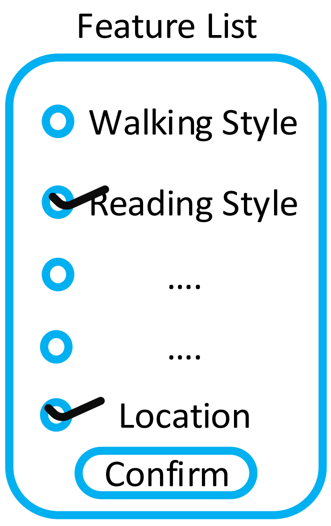


Figure 5.7: Reinforcement authentication (RA).

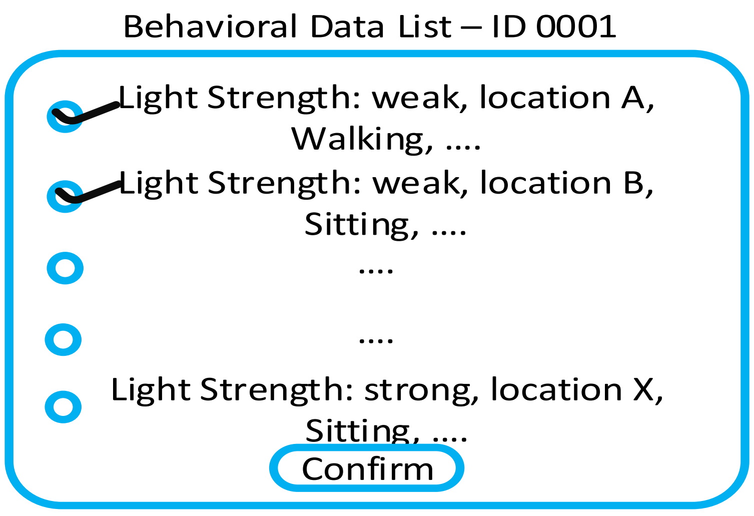
needed. Redundant features may reduce the performance of the system. Inputting a small amount of energy, RA utilizes *human-centered feature selection* to pinpoint a personal feature set and eliminate redundant features. In return, the total energy and time consumption are reduced, especially for the training and testing phases. The detail of *human-centered feature selection* is described in Section 5.5.2.

Besides *human-centered feature selection*, to filter out deviation noise, RA introduced *behavioral data alignment*. Similar to *human-centered feature selection*, *behavioral data alignment* is also a human-centered approach. As shown in Fig. 5.7, the system sends every user a behavioral data list that describes their routine behaviors. Each entry in the behavioral data list describes the main component of the user's behavior derived by k-means clustering algorithm. Among these components, some of them represent the user's routine behaviors, the others might be deviation noise. By choosing their routine behaviors, users can help the system identify the behavior deviation; and hence, the authentication accuracy is increased. The detail of *behavioral data alignment* is discussed in Section 5.5.3.

In reinforcement authentication, the communication between the user and system is achieved using two lists, feature list and behavioral data list as shown in Fig. 5.8. In addition,



(a)


Behavioral Data List – ID 0001
Light Strength: weak, location A, Walking, ...
Light Strength: weak, location B, Sitting, ...
....
....
Light Strength: strong, location X, Sitting, ...
Confirm

(b)

Figure 5.8: Feature list and behavioral data list (user-side view). (a) Feature list. (b) Behavioral data list.

to achieve *human-centered feature selection* and *behavioral data alignment*, multiple entries in the list can be selected and uploaded to the server. Before finally adopted list we also tried SMS message, dialog box, and another tool to exchange information between the user and system, but their performance is limited. Compared to another tool, the advantage of using the lists is obvious. Firstly, it is easy to use and be understood. To send feedback to the system, users only need to perform a multiple-choice selection instead of writing a message. Furthermore, since each entry in the list has a fixed structure, the feedback can be easily processed by the system. From the users' perspective, they choose their routine behaviors and corresponding features by answering the query on the lists. On the other hand, from the system's perspective, the feedback will further guide it to pinpoint a personal feature set and corresponding behavioral data for a specific user. To make sure legitimate users understand the query conveyed by the lists, in real usage, a short menu that describes the importance of their feedback and the meaning of each feature (including its value) is provided. In the real test, with some descriptions, users can easily understand the lists including the ones who do not familiar with reinforcement authentication.

We assume the first user is legitimate. Since *human-centered feature selection* and *behavioral data alignment* take place at the beginning of the usage, the system only uses legitimate users' data to derive the personal feature set and filter out deviation noise. However, the behavior deviation may also happen during the usage, where the users' behavior may need to be adjusted. To this end, reinforcement authentication utilizes a hidden PIN number, which is different from the password used to unlock the device. Once the PIN is input, *behavioral data alignment* will be launched; and the system will adjust the setting based on the current users' behavior. It is possible the attacker can guess the PIN, but it is beyond the scope of this work. Interested readers could find [58, 14, 73] for more details.

5.5.2 Human-Centered Feature Selection

To pinpoint a personal feature set, reinforcement authentication utilizes *human-centered feature selection* and feature list. As a vessel, the feature list is used to convey information between the user and the system. From the users' perspective, they only need to select features that closely relate to their routine behavior. In contrast, the system needs to perform a series of operations to adjust the feature set based on users' feedback. From the system perspective, the feature list is shown in Table 5.1. Initially, reinforcement authentication attaches each available feature in the device with a unique feature ID. Utilizing users' historical data it then performers the first fold feature selection and clusters the features to two categories, the system reserved feature and the ambiguous feature. The ambiguous feature refers to the features that have similar authentication accuracy, e.g., $\pm 0.1\%$ accuracy difference. As discussed in Section 5.5.1, after *human-centered feature selection*, some of the ambiguous features that best represent users' routine behavior will be selected. The final feature set is a union of reserved features and user selected features.

In implicit authentication, since the system only performs the first fold feature selection, the final feature set contains all the features including ambiguous ones, which significantly decrease the system's performance. To reduce the number of ambiguous features, one way is to analyze each user's behavioral data and derives a personal feature set. This approach, however, is not feasible in practice, given GB-size or even TB-size users' behavioral data. Another approach is to randomly choose features from a group of ambiguous features which have a similar authentication accuracy. Likewise, this approach is not feasible because it decreases the authentication accuracy. For instance, suppose touch feature and location

Table 5.1: Feature List (System-side View)

Feature ID	Feature Name	Selected	System Reserved*
01	Touch Strength	N/A	Yes
02	Touch Area	N/A	Yes
03	GPS	Yes	No
...	...	...	...
N	Light Strength	Yes	No

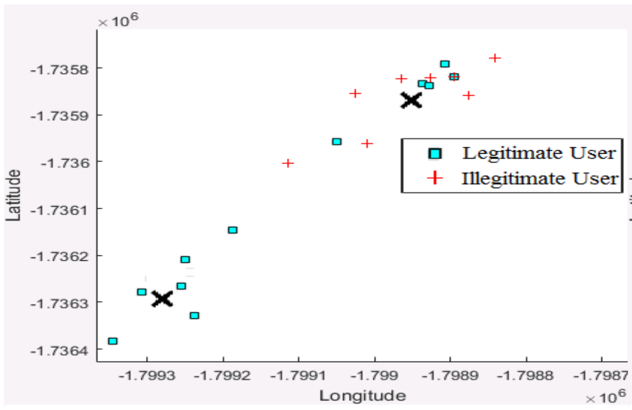
*The final feature set is a union of system reserved features and user selected features.

feature have similar authentication accuracy. If the system chooses the location feature instead of the touch feature, the authentication accuracy of users who have special touch behaviors, e.g., left-handed user, will decrease. Similarly, if the system chooses the touch feature instead of the location feature, the authentication accuracy of users who have a unique walking trajectory will decrease.

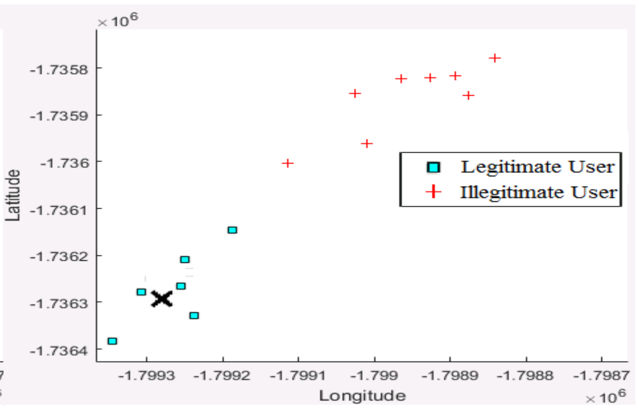
Leveraging *human-centered feature selection*, reinforcement authentication can pinpoint a personal feature set for each user with a small amount of energy consumption. As mentioned before, reinforcement authentication works in a duplex way. In *human-centered feature selection*, legitimate users' feedbacks enhance the feature selection process and improve the system's performance. Moreover, actively choosing their personal feature set, users play an important role in the feature selection phase. Subconsciously, due to operant conditioning [91, 90, 68], they tend to repeat the chosen behavior to pass the authentication. For example, left-handed users will tend to use their left hand more often after *human-centered feature selection*. The details of user-side enhancement and operant conditioning are discussed in Section 5.5.4.

5.5.3 Behavioral Data Alignment

This section describes related techniques used in reinforcement authentication, including *behavioral data alignment* and behavioral data list.



(a)



(b)

Figure 5.9: Impact of deviation noise. (b) Users' data that contains noise. (c) Users' data after filtered out noise.

Noise Filtering

For demonstration purpose, we adopt two dimensional GPS data in dataset [5] to show the impact of deviation noise in implicit authentication. Another sensors' data, e.g., light meter, accelerometer, and touch sensor, are similar. Each data sample is drawn in Fig. 5.9 (a). Note that multiple testing results may overlap together in one point in the figure. We use k-means clustering algorithm to find the main components of the data samples, which are marked using "**X**" sign. In this example, there are two main components among all the data samples, located on the bottom left and top right. We analyzed the data and found out the data samples closed to the top right **X** belong to the location that the user accidentally visited, which should be filtered out as noise. Some illegitimate users' data samples are also closed to this location. If the system fails to filter out the noise data samples, it cannot separate the illegitimate and legitimate users. Deviation noise is shown in Fig. 5.9 (a) significantly decreases the authentication accuracy.

Since the user actually traveled to that place marked by top right **X**, it contains sufficient data samples to become one of the main components. The existing noise filtering approaches cannot simply filter out the noise around the top right **X** without knowing the fact that the user only visited that place by accident. In fact, in implicit authentication, without additional calculation, the system will not know they are noise data due to the transparency.

Table 5.2: Behavioral Data List (System-side View)

B. ID.*	Feature 1	Value	Feature 2	Value	...
01	light strength	90-100	accelerometer	[x,y,z]	...
02	light strength	0-20	accelerometer	[x,y,z]	...
...	...	...	...	...	...
06	light strength	1500-2000	accelerometer	[x,y,z]	...

*The behavior ID (B.ID.) uniquely identifies each behavior.

Behavioral Data List

Table 5.2 shows behavioral data list from the system perspective. Each row in the behavioral data list refers to the behavior of the user. In behavioral data list, different behaviors that are uniquely identified by their ID were derived using k-means clustering algorithm. A behavioral data list is composed of feature names derived from a personal feature set and their corresponding values. From the users' perspective, the behavioral data list is shown as a query contained multiple choices. For example, given a behavioral data list shown in Table 5.2, the system will generate a six-entry query on the user-side application.

In addition, some of the feature names and their corresponding values are translated into human-readable text before sending to users. For example, accelerometer and its corresponding values are translated to gait patterns, e.g., sitting, walking, and running. The values of light strength are mapped to different levels, e.g., weak, moderate, and strong. The translate/map process follows the descriptions of [1, 2]. The interested reader could refer to the literature for more details.

Behavioral Data Alignment

In *behavioral data alignment*, as shown in Fig. 5.10, for every feature in the final feature set, the system applies k-means clustering algorithm to its corresponding data recently sampled from current users. Utilizing the main components returned from k-means clustering algorithm, the system constructs a behavioral data list, in which each entry in the behavioral data list corresponds to the main component returned by k-means clustering algorithm. The training server then sends the behavioral data list to users, who will select some of the behavioral data that best match their routine behaviors. At the end of the selection, the smart device will return the selected behavioral data to the training server. At this moment, the system already completed one round *behavioral data alignment*. While the system executing *behavioral data alignment*, it continuously fills databases with the new behavioral data sampled recently. Once more, the system will apply k-means clustering algorithm to the new-sampled data, but at this time it will also compare the difference of the main components between two rounds *behavioral data alignment* (previous round and current round). If there is no difference between them, *behavioral data alignment* is completed. If the current round *behavioral data alignment* contains new components that do not exist in the previous rounds, the system will launch another round *behavioral data alignment* until there is no more new component left in the data. *Behavioral data alignment* aims to achieve an agreement between the system and users. In the real test, *behavioral data alignment* often repeats three rounds on an average.

Since *behavioral data alignment* only takes place at the beginning of the usage, only legitimate users' data will be sent to the training server. Illegitimate users' data will not be used to generate behavioral data list. It prevents illegitimate users from taking advantage of the system by injecting their behavioral data to the dataset.

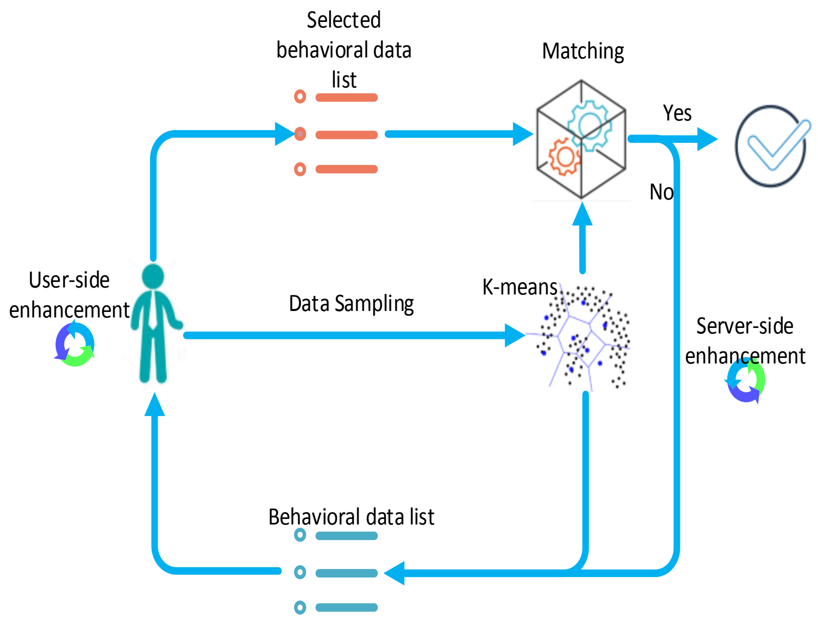


Figure 5.10: Behavioral data alignment.

In practice, to better achieve *behavioral data alignment*, a threshold parameter, σ, is used to filter out insignificant data samples. In reinforcement authentication, the system adopts wind vane algorithm [104] to dynamically adjust sampling frequency, in which a data sample contains all available sensors' value in a specific time slide, e.g., light strength, touch and GPS sensors at 4:04pm Oct 7 2018. Moreover, one sampling cycle may contain many data samples, in each of which the system records the data sample's value. Among these values, some of them may appear many times, e.g., light strength 97 may appear many times in one sampling cycle. Leveraging a hash map, the system stores sensors' values and the number of times they appeared in the current sampling cycle. To construct the new behavioral data list in the current round *behavioral data alignment*, only the data sample that appears enough number of times, e.g., larger than the threshold σ, is used.

In a sampling cycle p, given a specific data sample with the value of x, n denotes the number of time it outputs x. Among all sampling cycles P, the total number of data samples used to construct behavioral data list is:

$$N^{(x)} = \sum_{p=1}^{P} n_p^{(x)}, \tag{5.6}$$

where $n_p^{(x)} > \sigma$. Given $\mathbf{x}'$ contained only the data samples that satisfy the condition $n_p^{(x)} > \sigma$, the cost function in k-means clustering algorithm can be represented as:

$$\underset{S}{argmin} \sum_{i=1}^{k} \sum_{\mathbf{x}' \in S_i} \sum_{N} \parallel \mathbf{x}'_N - \mu_i \parallel^2, \tag{5.7}$$

where μ_i is the mean of points in S_i. In the real test, we also use the transformed cost function that minimizes the pairwise squared deviations of data samples in the same cluster to derive behavioral data list:

$$\underset{S}{argmin} \sum_{i=1}^{k} \frac{1}{2|S_i|} \sum_{\mathbf{x},\mathbf{y} \in S_i} \sum_{N} \parallel \mathbf{x_N} - \mathbf{y_N} \parallel^2 . \tag{5.8}$$

5.5.4 A Duplex Authentication

Reinforcement authentication is a duplex authentication, where the authentication process is enhanced by both the user and system. *Human-centered feature selection* and *behavioral data alignment* improve the authentication system in the aspects of energy consumption, time consumption, and authentication accuracy; and hence, from the system point of view, the authentication process is enhanced. Likewise, the authentication process is also implicitly enhanced by users.

The user-side enhancement leverage operant conditioning associated principles in psychology, named reinforcement and punishment [91, 90, 68]. One of the most important prerequisites of operant conditioning is both the chosen behaviors and corresponding stimulus are known by organisms (users), where the stimulus in this work refer to pass the authentication (reinforcement) or be blocked (punishment). For legitimate users, since they are main entities who perform *human-centered feature selection* and *behavioral data alignment*, the chosen behaviors is known, which are their routine behaviors. The stimulus is also known by users. Therefore, the chosen behaviors and their corresponding stimulus establish operant conditioning. Passing the authentication by repeating the chosen behaviors will enhance those behaviors. Finally, legitimate users will tend to repeat the chosen behaviors, while another behavior that had not been selected will appear less frequently. Therefore, the true accept ratio increases.

For illegitimate users, since they did not perform *human-centered feature selection* and *behavioral data alignment*, operant conditioning cannot be established. Furthermore, compared to implicit authentication schemes that focus on a specific feature set, e.g., gait, touch, and location, reinforcement authentication adopts all the available features, which reduce the chance of illegitimate users known the chosen behaviors. Due to the aforementioned reasons, the false accept ratio decreases.

In comparison, due to the transparency, neither the features used by the system nor the corresponding behavior data are known by legitimate users in implicit authentication. Hence, operant conditioning cannot be established. Because of its ignorance of users' feedbacks, and the authentication accuracy of implicit authentication is constrained.

Chapter 6

Evaluations

This section aims to present the performance of the PLDA topic model and various modules which have already been discussed. We will evaluate our methods from the following aspects: time consumption, stride size selection, and sampling rate.

We adopt the MIT Friends and Family Dataset [5] for our testing database, which contains 130 participants and has a total of 9 features (GPS, accelerometer, SMS, app installation, battery usage, call logs, app usage, blue-tooth devices log, Wi-Fi access points) recorded over five months. It is a complete dataset about human behavior based on sensor data. In 130 participants, we randomly select 23 people, labeled from User 01 to User 23, as our main dataset to evaluate our methods.

The original dataset was separated into eight different CSV files (not including survey forms), and most of these files have been attached with a timestamp. The users' behavioral dataset is derived by combining all of these eight CSV files according to the time stamp. Since some of the data stored in the dataset has less relation to our evaluation purpose, we have deleted these data and left the others unchanged. For example, the app installation data contains all the apps installed on the smartphone. These data will have fewer changes when an adversary steals the device.

The feature selection, achieved by tuning various parameters in the topic model, is integrated in the training phase in PLDA, in which the most related features are represented by the highest probability tokens in the topic. Due to the fact that users with heterogeneous data generally tend to generate different topic-token distributions, the main features that identify a specific legitimate user are usually unique and highly related to a user's ABP.

We prepare two datasets to imitate the normal usage (by the legitimate user) and abnormal usage (i.e., device captured by the adversary). The first dataset contains only data sampled from the legitimate user and is called the legitimate dataset. The second dataset is created by injecting data randomly sampled from other users into the legitimate dataset. It is called the synthetic dataset. We use the *splicing* approach [87] to inject data: we randomly select a user to be the legitimate user, and another user to be the adversary. We splice a portion of the legitimate user's segment with a portion of the adversary's segment. The point at which the two segments are concatenated is called the *splicing moment*. In practice, the *splicing moment* can be regarded as the time of device capture. The adversary's data portion is kept in the range of 25% to 50% of the new synthetic data.

6.1 Server-Side Evaluation

Now we consider the performance of PLDA in the synthetic dataset. First, we should choose a reasonable number of topics, which can be decided by calculating the perplexity[1]using 10-fold cross-validation. We used 155 MB of sensor data from 23 people over five-month periods from the Friends and Family Dataset with the following types of data: GPS, accelerometer, SMS messages, call logs, Bluetooth device logs, app installation data, app running data, and battery usage information. In the dataset used for simulation, pseudo-identifiers are applied to call log, SMS log and browser scans. All human-readable text, like phone numbers and text messages are captured as hashed identifiers, and never saved in clear text [5]. We plot the perplexity of the model and set the maximum number of topics to be 500. The lowest point occurs at around 300 topics, indicating that we need at least 300 topics to properly describe all 23 people during five months of usage as shown in Fig. 6.1 (a). Note that after reaching 300 topics, the perplexity slightly ramps up, which is due to overfitting and confirms that our choice of 300 topics is optimal. The hyper parameters, α and β, in the Dirichlet distribution, control the shape of the distribution. They are set to 0.01 to prefer the distinctive behaviors in each person, as more weight will be assigned to key behaviors when the hyper parameters are small. The number of iterations indicates the number of rounds of Gibbs sampling, which is the main technique used in PLDA to approximate topic and token distributions. This result also shows a significant deviation of human behaviors over five months, where most behaviors change with time and there are a large amount of behaviors that are unique for a specific individual.

For each person, the topic with the highest probability that occurs only with them is selected to represent their key behavior. We tested our system using the all-against-all method in which we compared each person against all other 22 people using the synthetic dataset. We evaluated the precision rate [2]$(\frac{TP}{TP+FP})$, which on average achieves 93.3% in the synthetic dataset. We also calculated the accuracy rate $(\frac{TN+TP}{TP+FP+TN+FN})$, which on average achieves 98.6% using the same dataset. Accuracy and precision for each person are shown in Fig. 6.1 (b) in detail. The state-of-the-art IA techniques have accuracies between 90% and 93% [62, 60]. In our previous work [106], we also compared the accuracy of different machine learning techniques widely used in the IA research, and showed that PLDA has the highest accuracy in differentiating users.

6.1.1 Retraining

Since the data set is very large (more than 17GB) with various of different data, we further divide and sort the data based on the type of features, and the final dataset contains 31 features plus 1 index with total 132960052 records.

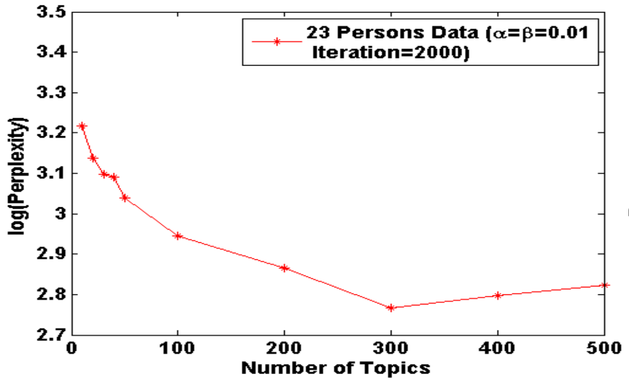
We run several different kinds of machine learning methods using k fold cross-validation method. For each different types of machine learning algorithms, we record its accuracy[3]and the corresponding timeline. In this experiment, we only train **once** at the beginning and we want to show the accuracy curve from one day to one-month time range. Since the data

[1]The perplexity is defined as

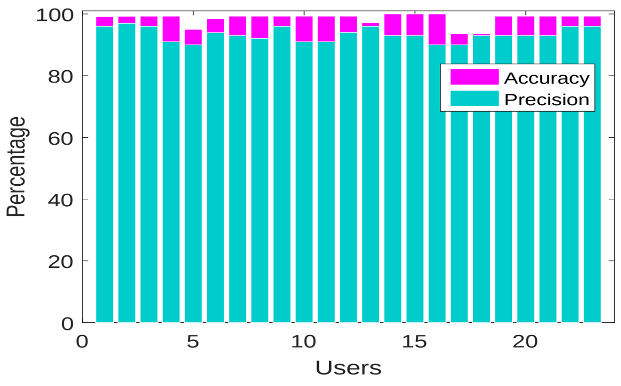
$$perplexity = exp\{-\frac{\sum_{d=1}^{M} logp(w_d)}{\sum_{d=1}^{M} N_d}\} \tag{6.1}$$

where w indicates the token for person d, and N is the total number of token, the numerator $\sum_{d=1}^{M} logp(w_d)$ is called log likelihood.

[2]where TP (true positive) indicates the system correctly passing legitimate users, TN (true negative) indicates the system correctly blocking illegitimate users, FP (false positive) indicates the system incorrectly passing illegitimate users, and FN (false negative) indicates the system incorrectly blocking legitimate users.



(a)



(b)

Figure 6.1: Initialization of PLDA topic model on server side. (a) Perplexity, (b) Accuracy and precision in synthetic dataset.

contains 5 months of information, we take the average of these 5 months. For the fine grain data like days and weeks, we also take the average value.

Fig. 6.2 shows the accuracy curves for PLDA, LDA, SVM with linear kernel and SVM with RBF kernel in the one-month time range. Since LDA uses unsupervised learning method, the accuracy is much lower than the others. From Fig. 6.2, we can clearly see that the accuracy drops significantly in between 5D(ays) to 6D(ays), 6D to 1W(eek), 1W to 2W, 2W to 1M(onth) due to the behavioral change[4]. Thus, even the adversary knows the user's historical data and uses it to imitate the legitimate user, the IA mechanism will still lock the device because of the behavioral change. However, the behavioral changes may also lead a locking of the device for a legitimate user. To prevent such unfriendly locking, we need to retrain the model. The following subsection will present details of retraining, which can be done automatically.

[3]More formally, we can define the accuracy as

$$\frac{TP+TN}{TP+FP+TN+FN} \tag{6.2}$$

where TP indicates the number of true positive authentications, FP indicates the number of false positive authentications, TN indicates the number of true negative authentications and FN indicates the number of false negative authentications

[4]There are slight differences in between PLDA and the other two SVM methods, but generally speaking, they all follow the same trend.

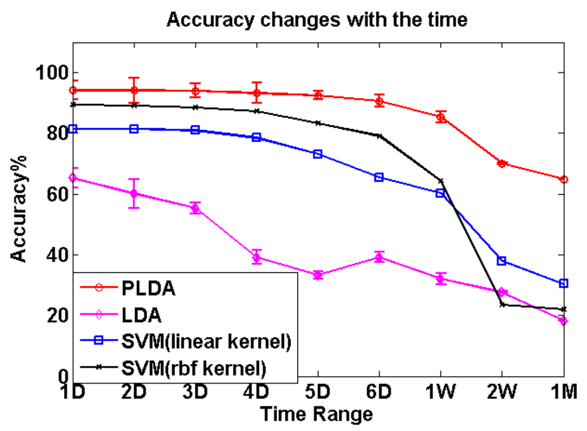


Figure 6.2: Accuracy for different machine learning methods

6.1.2 JS-Distance in Long-Term Testing

In this experiment, we calculate the JS-dis in between each day and all the days including itself for each person through the monthly time range. For example, we calculate the first day with the first day, the first day with the second day and first day to n day for each person. We take the average value of all persons. Then we draw the comparison result in the first row of the heat map. Similarly, the n row represents the comparison result in between n day's data and all the other day's data including itself. The average comparison time is trivial - it only takes less than 0.1 seconds to calculate the average JS-dis in between 1 day's samples for each person.

Fig. 6.3 shows the heat map for the average value of 1 day through the whole month using PLDA. The distance has been marked by different colors. Red color indicates long JS-dis with the maximum of 0.5. Deep blue indicates short JS-dis with minimum 0. The 0 value is resulted by compared with themselves.

From Fig. 6.3 we can clearly see that there are more red color slots in between the 5D to 1M than the former days. Moreover, in each time the accuracy dropped (Fig. 6.2), the JS-dis also fluctuates (Fig. 6.3) - this experiment shows the relationship in between the accuracy of machine learning model and the JS-dis for user behavioral dataset. By observing this fluctuation of JS-dis we can indirectly decide when to retrain the model.

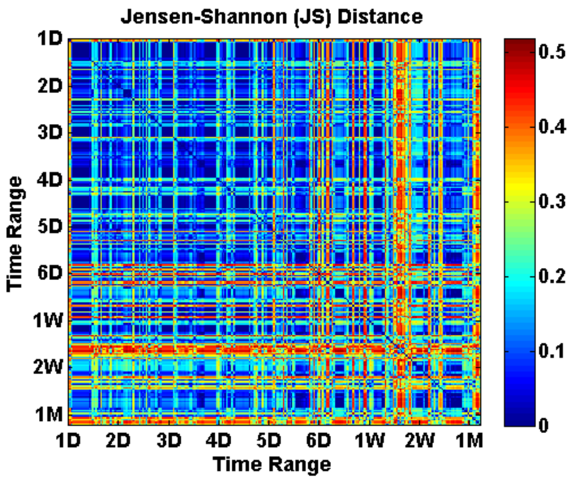


Figure 6.3: JS-dis heat map: red color indicates long JS-dis and deep blue indicates short JS-dis

6.1.3 Retraining Frequency

By taking the standard deviation of each stride k in the first row in Fig. 6.3, we can clearly see the fluctuation of the JS-dis. All the strides reflect the accuracy change rate except the first stride (containing 0 distance).

From Fig. 6.3 we can clearly see that there exist fluctuations in between day5 to week2. These fluctuations correspond to the accuracy changes in Fig. 6.2. The largest value of the difference in between any two standard deviations is the value within the range of 6D to 2W with the value of 0.024, and we can set the legitimate threshold $\varepsilon = 0.02$ for this dataset.

6.1.4 Levels for Dynamic Privilege

We utilize empirical data, which is the average JS-dis in between TPs and FNs in the dataset, to initialize the privilege levels. In practice, we do not know the correctness (e.g. FN, FP, TN, TP) of IA unless the user sends feedback to the training server, but we know its correctness using empirical data. We assume the empirical data reflect the key attributes of the legitimate user. Actually, the retraining process can reinforce the effectiveness of empirical data, and we can readjust the privilege level in each retraining.

In this experiment, we compare the JS-distances using the data of the same person and take the average of these values. We select three persons from the dataset to demonstrate

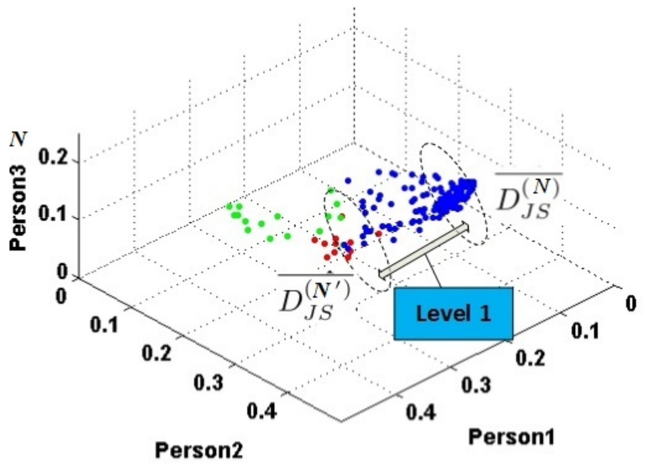


Figure 6.4: Level in dynamic privilege

our method and the result is presented in Fig. 6.4. First, we pick up one person from the dataset, marked as Person1. Second, we find out first two persons who have been mistakenly marked - FN (Actually, "they" are the same person - Person1). Fig. 6.4 shows the result of this test. The blue points indicate the TP for identifying Person1 as the legitimate user. From Fig. 6.4, we can see these points are closer to the Person1. The red points indicate that the FN of mistakenly identifying the Person1 to be Person2. Similarly, these points are closer to the Person2. The green points indicate the FP of mistakenly identifying the Person1 to be Person3.

The level can be defined as the following: First, we find the average JS-dis in between each TP sample and the other TP samples for all persons in the whole dataset, called $\overline{D_{JS}^{(N)}}$. In Fig. 6.4, it is the average JS-dis in between each blue point and other blue points. After that, we find out all the FN samples for all persons and average the JS-distances in between these FN samples and all the previous TP samples. In Fig. 6.4, the first FN sample for Person1 is the first time that red/green point occurs, and we average the JS-distances in between this point and all the other blue points to derive an average JS-dis. Similarly, we can calculate all the average JS-distances between m FN samples and all the previous TP samples, and we sort these m JS-distances based on their value. Then, we divide these values into different clusters, and further define the first level to be the average value of first clusters, marked as $\overline{D_{JS}^{(N')}}$. Utilizing the same method, we can define the second level $\overline{D_{JS}^{(N'')}}$, where $\overline{D_{JS}^{(N'')}}$ is corresponding to the average JS-dis of the second cluster, in which we average JS-distances between samples.

FN test

Using the same dataset, we will show the performance of four levels dynamic privilege regarding the FN rate. First, we find out (for each individual) the average JS-dis $\overline{D_{JS}^{(N)}}$ for each FN test. Then we use the same technique discuss previously to define each level rule. The results are shown in Table 6.1.

Using $\overline{D_{JS}^{(K)}}$ in Table 6.1 as the distance rule for our testing dataset, we rerun the former experiment which uses data from the same person. The detail of this experiment is described in the following: in the initialization, after we find all the FN samples in the historical dataset, we further calculate their JS-dis and sort them by their value (the detail is described in the previous subsection). When we finish the sorting procedure, we cluster these values and average each cluster to derive level rule as shown in the second row $(\overline{D_{JS}^{(K)}})$ in the Table 6.1. In the testing phase, each stride[5]has been input into the user model, in which we also calculates $\overline{D_{JS}}$ in between the current stride and previous TP samples. Suppose we are in L1 and the user model produces a $\overline{D_{JS}}$ larger than 0.342, the dynamic privilege control units will lower the current user's privilege to L2. In each testing, we record the number of FN and their corresponding privilege levels to see whether or not the dynamic privilege mechanism can map the user to a reasonable level. Because we use the same person's data, we expect the most of the testing results will be in the high level with minimum privilege limitation. Furthermore, we also expect the FN will have less impact of the current user.

In Table 6.1 row 3, the result shows that: 79.93% of the FNs make the system run at L1. In this level, there is no privilege limitation for users. 11.03% of the FNs make the system run at the L2 with bank and high privacy apps locked. 5.92% of the FNs make the system run at the L3 with contacts, social and some low privacy apps locked. In this level, one can perform a few basic operations on the phone such as call, SMS, time checking and so on. Only 3.12% of the FNs can lead a directly locking of the device. From the result, we can see that the dynamic privilege could largely reduce the unfriendly user experience by

[5]Please refer section 2.5.3 (Retraining Process).

Table 6.1: Levels and their performances

	L1	L2	L3	L4
$\overline{D_{JS}^{(K)}}$	0.342	0.387	0.467	N/A
FN	79.93%	11.03%	5.92%	3.12%
$Adversary$	1.31%	13.09%	11.04%	73.56%

*L1: full privilege. L2: lower privilege with bank, contacts, email list and high privacy apps locked. L3: lowest level privilege with all apps locked except some low privacy apps such as call, SMS, time and so on. L4: lock.

reducing the effect of FN. The reason is that the legitimate users may have some behavior changes but these changes are much smoother compared with the adversaries. The dynamic privilege will take the longer time to reach to the lowest level than the traditional method. If we bring in retraining method talked above, such lock will seldom happen.

Adversary test

We rerun the experiment using adversary setting. In this setting, we simulate the "stolen event" by injecting other users' data. For example, in Fig. 6.4, we use the data from Person2 and Person3 to rerun the test based on the training data of Person1. In this work, the testing JS-dis is the average distance in between all the Person1 training samples and Person2/Person3 testing samples. In this experiment (as shown in Table 6.1 row 4), there are 73.56% of the adversaries have been directly lowered to level 4, which means the user will be directly locked. 11.04% of the adversaries have been lowered to L3 with the minimum privilege. 13.09% of the adversaries can reach to L2. Only 1.31% of the adversaries can have the full control of the device with L1 privilege. Compared with the basic IA without dynamic privilege control, we improve the average precision rate from 79.75% to 98.69%. From this experiment, we can see that the dynamic privilege can improve the IA performance dramatically. The reason is that the behavior of adversary has larger difference compared with legitimate user's. From the other point of view, the average JS-dis $\overline{D_{JS}^{(N)}}$ is large in between the legitimate user and the adversary. As the result, the dynamic privilege will directly drop to the lowest level and lock the device.

Since we can define the rule of first privilege level to be a small number, this setting can prevent the adversary from accessing the device even if he can imitate the behavior of legitimate user with minor difference. Furthermore, in practice, since it is very hard to imitate the legitimate user in a long term, the IA mechanism will eventually lock the device once it finds the behavioral pattern mismatch.

6.2 Client-Side Evaluation

6.2.1 Time Consumption for Different Modules

Due to the importance of energy consumption and computation time for the smartphone, we evaluate them first.

We evaluate the performance of four different modules in Fig. 6.5 (a) using the legitimate dataset. We see the Dynamic Stride Module and WVM outperform the others. Since the dataset contains only one user, the performances for both the Dynamic Stride Module and WVM are the same in this measurement.

In Fig. 6.5 (a), the x-axis indicates the number of samples (cache size), and the y-axis shows the time consumption at the authentication stage. We calculated the time consumption using a Samsung Nexus S with 5000 to 15000 samples in the cache. It has some fluctuations in the curve of the Basic Module, but the Dynamic Stride Module and the WVM were faster than the others.

Since computing the JS divergence and choosing the best behavioral pattern of the current user is the one of the most energy-consuming parts of our system, we want to minimize the cost of these processes for each stride. In the experiment, we used the legitimate dataset to evaluate the stride size changing for each series of samplings. The results are shown in Fig. 6.5 (b).

In Fig. 6.5 (b), the x-axis shows each sampling point in the initialization step. The time gap between each sampling point is calculated by $T = \frac{t}{D_{JS}^{\gamma}}$, where $t = 10000ms$ and $\gamma = 1.5$. For each time point, the corresponding D_{JS} is shown in Table 6.2. The y-axis indicates the size of the current stride.

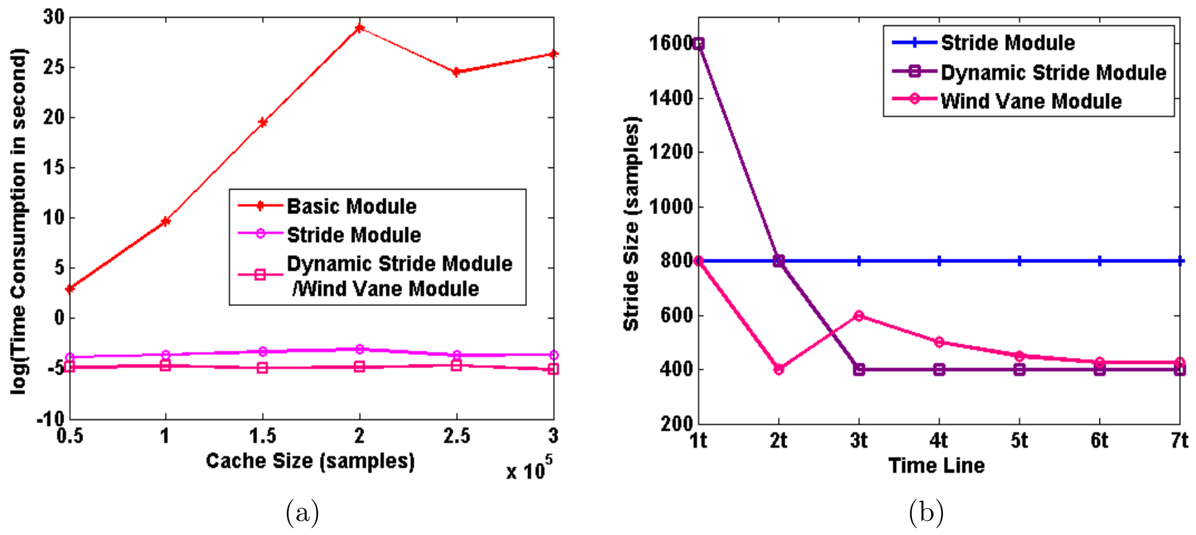


Figure 6.5: Time consumption and stride size in the initial stage. (a) Time consumption, (b) Stride size.

In the Stride Module, the stride never changes after initialization. The average stride size for the Dynamic Stride Module and WVM is less than 800 (about 400 when reaching stable status). As mentioned before, it is possible to initialize the fixed stride size, which is suitable for a specific dataset. However, in practice the behavioral patterns for different people are distinct. In Fig. 6.5 (b), we chose the best stride (425) at the beginning. It is only suitable for this dataset and may be inapplicable in practice. Thus, we design the size of each stride to be changed adaptively to match different behavioral patterns in the Dynamic Stride Module and the WVM.

6.2.2 Stride Size

To gain a better understanding of how D_{JS} affects the stride size and sampling rate in WVM, we record the time consumption for each sampling in Table 6.2.

From Table 6.2, at the initialization stage, we have an average JS divergence of 0.44, and the WVM assumes that the first user as being legitimate. However, the best stride size is unknown, and the WVM has two choices - enlarge or shrink the stride size. Since the current user is legitimate, the WVM will adjust the stride size to reduce JS divergence. Here, it chooses to shrink the stride n times, where $n = 2$ is predefined to simplify the problem

Table 6.2: JS divergence for each time slot and the corresponding calculation time

	1t	2t	3t	4t	5t	6t	7t
$\overline{D_{JS}}$	0.44	0.33	0.40	0.37	0.35	0.35	0.35
Time (s)*	0.03	0.02	0.02	0.02	0.02	0.02	0.02
Gap (s)	33.7	50.5	38.6	43.8	47.7	47.5	47.5

*The average time (in seconds) for calculating JS divergence in each sampling point beginning from 1t to 7t. "Gap" indicates the time gap (in seconds) between each sampling point.

to binary search, since comparing to the opposite operation "enlarging," the "shrinking" operation reduces $\overline{D_{JS}}$ to 0.33. Then it reaches to the next sampling point "3t" and chooses to enlarge the stride size because this will reduce D_{JS} comparing to the "shrinking" operation. This process repeats 6 times and converges at 7t, which takes a total of 309 seconds. We can reduce response time by decreasing the weight parameter γ, but this potentially increases the energy consumption. We will discuss the way of choosing γ in Section 6.5.

6.3 System Implementation

We have implemented an experimental system using the techniques presented in the previous sections. The prototype called *PersonaIA*, has four main components: background service, database, data upload/export control unit, and front UI (User Interface). The system is implemented using Android SDK with API level 7, and the WVM is developed and embedded as a background service. *PersonaIA* runs on a Samsung Nexus S smartphone, which has a total of 383 MB memory and 1000MHz of CPU bandwidth. It periodically samples the following sensors' data: accelerometer, location, light meter, compass and touch. The system UI is shown in Fig. 6.6 (a) and (b).

First, a background service periodically captures all sensors' data. Since all the apps in the Android system are running within their sandbox, the actions performed in one app can not be captured in another app. To avoid this technical obstacle, we implement a background service, which periodically captures the sensors' data and passes them back to *PersonaIA*. For efficiency purposes, the WVM is designed in the service, and for each sampling, it reads "wind strength" and compares it with the predefined threshold δ. Also, it records the "wind duration" m of a current user, which will further be used to decide behavior direction (legitimate or illegitimate). In each behavior change, *PersonaIA* will re-initialize the stride size to fit the behavioral pattern of the current user (finding the best behavior pattern for the user).

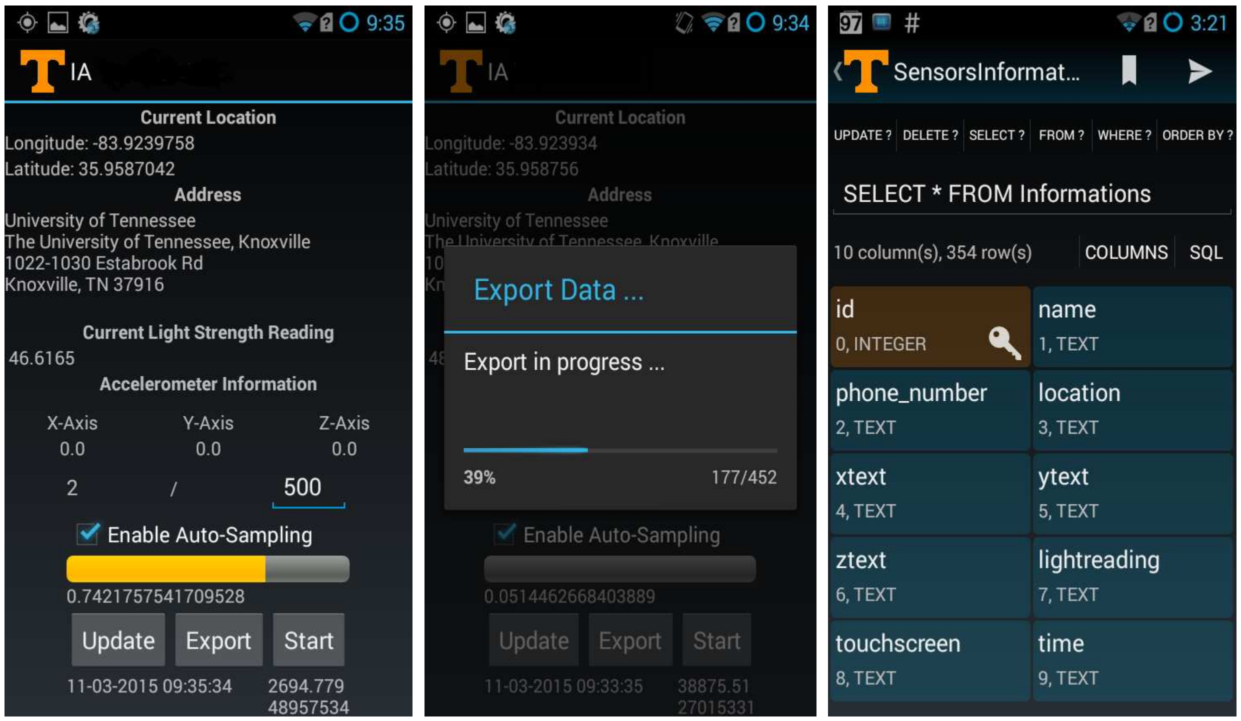
All sensor data is serialized and sent to the cache (shown in Fig. 6.6 (c)), which will de-serialize the received data and store the trimmed data in a SQLite database. Fig. 6.6 (c) shows parts of the data fields in the database using SQLite Debugger. The database utilizes user id, name and phone number to uniquely identify each device while keeping this information as keys for querying purposes. The cache size can vary to meet different requirements - one can set the cache size through system UI shown in Fig. 6.6 (a). The actual cache size setting depends on the memory size for various devices. For example, Samsung Nexus S can hold roughly 6000 samples, but when the number of samples reaches 6000, it may consume all the memory in the device and slow down the calculation speed of the system significantly. The system will automatically upload or export (depending on the implementation of the Function Interface) the samples once it reaches to a predefined size.

6.3.1 User Interface

Most of the sensor data is shown in the UI including fine/coarse-grain location, light meter, and accelerometer. Since constantly refreshing the UI may consume extra energy, we only update the JS divergence, current cache size, and sampling rate in each behavioral change (calculated by $\frac{t}{D^{\gamma}_{JS}}$). The user must click the update button to check other sensors' data.

On first usage, *PersonaIA* will ask the user to establish the *reference behavior pattern* by clicking the "Export" button shown in 6.6 (a). Before clicking the button, the user should wait for *PersonaIA* to collect enough data from their routine behaviors, which is critical for the future authentication. There is no default setting here, since for each user the routine behaviors vary, and is the same as the number of samples.

PersonaIA allows the researcher to select different cache sizes. For example, in the Fig. 6.6 (b), the current cache size is 500. If the total samples reach 500, *PersonaIA* will either upload the data to the server or export the data to local storage. Adversary testing score is shown in the yellow bar in Fig. 6.6 (a) with maximum setting of 1 and minimum setting


9:35

IA [redacted]

Current Location
Longitude: -83.9239758
Latitude: 35.9587042

Address
University of Tennessee
The University of Tennessee, Knoxville
1022-1030 Estabrook Rd
Knoxville, TN 37916

Current Light Strength Reading
46.6165

Accelerometer Information

X-Axis	Y-Axis	Z-Axis
0.0	0.0	0.0

2 / 500

✓ Enable Auto-Sampling

0.7421757541709528

Update Export Start

11-03-2015 09:35:34 2694.779 48957534

(a)

9:34

IA

Current Location
Longitude: -83.923934
Latitude: 35.958756

Address
University of Tennessee
The University of Tennessee, Knoxville

Export Data ...

Export in progress ...

39% 177/452

✓ Enable Auto-Sampling

0.0514462668403889

Update Export Start

11-03-2015 09:33:35 38875.51 27015331

(b)

3:21

SensorsInformat...

UPDATE ? DELETE ? SELECT ? FROM ? WHERE ? ORDER BY ?

SELECT * FROM Informations

10 column(s), 354 row(s) COLUMNS SQL

id 0, INTEGER	name 1, TEXT
phone_number 2, TEXT	location 3, TEXT
xtext 4, TEXT	ytext 5, TEXT
ztext 6, TEXT	lightreading 7, TEXT
touchscreen 8, TEXT	time 9, TEXT

(c)

Figure 6.6: Screenshots of *PersonaIA*. (a) Identification score and various parameters, (b) Data exporting using authentication interface and (c) Cache database overview using SQLite debugger.

of 0. In this setting, if the score goes below 0.5, the system will consider the current user as legitimate, because a lower score indicates that the current user is most likely legitimate. In Fig. 6.6 (a), the score is above 0.5 and the system will consider the current user as an adversary. Finally, after the stride size converges, the sampling rate (ms), which has been shown in the right corner of UI, will be stable when the sampling rate difference is between 0.5 - 1 per second.

The user can choose to either enable or disable WVM by checking the box shown in the Fig. 6.6 (a). The system will automatically upload/export the data to the remote/local storage if the box is checked. Otherwise, the exporting unit, which is shown in Fig. 6.6 (b), will not be triggered.

Table 6.3: CPU and memory consumption for different apps

	T.W.C.*	Amazon	F.B.*	Gmail	PIA*
CPU Avg%	16	46	21	24	12
CPU Max%	39	100	39	24	23
Mem.(MB)	68	120	136	26	29

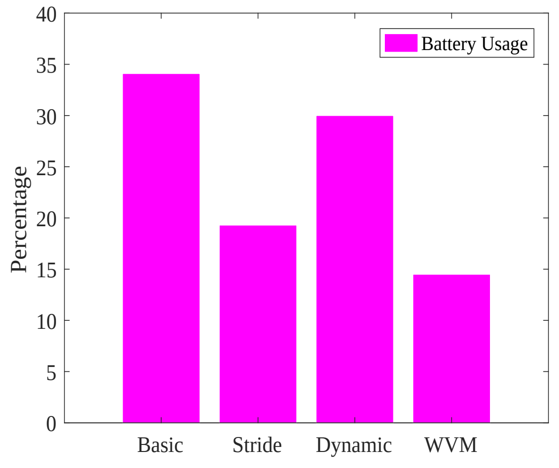
*The abbreviation T.W.C. is for The Weather Channel app, F.B. for FaceBook, PIA for *PersonaIA*.

6.3.2 Utilization Information

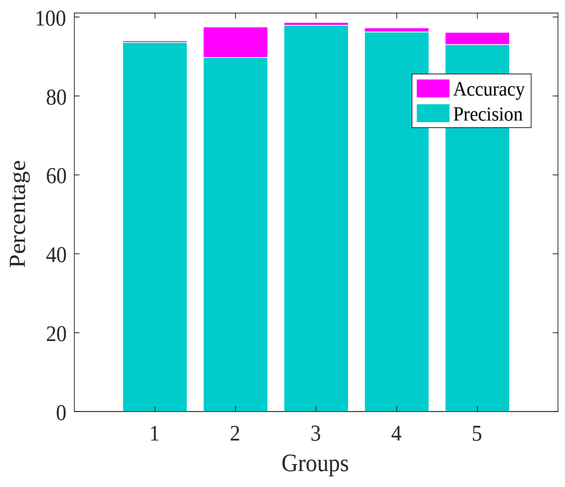
CPU Utilization: We measure the CPU utilization of our system on the Samsung Nexus S using the CPU monitor app, which allows one to monitor the CPU utilization of all the processes running on a smart device. We first start our service and it consumes 5% CPU utilization at the beginning and then reaches 10%. After nine minutes, the CPU utilization was stabilized between 11% - 13%. When the service is running the stride initialization stage, the CPU utilization will reach 22% - 23% at most, and then goes back to 11% - 13%. Since the initialization step only spans a few sampling cycles in average, most of the time the device is running in the low CPU consumption stage with CPU utilization between 11% - 13%. We compared the CPU usage of *PersonaIA* with other popular apps. The list contains The Weather Channel, Facebook, Gmail, and Amazon (as shown in the Table 6.3). We found that *PersonaIA* has the lowest CPU usage. We can conclude that *PersonaIA* is lightweight even on the relatively low-end Nexus S smartphone.

Memory and Storage: The memory consumption of our system is nearly constant. It consumes 28 MB($\pm$4 MB) memory of the total 383 MB on the device, if we select the total cache size of 1000 samples. We can further reduce the memory consumption by setting the cache to some small value, e.g., 500. The front app consumes about 4.08 MB storage in the device, but the USB storage consumption could vary from 1 MB to several GB depending on the frequency of data exporting. If we have a long time (one month) with no network connection, we should clean the external storage.

Battery Usage: In addition to the CPU and memory usage, we also conduct a test to compare the battery usage between four modules. For each module, we only consider the client-side matching algorithm and authentication mechanism with the training phase offloaded to the server side. We run each module independently by the same group of users each time, and capture the battery usage using GSam battery monitor app, which shows the battery usage in percentage of the total usage. To minimize the interference between modules, we run each module independently for 24 hours. The testing results are shown in Fig. 6.7 (a), in which "Basic" indicates the Basic Module, "Stride" indicates the Stride Module and "Dynamic" indicates the Dynamic Stride Module respectively. The current IA frameworks, e.g., [49, 53, 62], belong to either the Basic Module or the Stride Module. As shown in Fig. 6.7 (a), even if they have used JS divergence, the performance would be inferior to the proposed WVM. The battery usages of the Basic Module, Stride Module, Dynamic Stride Module and WVM are 34.1%, 19.3%, 30.0% and 14.5% of the devices' total battery usage. We conclude that the Basic Module and Dynamic Stride Module consume more energy than the Stride Module and WVM because they use all samples in the cache database to identify a user. The size of stride in Stride Module is selected to be the one that minimizes classification errors, but remains constant for different users, which may be infeasible in practice even if it consumes a small amount of energy as we discussed in Section 3.5.2. The WVM outperforms other modules due to its adaptiveness in controlling sampling rate and stride size.



(a)



(b)

Figure 6.7: Energy efficiency, accuracy and precision. (a) Battery usages in different modules, (b) Accuracy and precision in real usage.

Accuracy in Real Usage: To further study the performance of *PersonaIA*, we collected 21074 samples from 7 volunteers of different ages and genders during six-week-testing periods. The test is performed by randomly matching two volunteers in a group. There are 6 groups in total. One person in the group is the legitimate user while the other is the adversary. For the training phase, the legitimate users are asked to carry the device for more than one week and send their data to the server for training. In the testing phase, the WVM is enabled to authenticate users with an average wind strength $\delta = 0.5$. Users, both legitimate and adversary, are required to report the number of times *PersonaIA* fails to identify them out of the total number of times the system is used. In addition, to evaluate the sensitivity of *PersonaIA*, we calculated the accuracy and precision rate in the first five minutes of the volunteers' usage in the testing phase. Finally, we derive the overall accuracy and precision for each group, which is shown in Fig. 6.7 (b).

We observe in Fig. 6.7 (b) that for Group 1 through Group 5, the average precision (94.09%) and accuracy (96.70%) are similar to the tests using the synthetic dataset, indicating that the most probable topic among all topics can be used to separate users in these groups. However, *PersonaIA* failed to identify the illegitimate user in Group 6 where the accuracy and precision are less than 50%. The reason for this is that in Group 6, both users have similar behaviors: they live in the same apartment and work in the same place. In this case, to capture the difference between the two users, *PersonaIA* can only rely on touch and accelerometer data out of all the data types we used. After analyzing the data, we found that the columns for these two data types are almost empty in most samples, which means that the users had left the device somewhere without actually using it. Interestingly, *PersonaIA* achieves very high accuracy (97.83%) in differentiating between users from Group 6 and users from Groups 1 through 5. In practice, the data contained in Group 6 can be filtered out without losing authentication correctness.

We evaluated the accuracy of *PersonaIA* in the first five minutes of usage, using data from legitimate and illegitimate users. The average accuracy in identifying legitimate users is 90.39% considering all groups and 98.47% considering only Groups 1 through 5. Table 3 shows the average accuracies for every minute. As the WVM adjusts the stride size

Table 6.4: Accuracies In Five Minute Testing

	1min	2min	3min	4min	5min
Acc%	95.37	96.48	95.91	97.82	98.47
Acc%*	87.81	88.74	88.26	89.85	90.39

*The first row contains testing only from Group 1 to Group 5, and the second row contains all group's testing results.

and sampling rate in each time period, e.g., one to two minutes, the accuracy increases accordingly, but the rate at which it increases is slower than expected.

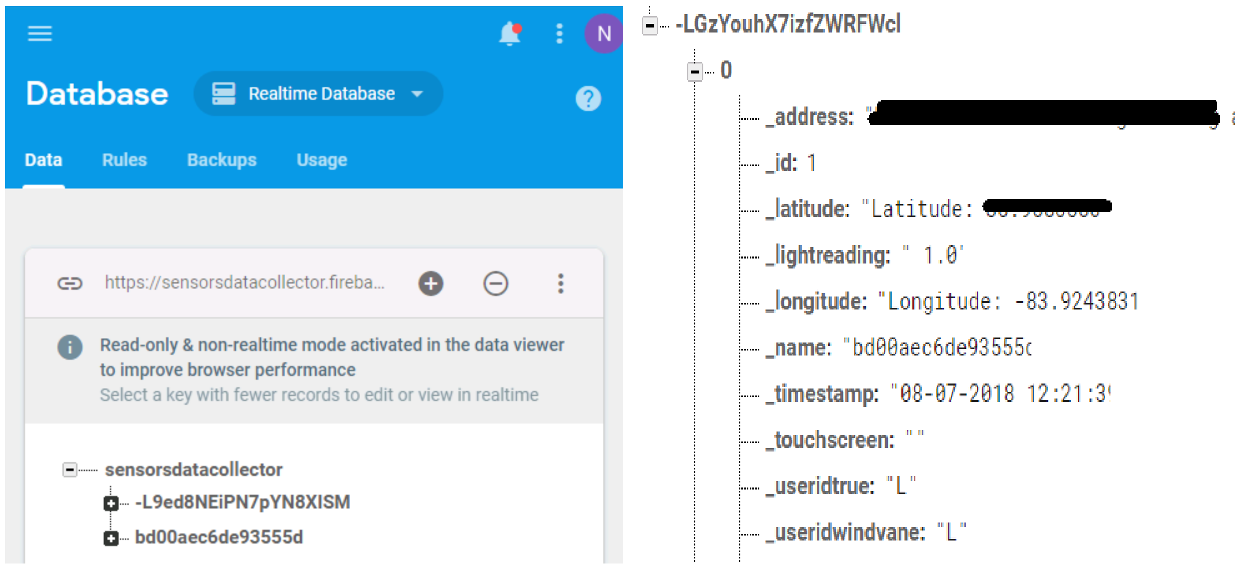
6.4 Another Measurement

6.4.1 User-Side Services

Running on the Android system, user-side services were written by Java using Google-SDK. Besides system-level services, a sampling service periodically wakes up sampling algorithm, which gathers all the available sensors' data and stores them in the cache database developed using SQLite. The sampling speed and wake-up time is dynamically adjusted for the energy-saving purpose. Device-server data transmitting service contains behavior matching and data transmitting units. The behavior matching unit is responsible for identifying users by comparing their behaviors using the model returned from the training server. The data can be further encrypted in the data transmitting units. In this work, however, we mainly focus on BMap; the data privacy preserving and associated techniques are beyond the scope of this study. Interested reader can refer [81, 12] for more details. Data transmitting unit is used to upload sensors' data stored in the cache database to database server through a secured channel. The user-side services were installed and tested on Motorola G2 with Andoird version of 6.0. It has 1GB memory, 8GM local storage, a Quad-core 1.2 GHz GPU, and Adreno 305 CPU. Running in the background, the use-side services are transparent to users.

6.4.2 Server-Side Implementation

The database server is implemented using Firebase which is a NoSQL database and can easily handle multiple users' data transmissions. The training server utilizes a Lenovo 16GM memory quad-core processor machine with 2.4GHz frequency in each core. The server has two GPUs, which are an integrated Intel HD Graphics 4000 GPU and a Nvidia GeForce GTX 660M GPU. All users' data are stored in JSON format as shown in Fig. 6.8. The data uploading procedure has several steps discussed as following. In the beginning, based on device ID, behavioral data is stored in different branches in the JSON tree with distinct label, e.g., L9ed8NEiPN7pYN8XISM, as shown in Fig. 6.8 (a). The device ID and label are one-to-one correspondence relationship. A monotonically increasing index is created to store each data sample as shown in Fig. 6.8 (b). The index and time stamp uniquely identify each data sample. For example, in 6.8 (b) index "0" and time stamp "08-07-2018 12:21:3" uniquely identify the data sample in the JSON tree. Behavioral data sampled from various sensors is stored in different branches, e.g., in the index "0" it stores _address, _id, _latitude, and another seven different types of behavioral data. To bridge database server and training server, we implemented a lightweight program using Javascript, which can achieve a fast data transmission.



https://sensorsdatacollector.fireba...

Read-only & non-realtime mode activated in the data viewer to improve browser performance
Select a key with fewer records to edit or view in realtime

```
sensorsdatacollector
  -L9ed8NEiPN7pYN8XISM
  bd00aec6de93555d
```

```
-LGzYouhX7izfZWRFWcl
  0
    _address: "[redacted]
    _id: 1
    _latitude: "Latitude: [redacted]
    _lightreading: " 1.0'
    _longitude: "Longitude: -83.9243831
    _name: "bd00aec6de93555c
    _timestamp: "08-07-2018 12:21:3
    _touchscreen: " "
    _useridtrue: "L"
    _useridwindvane: "L"
```

(a) (b)

Figure 6.8: Database server. (a) Users are stored in different branches of database. (b) Data is stored in JSON format.

6.4.3 Real Experiment

To evaluate the performance of BMap, we conducted a long-term real test since April 2016. Spanning two years and eight months, we tracked the usage of 13 different volunteers using the proposed system. In the experiment, each device only has one legitimate user, while another user is deemed as an illegitimate user. In addition, every illegitimate user is required to guess password and mimic legitimate users' behavior during the usage. To this end, the devices' passwords are randomly chosen with length of eight characters that contain both letters and numbers. For each device, one of 13 volunteers was using it in a period of time (longer than two weeks); and thus, each device stores data from 12 illegitimate users and one legitimate user.

In the previous experiment, due to the limitation of the dataset, we can only evaluate BMap on Shi scheme and Multi-Sensor scheme. However, in the long-term real test, we gathered rich usage information from all users. Besides the features used in the synthetic experiments, we also collected touch-related data, e.g., trajectory, pressing time, and corresponding accelerometer reading, which makes the evaluation of BMap on Gait scheme [35] and SilenSense scheme [19] possible. To this end, we implemented Gait scheme and SilentSense scheme in our system. Similar to the implementation of Shi scheme and Multi-Sensor scheme, we use the recommended settings of Gait scheme and SilentSense scheme from their original papers. In addition, we applied k-fold cross-validation to choose the best value of the parameters in each scheme, and to conduct training and testing. The feature selection strictly follows the description of the original papers.

Authentication Accuracy

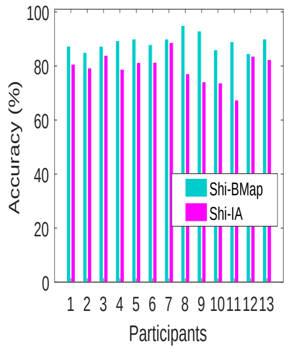
To evaluate the accuracies' improvement of BMap on different schemes, in the beginning, we measured the authentication accuracies of original schemes using testing dataset. Using the same setting, we then applied BMap to the schemes, and repeat the measurement on the same testing dataset. The results are shown in Fig. 6.9. As shown in the figure, for most of the users, BMap-based schemes have higher authentication accuracy than the original schemes. We calculated the average accuracy for original Shi scheme, Muti-Sensor scheme, Gait scheme, and SilentSense scheme, which are 79.39%, 84.71%, 74.46%, and 73.56% respectively. We also calculated the average accuracy for BMap based Shi scheme, Multi-Sensor scheme, Gait scheme, and SilentSense scheme, which are 88.79%, 94.68%, 81.61%, and 85.85% respectively. For all four schemes, BMap boosts their authentication accuracy significantly, especially for the Multi-Sensor scheme and SilentSense scheme. The authentication accuracy improvements of Shi scheme, Muti-Sensor scheme, Gait scheme, and SilentSense scheme are 9.4%, 9.97%, 7.15%, and 12.29% respectively. Although none of the illegitimate users successfully guessed the correct passwords, they can still mimic legitimate users' behavior and pass the authentication, which is one of the biggest problems in today's implicit authentication schemes [56]. Especially for the Gait scheme and SilentSense scheme, their corresponding mimicry attacks are very effective. However, as shown in Fig. 6.9, BMap can still reduce the success rate of the attacks and increases the authentication accuracies of the schemes.

In this test, for both original schemes and BMap-based schemes, we stored the number of times legitimate user been locked by the device within 7250 attempts. To compare, we recorded the number of times illegitimate user been locked by the device in 7250 attempts. The testing results are shown in Table 6.5. Comparing to the original scheme and BMap-based scheme, we can see the number of time legitimate user been locked is reduced; and hence, the usability of the system is increased. Similarly, the number of time illegitimate user been locked is increased. The security of the system is enhanced since illegitimate user will have larger chance been blocked.

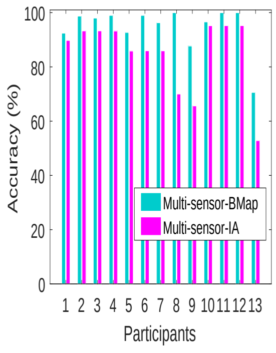
Table 6.5: The number of times users been locked

Scheme	Original scheme		BMap-based scheme	
	Legi.* locked	Ille.* locked	Legi.* locked	Ille.* locked
Shi scheme	213	4863	179	6243
MultiSensor	187	5640	53	7015
Gait scheme	379	4310	205	5186
SilentSense	179	3973	174	5800

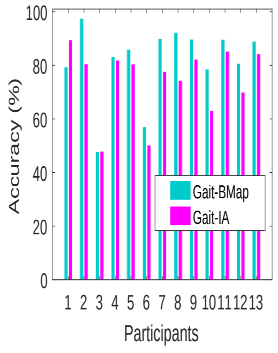
* Legi. denotes the legitimate user. Ille. denotes the illegitimate user.



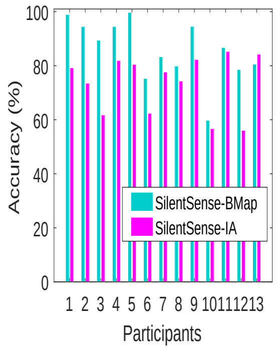
(a)



(b)



(c)



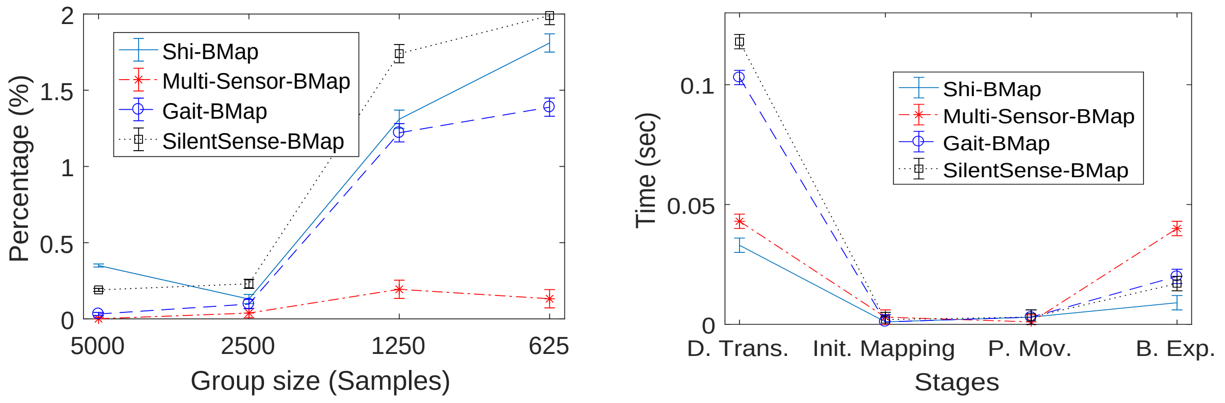
(d)

Figure 6.9: Accuracy improvement for both original schemes and BMap-based schemes. (a) Shi scheme. (b) Multi-Sensor scheme. (c) Gait scheme. (d) SilentSense scheme.

Time Consumption

We evaluated the time consumption of BMap, and compared it with time consumptions of training, data transmitting, data initialization, and data exporting in the original systems. Table 6.6 shows the time consumption for different schemes. The column denotes different stages of various schemes. As shown in the table, the time consumptions of BMap on different schemes are very small compared to another operation. The total time consumption of each scheme after applied BMap is shown in the final column of the table.

In addition, since implicit authentication utilizes a group of data exported recently to identify users, e.g., 5000 samples in each group, the size of the group impacts the time consumption of the system. Given different data exporting frequency, the time increment of BMap is shown in Fig. 6.10 (a). Furthermore, among different group sizes, we calculated the average time-consumption percentages of BMap in different schemes, which are 0.9%, 0.0912%, 0.657%, and 1.037% for Shi scheme, MultiSensor scheme, Gait scheme, and SilentSense scheme correspondingly. Specifically, the time consumptions of BMap in Data Transmission, *Initial Mapping*, *Privilege Movement*, and *Bubble Expansion* are shown in Fig. 6.10 (b).



(a) (b)

Figure 6.10: Time Consumption. (a) Time consumption percentage in different group sizes. (b) Time consumptions of different stages.

Energy Consumption

Besides the time consumption, we also conducted several experiments to measure the difference between original schemes and BMap-based schemes in the aspect of energy consumption. In the experiment, we measure the batter usage in the original schemes by calculating the average working hours of battery after fully charged. To compare, we also measured the batter usage in the BMap-based schemes. The details are shown in Fig. 6.12. As shown in Fig. 6.12 (a), the average working hours of original schemes and BMap-based schemes are almost the same. Specifically, we calculated the battery working hour reduction by applying BMap, which is less than 0.9% of the total working time. The average battery consumptions in Data Transmission, Initial Mapping, Privilege Movement, and Bubble Expansion are shown in Fig. 6.12 (b).

6.4.4 Other Performance Measurements

We calculated the percentage of behavior scores that were mapped to each privilege level in BMap, for the legitimate user and illegitimate users. As shown in Fig. 6.11 (a), less than 0.4% of the behavior scores are mapped to the observation levels, which indicates that BMap is fast and highly effective in making the final decision.

Table 6.6: Time consumption for different schemes (sec)

	BMap	Training	Trans.*	Init.*	Export	Total
Shi scheme	0.046	0.524	0.96	1.57	1.03	4.13
MultiSensor	0.051	40.63	4.69	2.137	5.16	52.66
Gait scheme	0.127	14.28	0.95	2.917	1.03	19.30
SilentSense	0.140	3.291	1.03	1.918	1.06	7.43

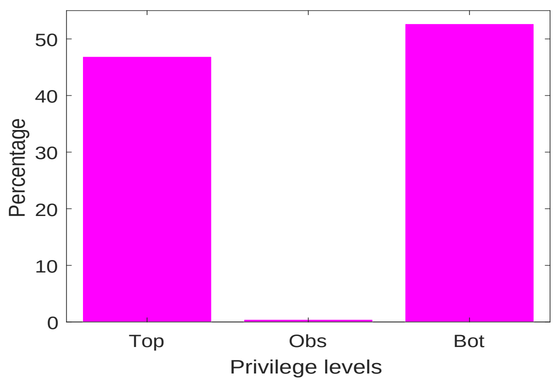
* Trans. denotes the total data transmission time consumption in the system except BMap. Init. denotes the time consumption of data initialization. In addition, the data initialization contains data formatting and noise filtering.

We also calculated the behavior score distributions in each privilege level for time windows 10 through 100 as shown in Fig. 6.11 (b). The z-axis denotes the number of behavior scores. The y-axis denotes the time windows. The x-axis denotes the privilege levels, where the left three levels that contain top, observation, and bottom levels, are plotted from the legitimate user's behavior scores; similarly, the right three levels are plotted from illegitimate users. For both users, the number of scores which fall in the observation level is small, less than 118 given total of 27138 scores, which is similar to the result in Fig. 6.11 (a).

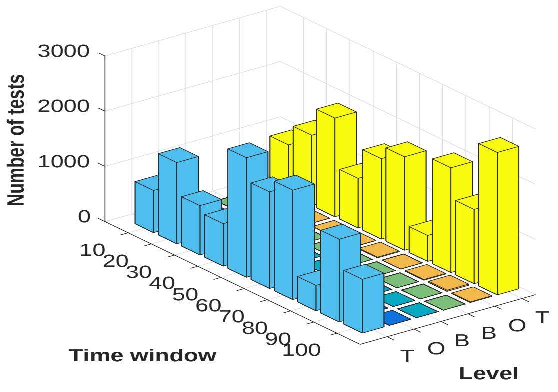
6.5 Discussion

This section aims to answer and discuss some important questions about *PersonaIA*.

Local min./max. values for legitimate/illegitimate user:
The rationale behind our design is that the most recent behaviors tend to have more interpreting power of the user than past behaviors. This is observed from our experimental results using the Friends and Family dataset, which showed that human behaviors deviate with time (i.e., topics found by our topic model are diverse). For the legitimate user, the WVM will find the most recent local minima, which is reflected in the JS divergence and the corresponding stride size. For the illegitimate user, the WVM will find the most recent local maxima. We agree that it would be interesting and useful to study the global minima/maxima for the legitimate/illegitimate user, since behaviors should be repetitive. This global minima/maxima can in fact be obtained from the topic model because the topic



(a)



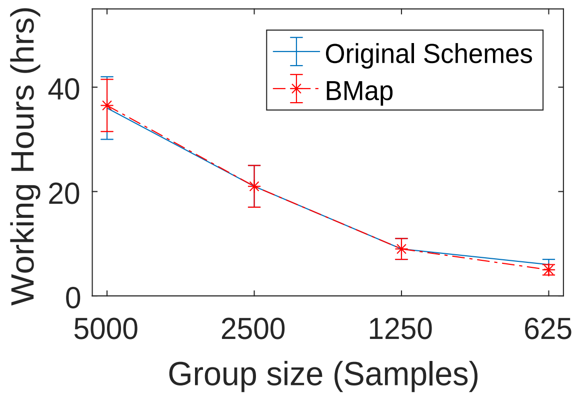
(b)

Figure 6.11: The proportion of behavior scores in each level. (a) Average proportion. (b) Proportion in time windows 10 through 100.

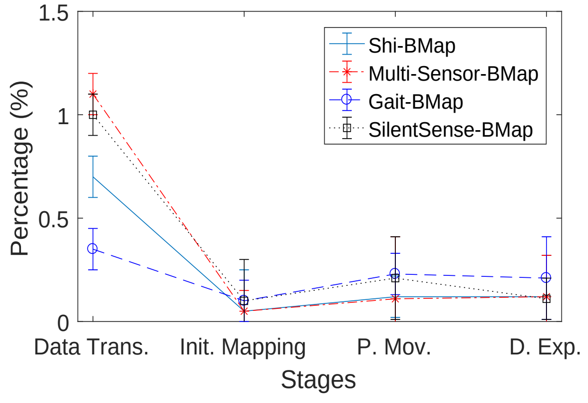
model takes the entire dataset (recent and past) to run. We plan to explore it in our future research given the page limit of this work.

How to choose the values of parameter γ, m and δ: Intuitively, γ represents the strength of JS divergence or the system's belief on the effectiveness of JS divergence. Together with JS divergence, it adaptively controls the time gaps between two samples by $\frac{t}{D_{JS}^{\gamma}}$. When γ is large, the system has a strong belief on JS divergence and the WVM. To avoid bias, we choose γ by averaging over the historical testing results, in each of which a mis-authentication should decrease γ until $\gamma = 0$ and a correct authentication will increase γ until γ reaches the maximum, e.g., 1.5, equivalent to about five-minute-sampling gaps when $D_{JS} = 0.1$.

The duration time m is similar to γ, which controls the response time of the WVM. Due to behavior deviation of the legitimate user, the WVM may obtain a large JS divergence but it should not change direction immediately until it collects more evidence about the illegitimacy of the user. Therefore, m cannot be too small. On the other hand, a large m reduces the sensitivity of the WVM in detecting illegitimate users. Using the training set, we can easily find the most reasonable m which minimizes the classification errors. We can find a suitable wind strength of δ in a similar way. Please note that we may still have the bias on the training set, even though we used k-fold cross-validation. Retraining may be necessary in this case, the detail of which can be found in our paper [106].



(a)



(b)

Figure 6.12: Battery Consumption. (a) Average battery consumptions for both original schemes and BMap-based schemes. (b) Average battery consumptions of different stages.

Data privacy and protection:

It is possible that the adversary compromises the database server and obtains sensitive behavioral data from the users. Several techniques, including hash/obfuscation, TrustZone [60] and Software Guard eXtensions (SGX) [45], have been developed against the attack. This problem, although important, is out of the scope of this work.

Chapter 7

Conclusions and Future Works

In this study, we investigated various problems in implicit authentication and proposed dynamic retraining (RU), wind vane module (WVM), BubbleMap (BMap), and reinforcement authentication (RA) to improve the efficacy of implicit authentication (IA).

Although the accuracy of IA can increase with more advanced technology used in smart devices, the retraining and authentication failure problems still hinder realistic deployment of IA systems. How and when to retrain the user behavior model and what to do when the legitimate user fails the authentication remained unsolved. In Chapter 2, to address the retraining problem, we proposed a technique using JS-distance to determine the best retraining frequency. For authentication failure, we introduced the dynamic privilege mechanism with finer privilege levels. Compared with the predefined privilege rule, we can decide which level should be assigned to the user based on his/her current behavior. Compared with the lock-only mechanism in the existing-related work, the dynamic privilege-based access control can largely reduce unpleasant user experience by only locking part of the device.

In Chapter 6, we tested our methods on a dataset of 130 persons with more than 5-months worth of records. The simulations showed that our retraining techniques can successfully detect the accuracy changes, suitable for use in IA system. The results also show that the dynamic privilege mechanism can largely reduce the effect of false-negative authentication failure.

In Chapter 3 and 4, we introduced the W-layer and the WVM to achieve lightweight IA, while supporting server-based IA with embedded topic models. The W-layer challenges the traditional concept of server-based IA by adding several components between the server and sensors to achieve energy and computational efficiency, lightweight authentication, sampling control, and high compatibility. In addition, we proposed the WVM to further reduce energy consumption and improve the correctness of authentication. The WVM can extract the best representative behavioral pattern of the current user, and compare it with the historical behavioral pattern of a legitimate user. Since the WVM is designed in the W-layer, it inherits all the properties of the W-layer. On the server side, we implemented PLDA for more accurate authentication, and it achieves 93.3% precision and 98.6% accuracy in the synthetic dataset.

In Chapter 5, we proposed BubbleMap (BMap) and reinforcement authentication (RA) to enhance the performance of various implicit authentication (IA) schemes. As a framework seamlessly lying above them, BMap can significantly boost the performance of the original schemes. In BMap, we modeled the privilege changing process of users and bridged the privilege control mechanism to implicit authentication. To this end, we introduced *Initial Mapping*, *Privilege Movement*, and *Bubble Expansion* techniques. In addition, we evaluated BMap in a large-scale simulation on state-of-the-art IA schemes. We also implemented BMap and performed a long-term test over two years and eight months. The test results show BMap can increase the performance of the original schemes with a small amount of energy consumption. Specifically, in the real experiment, the average authentication accuracies of Shi scheme, Multi-sensor scheme, Gait scheme, and SilentSense scheme are 79.39%, 84.71%, 74.46%, and 73.56% respectively; and the average authentication accuracies after applied BMap are 88.79%, 94.68%, 81.61%, and 85.85% respectively. The time consumption increased by BMap is less than or equal to 1% for all four of the schemes. Similarly, the battery consumption increased by BMap is less than 0.9% of the total working time. We also proposed reinforcement authentication that achieves human-centered authentication utilizing *human-centered feature selection* and *behavioral data alignment*. Compared to the machine-centered approaches used in implicit authentication, the proposed methods achieve better user authentication in the aspect of authentication accuracy, time utilization, and energy efficiency. In the future, we will share the system's source code, parameter setting, and dataset on our lab website to benefit associated research.

Bibliography

[1] Animations and touch events. 79

[2] Respond to touch events. 79

[3] (11/21/2014). Lock screen bypass flaw found in samsung androids. 1, 12, 25

[4] (11/23/2014). sprint-and-lookout-survey. 1, 12, 25

[5] (2014). Friends and family dataset - publications and findings. 39, 61, 68, 78, 83, 84

[6] (2015). Jensenshannon divergence. 4

[7] Abramson, M. and Aha, D. W. (2013). User authentication from web browsing behavior. Technical Report 12-121-4465, DTIC Document. 43, 60

[8] Aharony, N., Pan, W., Ip, C., Khayal, I., and Pentland, A. (2011). Social fmri: Investigating and shaping social mechanisms in the real world. *Pervasive and Mobile Computing*, 7(6):643–659. 61

[9] Alzubaidi, A. and Kalita, J. (2016). Authentication of smartphone users using behavioral biometrics. *IEEE Communications Surveys & Tutorials*, 18(3):1998–2026. 6, 59

[10] Andriotis, P., Oikonomou, G., Mylonas, A., and Tryfonas, T. (2016). A study on usability and security features of the android pattern lock screen. *Information & Computer Security*, 24(1):53–72. 1

[11] Andrzejewski, D., Mulhern, A., Liblit, B., and Zhu, X. (2007). Statistical debugging using latent topic models. In *Machine Learning: ECML 2007*, pages 6–17. Springer. 44

[12] Aziz, A. and Diffie, W. (1994). Privacy and authentication for wireless local area networks. *IEEE Personal Communications*, 1(1):25–31. 100

[13] Babu, B. M. and Bhanu, M. S. (2015). Prevention of insider attacks by integrating behavior analysis with risk based access control model to protect cloud. *Procedia Computer Science*, 54:157–166. 58

[14] Bellovin, S. M. and Merritt, M. (1992). Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *Research in Security and Privacy, 1992. Proceedings., 1992 IEEE Computer Society Symposium on*, pages 72–84. IEEE. 76

[15] Berndt, D. J. and Clifford, J. (1994). Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA. 57

[16] Bishop, C. M. (2006). Pattern recognition. *Machine Learning*, 128:1–58. 59

[17] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022. 12, 44, 47

[18] Bo, C., Zhang, L., Jung, T., Han, J., Li, X.-Y., and Wang, Y. (2014). Continuous user identification via touch and movement behavioral biometrics. In *Performance Computing and Communications Conference (IPCCC), 2014 IEEE International*, pages 1–8. IEEE. 5, 6, 13, 15, 27, 43, 45, 58, 59

[19] Bo, C., Zhang, L., Li, X.-Y., Huang, Q., and Wang, Y. (2013). Silentsense: silent user identification via touch and movement behavioral biometrics. In *Proceedings of the 19th annual international conference on Mobile computing & networking*, pages 187–190. ACM. 2, 6, 27, 43, 45, 56, 57, 58, 59, 60, 64, 102

[20] Bruzzone, L. and Prieto, D. F. (2001). Unsupervised retraining of a maximum likelihood classifier for the analysis of multitemporal remote sensing images. *Geoscience and Remote Sensing, IEEE Transactions on*, 39(2):456–460. 13

[21] Castelluccia, C., Duermuth, M., Golla, M., and Deniz, F. (2017). Towards implicit visual memory-based authentication. In *Network and Distributed System Security Symposium (NDSS)*. 56, 57

[22] Chow, R., Jakobsson, M., Masuoka, R., Molina, J., Niu, Y., Shi, E., and Song, Z. (2010). Authentication in the clouds: a framework and its application to mobile users. In *Proceedings of the 2010 ACM workshop on Cloud computing security workshop*, pages 1–6. ACM. 3, 12, 16, 27

[23] Clarke, N. L. and Furnell, S. M. (2007). Authenticating mobile phone users using keystroke analysis. *International Journal of Information Security*, 6(1):1–14. 2, 57

[24] Conti, M., Zachia-Zlatea, I., and Crispo, B. (2011). Mind how you answer me!: transparently authenticating the user of a smartphone when answering or placing a call. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, pages 249–259. ACM. 27, 45, 58

[25] Crampton, J. and Huth, M. (2010). Towards an access-control framework for countering insider threats. In *Insider Threats in Cyber Security*, pages 173–195. Springer. 58

[26] Crawford, H., Renaud, K., and Storer, T. (2013). A framework for continuous, transparent mobile device authentication. *Computers & Security*, 39:127–136. 57

[27] De Luca, A., Hang, A., Brudy, F., Lindner, C., and Hussmann, H. (2012). Touch me once and i know it's you!: implicit authentication based on touch screen patterns. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, pages 987–996. ACM. 2, 6, 12, 14, 27, 43, 45, 57

[28] Draffin, B., Zhu, J., and Zhang, J. (2013). Keysens: Passive user authentication through micro-behavior modeling of soft keyboard interaction. In *International Conference on Mobile Computing, Applications, and Services*, pages 184–201. Springer. 60

[29] Farrahi, K. and Gatica-Perez, D. (2011). Discovering routines from large-scale human locations using probabilistic topic models. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(1):3. 2, 25, 44, 48

[30] Fawcett, T. and Provost, F. J. (1996). Combining data mining and machine learning for effective user profiling. In *KDD*, pages 8–13. 4

[31] Fei-Fei, L. and Perona, P. (2005). A bayesian hierarchical model for learning natural scene categories. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 524–531. IEEE. 44

[32] Feng, T., Yang, J., Yan, Z., Tapia, E. M., and Shi, W. (2014). Tips: Context-aware implicit user identification using touch screen in uncontrolled environments. In *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications*, page 9. ACM. 2, 4, 6, 56, 57

[33] Ferraiolo, D. F., Sandhu, R., Gavrila, S., Kuhn, D. R., and Chandramouli, R. (2001). Proposed nist standard for role-based access control. *ACM Transactions on Information and System Security (TISSEC)*, 4(3):224–274. 58, 62

[34] Francis, L., Mayes, K., Hancke, G., and Markantonakis, K. (2010). A location based security framework for authenticating mobile phones. In *Proceedings of the 2nd International Workshop on Middleware for Pervasive Mobile and Embedded Computing*. 12

[35] Frank, J., Mannor, S., and Precup, D. (2010). Activity and gait recognition with time-delay embeddings. In *AAAI*. 2, 57, 102

[36] Frank, M., Biedert, R., Ma, E.-D., Martinovic, I., and Song, D. (2013). Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication. *Information Forensics and Security, IEEE Transactions on*, 8(1):136–148. 4, 6, 43, 59, 60

[37] Gafurov, D., Helkala, K., and Søndrol, T. (2006a). Biometric gait authentication using accelerometer sensor. *Journal of computers*, 7:51–59. 4

[38] Gafurov, D., Helkala, K., and Søndrol, T. (2006b). Biometric gait authentication using accelerometer sensor. *JCP*, 1(7):51–59. 57

[39] Gascon, H., Uellenbeck, S., Wolf, C., and Rieck, K. (2014). Continuous authentication on mobile devices by analysis of typing motion behavior. *Sicherheit 2014–Sicherheit, Schutz und Zuverlässigkeit*. 5, 6, 58, 59

[40] Georghiades, A. S., Belhumeur, P. N., and Kriegman, D. J. (2001). From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6):643–660. 2

[41] Giuffrida, C., Majdanik, K., Conti, M., and Bos, H. (2014). I sensed it was you: authenticating mobile users with sensor-enhanced keystroke dynamics. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 92–111. Springer. 4, 27, 45, 58

[42] Gurary, J., Zhu, Y., Alnahash, N., and Fu, H. (2016). Implicit authentication for mobile devices using typing behavior. In *International Conference on Human Aspects of Information Security, Privacy, and Trust*, pages 25–36. Springer. 56

[43] Hayashi, E., Riva, O., Strauss, K., Brush, A., and Schechter, S. (2012). Goldilocks and the two mobile devices: going beyond all-or-nothing access to a device's applications. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, page 2. ACM. 58, 62

[44] Heinrich, G. (2005). Parameter estimation for text analysis. Technical report, Technical report. 4, 15, 16, 53

[45] Hoekstra, M., Lal, R., Pappachan, P., Phegade, V., and Del Cuvillo, J. (2013). Using innovative instructions to create trustworthy software solutions. In *HASP@ ISCA*, page 11. 108

[46] Hospedales, T. M., Li, J., Gong, S., and Xiang, T. (2011). Identifying rare and subtle behaviors: A weakly supervised joint topic model. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(12):2451–2464. 44, 45

[47] Huynh, T., Fritz, M., and Schiele, B. (2008). Discovery of activity patterns using topic models. In *Proceedings of the 10th international conference on Ubiquitous computing*, pages 10–19. ACM. 44, 48

[48] Jain, A. K., Ross, A., and Prabhakar, S. (2004). An introduction to biometric recognition. *IEEE Transactions on circuits and systems for video technology*, 14(1):4–20. 57

[49] Jakobsson, M., Shi, E., Golle, P., and Chow, R. (2009). Implicit authentication for mobile devices. In *Proceedings of the 4th USENIX conference on Hot topics in security*. 1, 25, 27, 31, 45, 64, 98

[50] Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45. 60

[51] Kate, M. K., Hake, M. J., Ahire, M. S., and Shelke, M. H. (2017). Authentication of smartphone users using behavioral biometrics and opass technique. *International Journal*, 2(1). 58

[52] Khan, H., Atwater, A., and Hengartner, U. (2014a). A comparative evaluation of implicit authentication schemes. In *International Workshop on Recent Advances in Intrusion Detection*, pages 255–275. Springer. 2, 6, 7, 57

[53] Khan, H., Atwater, A., and Hengartner, U. (2014b). Itus: an implicit authentication framework for android. In *Proceedings of the 20th annual international conference on Mobile computing and networking*, pages 507–518. ACM. 1, 4, 5, 6, 11, 12, 13, 14, 15, 26, 27, 31, 43, 45, 58, 59, 98

[54] Khan, H. and Hengartner, U. (2014a). Towards application-centric implicit authentication on smartphones. In *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications*, page 10. ACM. 4, 26, 43, 45

[55] Khan, H. and Hengartner, U. (2014b). Towards application-centric implicit authentication on smartphones. 11

[56] Khan, H., Hengartner, U., and Vogel, D. (2016). Targeted mimicry attacks on touch input based implicit authentication schemes. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, pages 387–398. ACM. 6, 7, 61, 62, 63, 69, 103

[57] Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, pages 79–86. 16

[58] Kumar, M. (2004). On the weaknesses and improvements of an efficient password based remote user authentication scheme using smart cards. *IACR Cryptology ePrint Archive*, 2004:163. 76

[59] Langley, D. J., Hoeve, M. C., Ortt, J. R., Pals, N., and van der Vecht, B. (2014). Patterns of herding and their occurrence in an online setting. *Journal of Interactive Marketing*, 28(1):16–25. 44

[60] Lee, W.-H. and Lee, R. (2016). Implicit sensor-based authentication of smartphone users with smartwatch. 31, 34, 85, 108

[61] Lee, W.-H. and Lee, R. B. (2015a). Implicit authentication for smartphone security. In *International Conference on Information Systems Security and Privacy*, pages 160–176. Springer. 4, 60

[62] Lee, W.-H. and Lee, R. B. (2015b). Multi-sensor authentication to improve smartphone security. In *Conference on Information Systems Security and Privacy*. 2, 4, 6, 26, 27, 31, 45, 55, 56, 57, 58, 59, 64, 85, 98

[63] Li, L., Zhao, X., and Xue, G. (2013). Unobservable re-authentication for smartphones. In *NDSS*, pages 1–16. 6, 13, 15, 27, 43, 45, 58, 59, 60

[64] Lin, J. (1991). Divergence measures based on the shannon entropy. *Information Theory, IEEE Transactions on*, 37(1):145–151. 28, 31, 49

[65] Lyle, J., Paverd, A., King-Lacroix, J., Atzeni, A., Virji, H., Flechais, I., and Faily, S. (2013). Personal pki for the smart device era. In *Public Key Infrastructures, Services and Applications*, pages 69–84. Springer. 15

[66] Maiorana, E., Campisi, P., González-Carballo, N., and Neri, A. (2011). Keystroke dynamics authentication for mobile phones. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, pages 21–26. ACM. 2, 57

[67] Mäntyjärvi, J., Lindholm, M., Vildjiounaite, E., Mäkelä, S.-M., and Ailisto, H. (2005). Identifying users of portable devices from gait pattern with accelerometers.

In *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on*, volume 2, pages ii–973. IEEE. 57

[68] McLeod, S. (2007). Bf skinner: Operant conditioning. *Retrieved September*, 9:2009. 77, 82

[Measure] Measure, F. Kernel density estimator. 59

[70] Miluzzo, E., Varshavsky, A., Balakrishnan, S., and Choudhury, R. R. (2012). Tapprints: your finger taps have fingerprints. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 323–336. ACM. 27, 45, 58

[71] Monrose, F. and Rubin, A. (1997). Authentication via keystroke dynamics. In *Proceedings of the 4th ACM conference on Computer and communications security*, pages 48–56. ACM. 14

[72] Muaaz, M. and Mayrhofer, R. (2013). An analysis of different approaches to gait recognition using cell phone based accelerometers. In *Proceedings of International Conference on Advances in Mobile Computing & Multimedia*, page 293. ACM. 57

[73] Murray, H. and Malone, D. (2018). Exploring the impact of password dataset distribution on guessing. In *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, pages 1–8. IEEE. 76

[74] Niebles, J. C., Wang, H., and Fei-Fei, L. (2008). Unsupervised learning of human action categories using spatial-temporal words. *International journal of computer vision*, 79(3):299–318. 44

[75] Platt, J. et al. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74. 59

[76] Ramage, D., Manning, C. D., and Dumais, S. (2011). Partially labeled topic models for interpretable text mining. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 457–465. ACM. 31, 47, 53

[77] Ravi, N., Dandekar, N., Mysore, P., and Littman, M. L. (2005). Activity recognition from accelerometer data. In *AAAI*, volume 5, pages 1541–1546. 57, 64

[78] Richman, J. S. and Moorman, J. R. (2000). Physiological time-series analysis using approximate entropy and sample entropy. *American Journal of Physiology-Heart and Circulatory Physiology*, 278(6):H2039–H2049. 15

[79] Riva, O., Qin, C., Strauss, K., and Lymberopoulos, D. (2012). Progressive authentication: Deciding when to authenticate on mobile phones. In *USENIX Security Symposium*, pages 301–316. 57

[80] Rosen-Zvi, M., Griffiths, T., Steyvers, M., and Smyth, P. (2004). The author-topic model for authors and documents. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 487–494. AUAI Press. 12, 44

[81] Safa, N. A., Safavi-Naini, R., and Shahandashti, S. F. (2014). Privacy-preserving implicit authentication. In *IFIP International Information Security Conference*, pages 471–484. Springer. 4, 100

[82] Sandhu, R. S. (1993). Lattice-based access control models. *Computer*, (11):9–19. 58, 62

[83] Scott, D. W. (2008). Kernel density estimators. *Multivariate Density Estimation: Theory, Practice, and Visualization*, pages 125–193. 59

[84] Serwadda, A., Phoha, V. V., and Wang, Z. (2013). Which verifiers work?: A benchmark evaluation of touch-based authentication algorithms. In *Biometrics: Theory, Applications and Systems (BTAS), 2013 IEEE Sixth International Conference on*, pages 1–8. IEEE. 57

[85] Shahzad, M., Liu, A. X., and Samuel, A. (2013). Secure unlocking of mobile touch screen devices by simple gestures: You can see it but you can not do it. In *Proceedings of the 19th annual international conference on Mobile computing & networking*, pages 39–50. ACM. 27, 45, 56, 57

[86] Sheng, Y., Phoha, V. V., and Rovnyak, S. M. (2005). A parallel decision tree-based method for user authentication based on keystroke patterns. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 35(4):826–833. 14

[87] Shi, E., Niu, Y., Jakobsson, M., and Chow, R. (2011a). Implicit authentication through learning user behavior. In *Information Security*. 4, 11, 12, 17, 26, 27, 40, 55, 57, 84

[88] Shi, W., Yang, F., Jiang, Y., Yang, F., and Xiong, Y. (2011b). Senguard: Passive user identification on smartphones using multiple sensors. In *Wireless and Mobile Computing, Networking and Communications (WiMob), 2011 IEEE 7th International Conference on*, pages 141–148. IEEE. 2, 43, 56, 57, 64

[89] Sinclair, S. and Smith, S. W. (2008). Preventative directions for insider threat mitigation via access control. In *Insider Attack and Cyber Security*, pages 165–194. Springer. 58

[90] Skinner, B. F. (1971). Operant conditioning. *The encyclopedia of education*, 7:29–33. 77, 82

[91] Skinner, B. F. (1990). *The behavior of organisms: An experimental analysis*. BF Skinner Foundation. 77, 82

[92] Studer, A. and Perrig, A. (2010). Mobile user location-specific encryption (mule): using your office as your password. In *Proceedings of the third ACM conference on Wireless network security*, pages 151–162. ACM. 2, 57

[93] Sun, J., Zhang, R., Zhang, J., and Zhang, Y. (2014). Touchin: Sightless two-factor authentication on multi-touch mobile devices. In *CNS, 2014 IEEE Conference on*, pages 436–444. IEEE. 6, 12, 14, 26, 27, 45, 56, 57

[94] Tamviruzzaman, M., Ahamed, S. I., Hasan, C. S., and O'brien, C. (2009). epet: when cellular phone learns to recognize its owner. In *Proceedings of the 2nd ACM workshop on Assurable and usable security configuration*, pages 13–18. ACM. 2, 4, 12, 14, 25, 56, 58

[95] Tey, C. M., Gupta, P., and Gao, D. (2013). I can be you: Questioning the use of keystroke dynamics as biometrics. 57

[96] Thomas, K., Grier, C., Ma, J., Paxson, V., and Song, D. (2011). Design and evaluation of a real-time url spam filtering service. In *Security and Privacy (SP), 2011 IEEE Symposium on*, pages 447–462. IEEE. 13, 14

[97] Wang, W., Liu, A. X., Shahzad, M., Ling, K., and Lu, S. (2015a). Understanding and modeling of wifi signal based human activity recognition. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, pages 65–76. ACM. 2, 25, 60

[98] Wang, Z., Liao, J., Cao, Q., Qi, H., and Wang, Z. (2015b). Friendbook: a semantic-based friend recommendation system for social networks. *Mobile Computing, IEEE Transactions on*, 14(3):538–551. 44, 48

[99] Welch, G. and Bishop, G. (1995). An introduction to the kalman filter. 58

[100] Wood, H. M. (1977). *The use of passwords for controlled access to computer resources.* US Department of Commerce, National Bureau of Standards. 12, 14

[101] Wright, J., Dawson Jr, M. E., and Omar, M. (2012). Cyber security and mobile threats: The need for antivirus applications for smart phones. *Journal of Information Systems Technology and Planning*, 5(14):40–60. 11

[102] Xing, X., Li, M., Hu, W., Huang, W., Song, G., and Xie, K. (2014). A spatial-temporal topic segmentation model for human mobile behavior. In *Web-Age Information Management*, pages 255–267. Springer. 44, 45

[103] Yang, H.-K. and An, Y.-H. (2012). Security weaknesses and improvements of a fingerprint-based remote user authentication scheme using smart cards. *International Journal of Advancements in Computing Technology*, 4(1). 2, 12

[104] Yang, Y. and Sun, J. (2017). Energy-efficient w-layer for behavior-based implicit authentication on mobile devices. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE. 1, 2, 4, 6, 57, 58, 81

[105] Yang, Y., Sun, J., and Guo, L. (2016). Personaia: A lightweight implicit authentication system based on customized user behavior selection. *IEEE Transactions on Dependable and Secure Computing*. 2, 4, 55, 57, 58, 64

[106] Yang, Y., Sun, J., and Li, P. (2015). Model retraining and dynamic privilege-based access control for implicit authentication systems. In *IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*. ACM. 5, 6, 58, 59, 63, 85, 107

[107] Ye, G., Tang, Z., Fang, D., Chen, X., Kim, K. I., Taylor, B., and Wang, Z. (2017). Cracking android pattern lock in five attempts. 1

[108] Yi, S., Naldurg, P., and Kravets, R. (2001). Security-aware ad hoc routing for wireless networks. In *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 299–302. ACM. 58, 62

[109] Zhao, W., Chellappa, R., Phillips, P. J., and Rosenfeld, A. (2003). Face recognition: A literature survey. *ACM computing surveys (CSUR)*, 35(4):399–458. 2

[110] Zhao, X., Feng, T., and Shi, W. (2013). Continuous mobile authentication using a novel graphic touch gesture feature. In *Biometrics: Theory, Applications and Systems (BTAS), 2013 IEEE Sixth International Conference on*, pages 1–6. IEEE. 5, 57

[111] Zhu, J., Wu, P., Wang, X., and Zhang, J. (2013). Sensec: Mobile security through passive sensing. In *Computing, Networking and Communications (ICNC), 2013 International Conference on*, pages 1128–1133. IEEE. 13, 15

Vita

Yingyuan Yang is a six-year Ph.D. student in the Department of Electrical Engineering & Computer Science at the University of Tennessee Knoxville working with Prof. Jinyuan Sun. He received his Bachelors in Computer Science from Yunnan University, China and Master's in Computer Science from Temple University, Philadelphia. Yingyuan's research interests include mobile security, machine learning, and implicit authentication. His research papers appear in prestigious journals and conferences such as IEEE TDSC and INFOCOM.