# Retraining and Dynamic Privilege for Implicit Authentication Systems

Yingyuan Yang*, Jinyuan Stella Sun*, Chi Zhang‡ and Pan Li§
*University of Tennessee, Knoxville, TN, 37996 USA
Email: {yyang57, jysun}@utk.edu
‡University of Science and Technology of China, Anhui, 230027 China
Email:chizhang@ustc.edu.cn
§Mississippi State University, MS, 39762 USA
Email:li@ece.msstate.edu

***Abstract*—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 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.**

## I. 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 [1]. 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 [2]. 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 [3]. 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 the smart devices and achieve user identification [4]. User behavioral data, such as walking style, swipe speed and location, are used to train user behavior models which are then used as 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 [5]. Even though biometrics are much harder to be stolen than passwords, there have been research works showing the feasibility of biometrics forgery [6]. 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 [7]. 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 password [8] .

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 [9], [10], and there are quite some research works dealing with how to select suitable machine learning algorithms for various activity types such as those obtained from touch [11], [12], accelerometer [13], location [14], etc., as well as how to provide a general framework for IA [1], [4], [15]. In this paper, on the other hand, we focus on two critical and difficult problems that affect the practical deployment of IA systems: model retraining and handling

978-1-4673-9101-6/15 $31.00 © 2015 IEEE
DOI 10.1109/MASS.2015.69

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 paper 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 [4], [16] or similar [17], [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 [19], [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. 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.

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

In this paper, we strive to conquer the above technical challenges to obtain improved and more realistic IA systems. Our main contributions in this paper include:

- We compare the different machine learning methods and provide a novel technique to achieve optimal retraining frequency. The results show that our technique can successfully detect the accuracy change and automatically determine the best retraining frequency.
- We propose a dynamic privilege mechanism for fine-grained access control to achieve secure and usable IA. Our method can not only reduce the negative effect of authentication failure on legitimate user but also achieve a better accuracy than the traditional lock-only method.
- We evaluate our proposed technique and mechanism using real user data. The dataset contains data from 130 persons with 31 features and 132960052 records. Our simulation results show the effectiveness and efficiency of our proposed solutions.

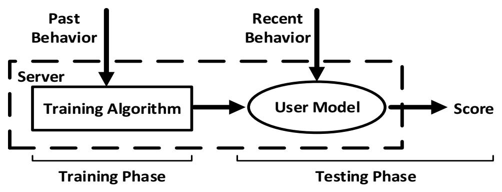
## II. Preliminary

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.

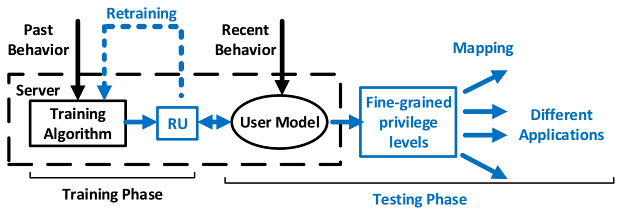
### A. *Entropy or Timeline*

Entropy, as it relates to dynamical systems, is the rate of information production [21]. In machine learning, the differences in between entropies are used to measure the similarity in between the testing data and the training data [22].

In this paper, 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.



(a) Basic IA framework



(b) Dynamic IA framework

Fig. 1: IA Frameworks

### B. KL Divergence and JS-Distance

The common way to measure the entropy difference in between two states is by calculating the KL divergence [23] between them. The KL divergence in between two discrete random variables X and Y is defined as:

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

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 [22]. Thus, we use a smoothed and symmetric extension, JS-dis for measuring the similarity. Using (1), we can further define the JS-dis as:

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

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

## III. The Proposed Implicit Authentication Framework And Adversary Model

The basic IA framework is shown in Fig. 1 (a), which was first proposed in [15]. We augment it with two key functionalities, retraining and dynamic privilege, to build practical IA systems.

As shown in Fig. 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. 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. 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 [1]. 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.

## IV. 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.

### A. How to Retrain

To measure the difference in between two individual samples in the training and testing dataset, using JS-dis in Eq. (2) we have:

$$D_{JS}(E||R) = \frac{1}{2}[D_{KL}(E||M) + D_{KL}(R||M)]. \quad (3)$$

$E$ in Eq. (3) 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:

$$\overline{D_{JS}^{(n)}} = \frac{\sum_{k=1}^{K} D_{JS}(E^{(n)}||R_k)}{K}, \tag{4}$$

where $K$ is the number of training samples (we call it stride), and $(n)$ denotes the $n$th 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 paper, 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:

$$s^{(n)} = (\frac{1}{K}\sum_{k=1}^{K}(D_{JS}(E^{(n)}||R_k) - \overline{D_{JS}^{(n)}})^2)^{\frac{1}{2}}. \tag{5}$$

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).

### B. 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 $\varepsilon$ which represents the acceptable distance such that the accuracy within $\varepsilon$ is enough high. If the most recent distance is larger than $\varepsilon$, 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 $\varepsilon$. 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.

### C. Retraining Process

Fig. 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, 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 paper, the training data could come from the original training or the previous retraining.

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, the current sample's standard deviation value is marked as "s". The final step is to compare the standard deviation with the predefined $\varepsilon$. For different implementations, we could choose different values of $\varepsilon$. 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.

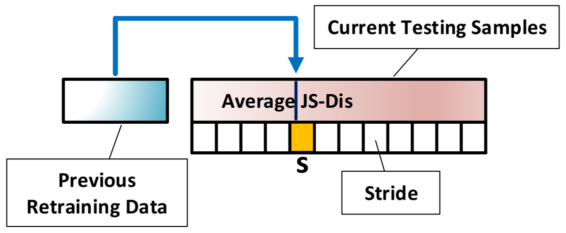


Fig. 2: Retraining Process

## V. 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.

### A. 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 such level 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 IV-A.

### B. 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 samples, and further decide the appropriate privilege for the user at this time.

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. 3 shows how the dynamic privilege access control works. The blue stripe in the average JS-dis array indicates the average distance ($\overline{D_{JSc}^{(n)}}$) in between the current testing sample and the previous retraining data (same as Fig. 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

**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;
```

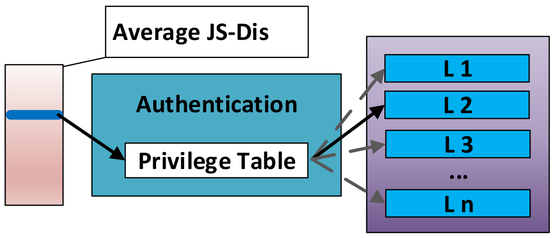


Fig. 3: Dynamic Privilege Overview

privilege level, e.g., L2 is a higher level than L3. To update user privilege, the dynamic privilege mechanism first compares $\overline{D_{JSc}^{(n)}}$ with the predefined rule in the **current** privilege level of the user. For example, if the current level is L2, the mechanism will compare $\overline{D_{JSc}^{(n)}}$ with the L2 rule, and will lower the current privilege if $\overline{D_{JSc}^{(n)}}$ is larger and keep the current privilege unchanged otherwise.

It is possible that $\overline{D_{JSc}^{(n)}}$ 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 $\overline{D_{JSc}^{(n)}}$ 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.

## VI. Performance Evaluation

In this section, we will verify our methods using the real dataset. There are two methods we want to test. The first one is using JS-dis to decide the retraining frequency. The second one is dynamic privilege mechanism and the privilege level selection.

### A. Dataset

In order to evaluate the first method, we want to find a dataset that records the long-term behavioral data for different users, and we require the dataset containing as many people as possible to test our second method. In addition, we also hope the dataset contains enough sensors. For these reasons, we select MIT friends and family [24] dataset as our testing database.

This dataset contains 130 participants with total number of 9 types of data (GPS, accelerometer, SMS, app installation, battery usage, call logs, app running, blue-tooth devices log). The dataset contains more than 5 months data.

### B. Accuracy in Long-Term Testing

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.

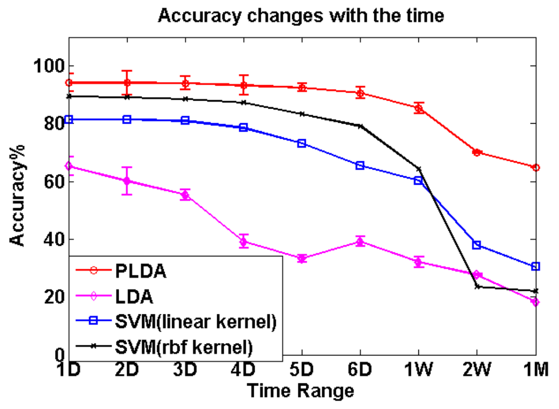


Fig. 4: Accuracy for different machine learning methods

We run several different kinds of machine learning methods using k fold cross validation method. For each different kinds of machine learning algorithm, we record its accuracy[2]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 contains 5 months 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. 4 shows the accuracy curves for PLDA, LDA, SVM with linear kernel and SVM with RBF kernel in one month time range. Since LDA uses unsupervised learning method, the accuracy is much lower than the others. From Fig. 4, 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[3]. 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 device for a legitimate user. To prevent such unfriendly locking, we need to retrain the model. The following subsection will present detail of retraining, which can be done automatically.

### C. JS-Distance in Long-Term Testing

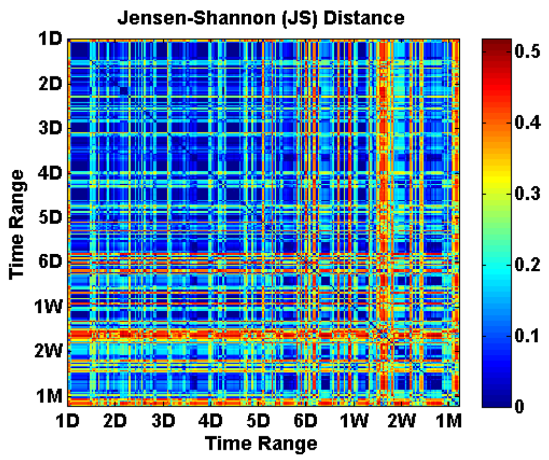


Fig. 5: JS-dis heat map: red color indicates long JS-dis and deep blue indicates short JS-dis

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 second to calculate the average JS-dis in between 1 day's samples for each person.

The Fig. 5 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 maximum 0.5. Deep blue indicates short JS-dis

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

$$\frac{TP+TN}{TP+FP+TN+FN} \tag{6}$$

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

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

with minimum 0. The 0 value is resulted by compared with themselves.

From the Fig. 5 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. 4), the JS-dis also fluctuates (Fig. 5) - 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.

## D. Retraining Frequency

By taking the standard deviation of each stride $k$ in the first row in Fig. 5, we can clear see the fluctuation of the JS-dis. All the strides reflect the accuracy change rate except the first stride (containing 0 distance).

From Fig. 5 we can clearly see that there exists fluctuations in between day5 to week2. These fluctuations correspond to the accuracy changes in the Fig. 4. The largest value of the difference in between any two standard deviations is the value within the range of 6D to 2W with value of 0.024, and we can set the legitimate threshold $\varepsilon = 0.02$ for this dataset.

## E. 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 user send feedback to training server, but we know its correctness using empirical data. We assume the empirical data reflect the key attributes of legitimate user. Actually, the retraining process can reinforce the effectiveness of empirical data, and we can readjust the privilege level in each retraining.

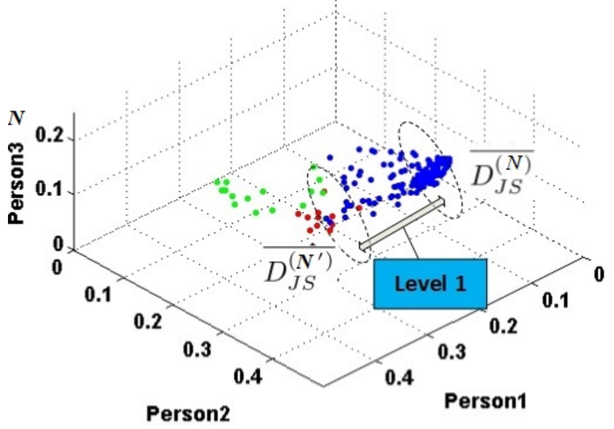


Fig. 6: Level in dynamic privilege

In this experiment, we compare the JS-distances using the data of same person and take the average of these values. We select three persons from the dataset to demonstrate our method and the result is presented in Fig. 6. 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 same person - Person1). Fig. 6 shows the result of this test. The blue points indicate the TP for identifying Person1 as legitimate user. From the Fig. 6, 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 person in the whole dataset, called $\overline{D_{JS}^{(N)}}$. In Fig. 6, 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, 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.

*1) 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 I.

Using $\overline{D_{JS}^{(K)}}$ in Table I 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 I. In the testing phase, each stride[4]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 I 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 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.

TABLE I: Levels and their performances

| | L1 | L2 | L3 | L4 |
|---|---|---|---|---|
| $\overline{D_{JS}^{(K)}}$ | 0.342 | 0.387 | 0.467 | |
| $FN$ | 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.

*2) 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, we use the data from Person2 and Person3 to rerun the test based on the training data of Person1. In this paper, 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 I 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.

[4]Please refer section IV-C (Retraining Process).

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.

## VII. Related Work

This paper is most related to implicit authentication (IA)-mechanisms based on user behavior [5], especially those implemented on smart devices [11]–[13]. 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 paper 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 [12], Sun et al. propose a multi-touch system to authenticate user based on the motion of different fingers. De Luca et al. [11] use touch screen pattern as main attribute to identify different persons. From another angle, Tamviruzzaman et al. [13] 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 [4] to simplify the IA development. Most of these works focus on one time training without further retraining. In [25], 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. [26] introduce a technique that can retrain part of the system when a sufficient sample has been collected. In [19], 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 [4], [16]–[18], [27]. 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.

## VIII. Conclusion and Future Work

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 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.

We tested our methods on a dataset of 130 persons with more than 5-month 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 showed that the dynamic privilege mechanism can largely reduce the effect of false negative authentication failure.

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.

## IX. Acknowledgements

This work was partially supported by the US National Science Foundation (NSF) under grants CNS-1422665. The work of P. Li was supported by the U.S. National Science Foundation under grants CNS-1149786 and CNS-1343220.

## References

[1] E. Shi, Y. Niu, M. Jakobsson, and R. Chow, "Implicit authentication through learning user behavior," in *Information Security*, 2011.

[2] J. Wright, M. E. Dawson Jr, and M. Omar, "Cyber security and mobile threats: The need for antivirus applications for smart phones," *Journal of Information Systems Technology and Planning*, vol. 5, no. 14, pp. 40–60, 2012.

[3] H. Khan and U. Hengartner, "Towards application-centric implicit authentication on smartphones," 2014.

[4] H. Khan, A. Atwater, and U. Hengartner, "Itus: an implicit authentication framework for android," in *Proceedings of the 20th annual international conference on Mobile computing and networking*. ACM, 2014, pp. 507–518.

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

[6] H.-K. Yang and Y.-H. An, "Security weaknesses and improvements of a fingerprint-based remote user authentication scheme using smart cards," *International Journal of Advancements in Computing Technology*, vol. 4, no. 1, 2012.

[7] "sprint-and-lookout-survey," 11/23/2014. [Online]. Available: https://blog.lookout.com/blog/2013/10/21/sprint-and-lookout-survey/

[8] "Lock screen bypass flaw found in samsung androids," 11/21/2014. [Online]. Available: http://threatpost.com/lock-screen-bypass-flaw-found-samsung-androids-030413/77580

[9] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *the Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003.

[10] M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P. Smyth, "The author-topic model for authors and documents," in *Proceedings of the 20th conference on Uncertainty in artificial intelligence*. AUAI Press, 2004, pp. 487–494.

[11] A. De Luca, A. Hang, F. Brudy, C. Lindner, and H. Hussmann, "Touch me once and i know it's you!: implicit authentication based on touch screen patterns," in *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*. ACM, 2012, pp. 987–996.

[12] J. Sun, R. Zhang, J. Zhang, and Y. Zhang, "Touchin: Sightless two-factor authentication on multi-touch mobile devices," *arXiv preprint arXiv:1402.1216*, 2014.

[13] M. Tamviruzzaman, S. I. Ahamed, C. S. Hasan, and C. O'brien, "epet: when cellular phone learns to recognize its owner," in *Proceedings of the 2nd ACM workshop on Assurable and usable security configuration*. ACM, 2009, pp. 13–18.

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

[15] R. Chow, M. Jakobsson, R. Masuoka, J. Molina, Y. Niu, E. Shi, and Z. Song, "Authentication in the clouds: a framework and its application to mobile users," in *Proceedings of the 2010 ACM workshop on Cloud computing security workshop*. ACM, 2010, pp. 1–6.

[16] J. Zhu, P. Wu, X. Wang, and J. Zhang, "Sensec: Mobile security through passive sensing," in *Computing, Networking and Communications (ICNC), 2013 International Conference on*. IEEE, 2013, pp. 1128–1133.

[17] C. Bo, L. Zhang, T. Jung, J. Han, X.-Y. Li, and Y. Wang, "Continuous user identification via touch and movement behavioral biometrics," in *Performance Computing and Communications Conference (IPCCC), 2014 IEEE International*. IEEE, 2014, pp. 1–8.

[18] L. Li, X. Zhao, and G. Xue, "Unobservable re-authentication for smartphones." in *NDSS*, 2013.

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

[20] L. Bruzzone and D. F. Prieto, "Unsupervised retraining of a maximum likelihood classifier for the analysis of multitemporal remote sensing images," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 39, no. 2, pp. 456–460, 2001.

[21] J. S. Richman and J. R. Moorman, "Physiological time-series analysis using approximate entropy and sample entropy," *American Journal of Physiology-Heart and Circulatory Physiology*, vol. 278, no. 6, pp. H2039–H2049, 2000.

[22] G. Heinrich, "Parameter estimation for text analysis," Technical report, Tech. Rep., 2005.

[23] S. Kullback and R. A. Leibler, "On information and sufficiency," *The Annals of Mathematical Statistics*, pp. 79–86, 1951.

[24] "Friends and family dataset - publications and findings," 2014. [Online]. Available: http://realitycommons.media.mit.edu/friendsdataset3.html

[25] F. Monrose and A. Rubin, "Authentication via keystroke dynamics," in *Proceedings of the 4th ACM conference on Computer and communications security*. ACM, 1997, pp. 48–56.

[26] Y. Sheng, V. V. Phoha, and S. M. Rovnyak, "A parallel decision tree-based method for user authentication based on keystroke patterns," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 35, no. 4, pp. 826–833, 2005.

[27] J. Lyle, A. Paverd, J. King-Lacroix, A. Atzeni, H. Virji, I. Flechais, and S. Faily, "Personal pki for the smart device era," in *Public Key Infrastructures, Services and Applications*. Springer, 2013, pp. 69–84.