Empirical techniques and algorithms to develop a resilient non-supervised touch-based authentication system

by
Nikhil Palaskar

Presented to the Computer Science, Engineering and Physics Faculty
at the University of Michigan Flint
in partial fulfillment of the requirements for the
Master of Science in Computer and Information Science

First Reader ______________________
Dr. Zahid Syed

Second Reader ______________________
Dr. Mike Farmer

December 14, 2016

Abstract

Touch dynamics (or touch based authentication) refers to a behavioral biometric for touchscreen devices wherein a user is authenticated based on his/her executed touch gestures. This work addresses two research topics. We first present a series of empirical techniques to detect habituation in the user's touch profile, its detrimental effect on authentication accuracy and strategies to overcome these effects. Habituation here refers to changes in the user's profile and/or noise within it due to the user's familiarization with the device and software application. With respect to habituation, we show that habituation causes the user's touch profile to evolve significantly and irrevocably over time even after the user is familiar with the device and software application. This phenomenon considerably degrades classifier accuracy. We demonstrate techniques that lower the error rate to 3.68% and sets the benchmark in this field for a realistic test setup. Finally, we quantify the benefits of vote-based reclassification of predicted class labels and show that this technique is vital for achieving high accuracy in realistic touch-based authentication systems.

In the second half, we implement the first ever non-supervised classification algorithm in touch based continual authentication. This scheme incorporates clustering into the traditional supervised algorithm. We reduce the mis-classification rate by fusing supervised random forest algorithm and non-supervised clustering (either Bayesian learning or simple rule of combinations). Fusing with Bayesian clustering reduced the mis-classification rate by 50% while fusing with simple rule of combination reduced the mis-classification rate by as much as 59.5% averaged over all the users.

DEDICATION

I dedicate this work to my parents

ACKNOWLEDGMENTS

I would like to express my special appreciation and thanks to my adviser Professor Dr. Zahid Syed, you have been a fabulous mentor for me. I would like to thank you for encouraging my research and for allowing me to grow as a researcher. Your advice on both research as well as on my career have been priceless. I would also like to sincerely thank my professor Dr. Micheal Farmer and professor Dr. Murali Mani for advising me during the critical period of my research. I also want to thank you for letting my research be an enjoyable process. I would also especially like to thank all the group members of Big Data group at University of Michigan Flint for allowing me to discuss my ideas and research at different points in time, and for your brilliant comments and suggestions, thanks to all. My sincere thanks to all the staff members of CSEP department for allowing me to access the lab 24*7 and more importantly for providing me the necessary funding to present my research at an international level of conference. All of you have been a great support for me when I needed the most and without your support and help I could have never completed my thesis.

Finally, I must express my very profound gratitude to my parents and my friends for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Contents

List of Figures

List of Tables

1 Introduction & Motivation

Portable touchscreen devices are now ubiquitous: 64% of the U.S. population owns a smartphone and more people rely on smartphones for online access than on desktop computers [1]. By the year 2018, it is projected that more than 50% of users will use smartphones for primary online activity [2]. However, with the growth in portable device usage, there has been a parallel rise in crimes related to cell phones [3]. This growth in usage and popularity attracted many cyber attackers to steal important data via authentication misuse. Smartphone robberies were up 23% in San Fransisco in 2013, and 18% in New York City [4]. Surprisingly, the most common software mechanism for protecting data on such portable devices is a fairly rudimentary PIN based static authentication system. While easy to deploy, it is a one-time authentication system that leave the device vulnerable once the initial phone is unlocked. It has been shown that the phone lock screen on both Android and iOS operating systems are vulnerable to software flaws [5, 6].

One avenue that has shown potential to overcome such limitations is a touch-based continual authentication system. Since the touchscreen is the primary mode of input for interaction with the device, an authentication system that uses these inputs to continually validate the user's identity provides an elegant solution for managing device security.

To develop a continual touch-based authentication system, the *genuine user* initially uses the touchscreen device for a specified period of time. This interaction is via touch *strokes* executed on the device's screen. At the atomic level, the device senses these strokes as a series of points. Each point provides data such as its location on the screen, pressure exerted, timestamp, etc. Using data from all points within a stroke, a set of statistics called *features* is generated for that stroke. The ordered tuple of these features is called a *feature vector.* Every feature vector characterizes that particular stroke. As the genuine user uses the device, the group of feature vectors generated for that user constitutes that user's *genuine data* and serves as his/her *touch profile.*

Similarly, feature vectors from a number of other users are collected to form the *impostor data.* Both the genuine and impostor data are used to train a *classifier* and generate a *user model.* The classifier is a machine learning algorithm. In this work, we use various 2-class based algorithms that require genuine and impostor data to generate a user model.

2 Contributions

2.1 Effects of Habituation

Since touch-based continual authentication is a behavioral biometric, it is more susceptible to noise due to environmental and behavioral factors such as device configuration, user gait, movement and posture [7]. A part of this paper addresses one such factor, i.e. habituation. By this term, we refer to the familiarization process a user undergoes as they become accustomed to the device and the software application. Habituation can manifest itself as two effects:

1. The erraticness in the user's touch-based behavior may decrease over time leading to less 'noise' in the user's touch profile.

2. The user's behavior and style of interaction with the device and app may change over time, i.e. the user's touch profile itself may transform over time.

It should be noted that both effects are independent of each other, i.e. one effect may be observed while the other is not. It is possible that such a phenomenon will significantly degrade classifier accuracy. In this case, the user model update strategy must be developed based on the severity of changes effected by habituation. Teasing apart the effects of habituation and developing an optimal technique to update the user model forms a key part of this work. In summary, this work addresses the following Research Questions (RQ) in Part I:

- **RQ1**: Do users' touch-based profiles exhibit either of the two effects of habituation? If so, do they impact the performance of a touch-based authentication system?

- **RQ2**: Based on the above analysis, what is the most effective strategy for updating the user model?

- **RQ3**: Which classifier algorithm is the most accurate and consistent after taking effects of habituation into account?

- **RQ4**: A quantitative comparison of the benefits of using a vote-based reclassification scheme to post-process classifier class predictions.

- **RQ5**: What is the benchmark performance when using a realistic train-test setup and a vote-based reclassification scheme?

2.2 Effects of different classifiers Fusion mechanism

In Section 2.1 we study and implement different classifier algorithms and provide the detailed benchmark performance when using a realistic train-test setup. Note that these classifier algorithms are all supervised classifier models. Supervised classification is the machine learning task of inferring the class of unlabeled test data based on a trained labeled dataset which consist of genuine as well as impostor data. However, one of the prime issue in continual authentication that haven't yet fully address is the need of an impostor data to train the classifier along with the genuine data to build the user model. This is to make sure that our user model or classifier know the properties or behavior of genuine data as well as impostor data and it can correctly classify between them. The amount of genuine and impostor data to use varies depending on the developers perspective and a type of application his targeting.

Therefore in this paper we also try to introduce the ***First Ever*** unsupervised classifier algorithm to be implemented in continual authentication. We have fused the results of non-supervised and supervised classifier algorithms to check if it has any improvement in predicting the correct class of an unknown test data sample. Two fusion mechanisms has been studied and implemented in this paper viz. Bayesian fusion and simple fusion. Teasing apart the effects of fusion mechanisms and developing an optimal techniques to reduce the mis-classification rate forms a second key part of this paper. In summary, this work addresses the following research questions in Part II of this paper:

1. **RQ1**: Which non-supervised clustering algorithms is the most accurate and consistent on separating the genuine and impostor dataset?
2. **RQ2**: Does fusion of non-supervised and supervised classification helps in improving the continual authentication accuracy?
3. **RQ**3: Can we Minimize the use of impostor dataset used for training the classifier?

3 Thesis Organization

The remainder of this paper is organized as follows:

Section 4 discusses the related literature in this touch-based authentication. Section 5 describes the dataset's characteristics and pre-processing steps undertaken. Part I and Part II address the questions of habituation and fusion mechanisms in continual authentication respectively.

4 Related Work

A number of researchers have referenced the effects of habituation in their work. Among them, Frank et al. [8] and De Luca et al. [9] have reported that their model's accuracy dropped over time and felt the need to update their user model. Habituation seems to be a consistent trend in other works that affects performance and deserves further scrutiny. Frank et al. [8] authenticated users by analyzing their regular device use patterns over time, through 34 different features extracted from touch strokes. This study achieved an equal error rate between 0%-4%. The authors also tested for inter-session authentication that showed that touch-based authentication can be used for long term authentication.

Li et al. [10] evaluated the performance of a live implementation of a smart phone-based touch authentication system. The touch data was collected in the background from 75 users who were asked to freely use the devices for a number of days. The collected data was used to create a SVM-based classifier that exhibited an equal error rate of 3%.

Feng et al. [11] used 53 touch and gesture features for classification. Additionally, they created a special digital sensor glove to achieve highly accurate continuous identification. The hand glove was used to capture 36 features when users performed touch activity. The glove data was collected for 11 subjects and the classifier trained using Random Forest, J48, and Bayes network algorithms. The authors achieved an accuracy of 2.15% FAR and 1.63% FRR when the digital glove was used. Without the glove, they reported an accuracy of 11.96% FAR and 8.53% FRR.

The closest study to ours that studied habituation empirically is by Xu et. al. [12]. The authors analyzed habituation using data from 3 genuine users collected over a period of 30 days. This dataset contained 1200 strokes per genuine user. Their analysis indicated that the performance of the classifier does not stay constant but rather fluctuates when trained on the 1^{st} day and evaluated on the next few days. Due to the small number of genuine users, the results claimed in this work do not achieve statistical significance and likely not represent general characteristics of their dataset.

In contrast to this, our work uses a dataset that controlled the device size and user posture. The empirical techniques we propose are novel to this field and more rigorous. The data was collected for a larger sample size of 31 users over a longer period of 42 days and generated ~900 strokes per user. The conclusions of Xu et al. will be further contrasted against ours in Section 10.1.

With reference to reporting benchmark accuracy in touch-based authentication, researchers have used diverse experimental setups that vary significantly with respect to number of users, samples per user, device(s) used, etc. The range of controlled and confounding factors hampers empirical comparisons. Frank et al. achieved an EER of 2-3% immediately after the experimental session which dropped to 4% after a week. Serwadda et al. [13] provided a benchmark analysis for touch based user authentication and showed that a Logistic Regression based classifier offers the best results (10.5-17.8% EER depending on device position and user stroke direction). Syed et al. investigated the effect of device size and user posture and concluded that both significantly affect the classifier performance [7]. They reported mean EERs of 3.8%-8.81% depending on the scenario under consideration. Shen et. al [14] achieved False Acceptance Rate (FAR) and False Reject Rate (FRR) of 7.52% and 5.47%, respectively, when the user performed unconstrained tasks, while in application specific tasks the FAR and FRR reduced to 4.68% and 1.17%. They also determined that user's touch behavior exhibits less variability in application specific tasks as compared to unconstrained tasks.

As can be seen from the related work, the research in touch based continual authentication is in its infancy. The number of features varied from 10 to 53 across the different studies. One study does not disclose the types of features used [11]. Some studies leverage feature selection [10] [8] and others do not[11]. Similarly, feature normalization is performed only by one study [8]. Some studies use EER for benchmark comparison [10] [8] [13] while others only report error rates[11].

5 Dataset used and preprocessing

In this work, we used a part of the dataset collected by Syed et al. [7]. While the original dataset contains data on 3 devices and 3 postures (9 scenarios), we present the result of our approach on only one scenario.

The data was collected from 31 users using a custom app on Samsung Tab 10” device (Android 4.1 OS) held in portrait orientation. The data was collected from each user over a period of 4 weeks. The custom app was a photo matching game where the objective was to search through a list of images to find a randomly selected image that the app displayed as an inset. This activity required the user to scroll vertically and swipe horizontally through the image list to find the desired image. As the user interacted with the app, the data from the strokes executed by the users was collected in the background.

Syed et al. [7] state that no additional software was installed on the devices except for the data collection app. All selected devices were based on the Android operating system. Android was chosen for multiple reasons: According to a developer survey conducted in April-May 2013, Android is used by 71% of mobile-software developers [15]. Furthermore, Android has close to 80% market share in global smartphone shipments in 2013 [16]. Thus, any results we obtain on these devices would be applicable to the larger segment of the mobile device population. Furthermore, Google, Android’s developer, provides a customized Eclipse IDE for Android app development. This allows for rapid development, prototyping and testing of apps [17]. The motivation behind a custom app was to create a means to capture the natural horizontal and vertical strokes from the user. The app consisted of a photo matching game where the objective was to find a randomly displayed image from a list of images. Further details are explained by Syed et al. [7].

The structure of the dataset used in this work is shown in Fig. 1. The data for any user i consist of n_i strokes. Each stroke represents a feature vector containing 17 features. These features are explained in Table 1. Note that the dataset used in this work consists solely of horizontal strokes executed by the users. The results for vertical strokes are omitted due to space considerations.

5.1 Data Pruning

During the data collection period, each user was allowed to use the experiment’s app freely to attain a certain objective. Therefore the number of touch strokes entered by each user is different. Before proceeding, we pruned the dataset as shown in Fig. 5.1. If user i submitted the least number of strokes n_i, all other users’ data was reduced to this number by discarding the later strokes. This was performed to attain a uniform length dataset for all users. After pruning, the dataset contains $l = 912$ strokes per user.

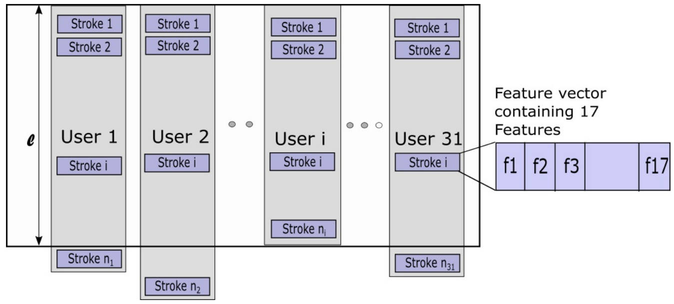


Figure 5.1: The dataset structure and the pruning method used. Since user i has the least number of strokes n_i, all the user data is pruned to this size by discarding all the later strokes.

5.2 Feature Normalization

As mentioned earlier, each stroke corresponds to a feature vector containing 17 features. Let l be the number of strokes per user in the dataset. Let f_i^j represent the i^{th} feature in j^{th} feature vector. Each feature f_i^j is normalized using a Standard Scaler algorithm. The normalized feature g_i^j is calculated as:

$$g_i^j = \frac{(f_i^j - \mu^j)}{\sigma_{f^j}},\ 1 \leq i \leq l,\ 1 \leq j \leq 17$$

where $\mu^j = (\frac{\sum_{i=1}^{L} f_i^j}{l})$; $\sigma_{f^j} = \frac{\left[\sum_{l=1}^{l}(f^j - \mu^j)\right]^2}{l}$

This causes the distribution of every feature in the entire dataset to have zero mean and unit variance. This is required to obtain valid results from certain machine learning algorithms such as Support Vector Machines, Linear Perceptron that are used in this work. However, please note that feature normalization using Standard Scaler algorithm was not applied during the experiment in Section 7. This is because the similarity measure used in this experiment is a distance based model which along with Standard Scaler algorithm is prone to outliers.

Table 1: List of Features and their description

Features	Description
StartX, StartY, StartPressure, StopX, StopY, StopPressure	Abscissa, ordinate and pressure at the location where the gesture began/ended
StrokeDuration	Duration of stroke (in µs)
Length_EE, Angle_EE	Distance and angle between beginning and end point (in pixels)
Length_Trj	Length of gesture's trajectory
Ratio_Trj2EE	This ratio between Length_EE and Length_Trj. This is a measure of deviation of the gesture from a straight line.
AverageVelocity	The average velocity of the gesture
InterStrokeTime	Delay between successive strokes
MidPress	Val Pressure at the midpoint of the gesture
Vel20, Vel50, Vel80	Average velocity after 20/50/80% of the stroke has been executed
Direction	Primary direction of the stroke (Horizontal/Vertical)

Part I
User Habituation in Continual Authentication

6 Section Organization

This part of work is organized as follows:

Sections 8 and 9 detail empirical techniques to detect habituation and quantify its effect on classifier performance. Based on these conclusions, we determine the most accurate user model updating strategy in Section 10. In Section 11 we quantify the benefits of a vote based classification approach to significantly enhance the accuracy of the user model. Section 12 concludes the paper with a summary and avenues for future work.

7 Measuring habituation via similarity measures

As mentioned before, we hypothesize that habituation affects the user's touch profile. In this experiment we use similarity measures to test our hypothesis that habituation manifests in two ways:

1. The intra-user stroke variance is greater in the early stages than at the end when the user becomes more familiar with the device and app.

2. The user profile itself changes over time.

Note that these two effects are independent of each other, i.e. one effect may be exhibited while the other is not. We used similarity measures as the first step since we detected habituation using similar measures in a previous work on keystroke dynamics [18]. Furthermore, this technique is fairly common in biometric systems [19, 20].

7.1 Experimental Setup

We quantified the first and second effect using similarity measures S^1 and S^2 respectively. Both are ordered tuples and use Mahalanobis distance to measure similarity and are calculated as illustrated in Fig. 7.2 and explained below:

1. Group 5 consecutive strokes beginning with the first stroke into blocks. Let these blocks be $B_1, \ldots, B_j, \ldots, B_{l/5}$

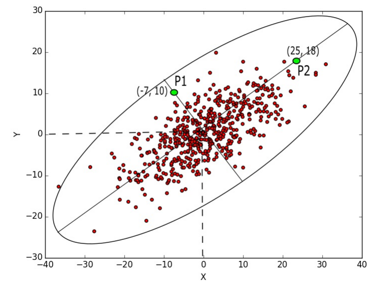


Figure 7.1: Visualizing Mahalanobis distance

2. Each stroke in the dataset is a p dimensional vector. For every Block B_j:
   (a) Calculate vector mean $\boldsymbol{M}_j$ and variance $\boldsymbol{V}_j$.
   (b) Calculate the j^{th} element of the S^1 tuple: $S_j^1 = d_m(\mathbf{V_j}, \mathbf{O})$ where $\boldsymbol{O}$ represent a Origin.
   (c) Calculate the j^{th} element of the S^2 tuple: $S_j^2 = d_m(\mathbf{M_j}, \mathbf{M_{j+1}})$.

When calculating similarity measure S^1, the origin represents zero variance. Therefore, the distance of variance $\boldsymbol{V}_i$ from origin $\boldsymbol{O}$ indicates the amount of variance in block B_j. Plotting this value for all groups depicts the intra-user stroke variation over time.

The similarity measure S^2 indicate the similarity between 2 consecutive blocks B_j and B_{j+1}. Plotting this value for all blocks depicts the change in user profile over time. If either of the two effects of habituation are present we expect the corresponding similarity measure to decrease.

The Mahalanobis distance d_m between two vectors $\boldsymbol{X}$ and $\boldsymbol{Y}$ is defined as:

$$d_m(\mathbf{X}, \mathbf{Y}) = \sqrt{(\mathbf{X} - \mathbf{Y})^t S^{-1} (\mathbf{X} - \mathbf{Y})}$$

where S is the covariance matrix.

Fig. 7.1 illustrates the concept of Mahalanobis distance. While point **P1** appears to be closer to the origin **O** than **P2** (based on Euclidean distance), the variance of the distribution is smaller on the Y-axis as compared to the X-axis. Due to this, the Mahalanobis distance $d_m(\mathbf{P1}, \mathbf{O})$ may be equal or greater than $d_m(\mathbf{P2}, \mathbf{O})$. Euclidean distance assumes that all components of the vector contribute equally towards the distance. In contrast the non-unitary covariance matrix S in Mahalanobis distance neutralizes the effect of differences in range and variance amongst features by compensating features with low variability and range.

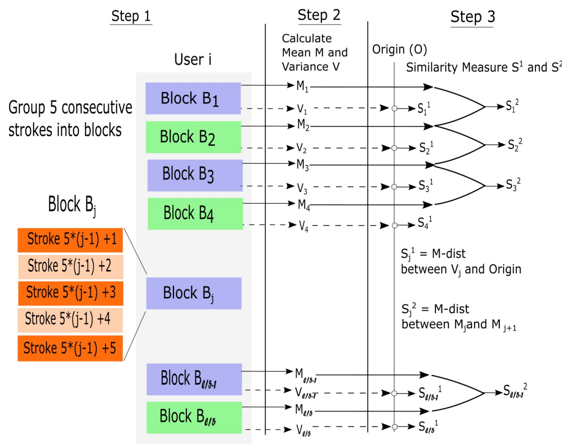
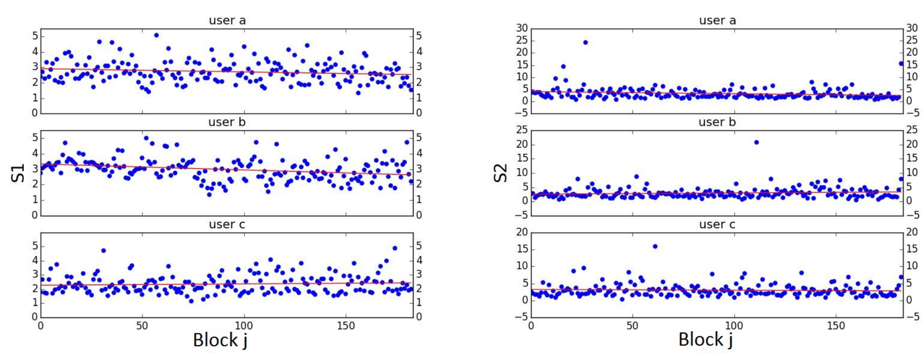


Figure 7.2: Calculating the similarity measures S^1 and S^2 for any user i. Each measure is an ordered tuple, S^1 of size *l/5* and S^2 of size *l/5-1*.



(a) S^1 - Intra-user change in variance over time

(b) S^2- Intra-user change in user profile over time

Figure 7.3: Variation in similarity measures S^1 and S^2 over time for 3 representative users.

7.2 Results and Conclusions

Fig. 7.3 shows the variation of S^1 and S^2 across the entire dataset for 3 representative users. The results indicate that:

- A sustained decrease in intra-user stroke variation is faintly present for some users, but other users exhibit no clear pattern. Out of 31 users, S^1 weakly decreased for 10 users with time. Thus, regardless of how habituated the user is, the intra-user variance remains constant over time.
- Similary, no significant pattern was discernible for changes in user profile. S^2 weakly decreased for 16 users.
- For the remainder users, no clear trend exists for both S^1 and S^2.

We concluded that, when using simple distance measures, a group of users do not collectively show change in either intra-user stroke variance or in user profile over time.

8 Effect of intra-user variance on classifier performance

As mentioned before, we hypothesize that habituation manifests in two ways:

1. The intra-user stroke variance is greater in the early stages than at the end when the user becomes more familiar with the device and app.
2. The user profile itself changes over time.

Note that these two effects are independent of each other, i.e. one effect may be exhibited while the other is not.

The experiments in this section (Section 8) quantify the effects of the first manifestation of habituation: decrease in intra-user variance in the user's touch profile over time. In Section 9, we quantify the effects of the second manifestation of habituation: change in user profile over time. Both sections use classifier performance as a metric for measuring these effects.

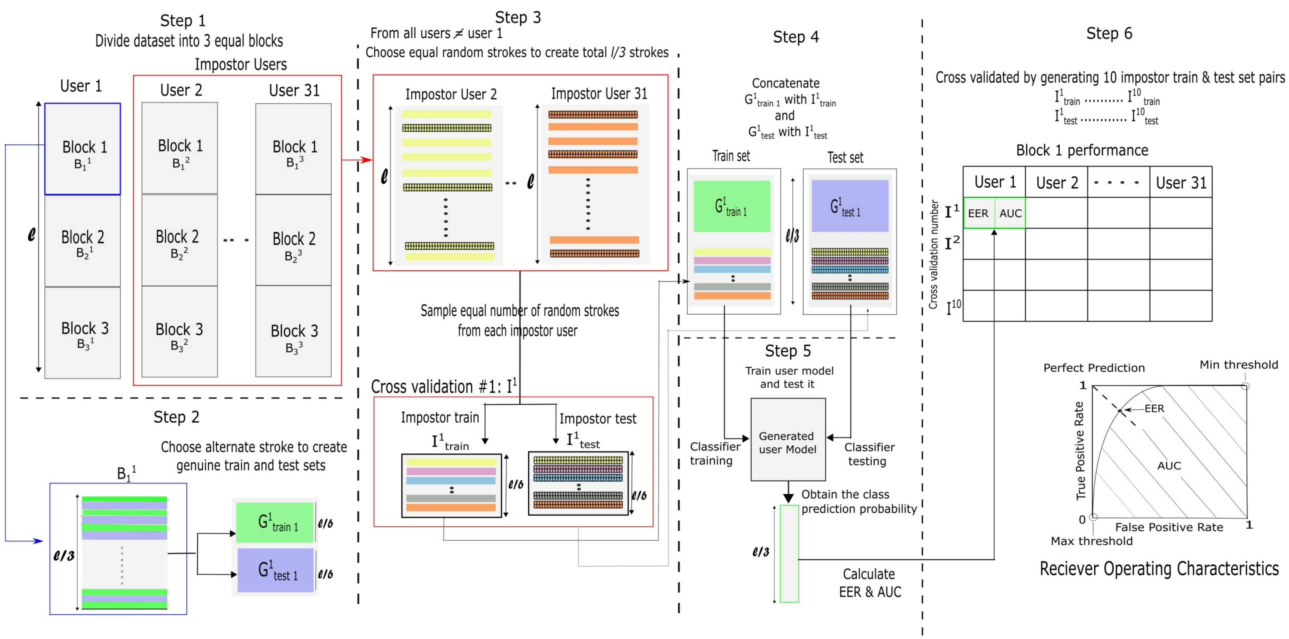


Figure 8.1: Experimental setup for evaluating user model performance in Section 8.1. For simplicity, we show the process specifically when User 1, Block 1 is used to sample the genuine train and test sets.

8.1 Experimental Setup

To detect the effects of intra-user variance on classifier performance, we resorted to developing a number of machine learning-based user models using data from distinct parts of the dataset. If intra-user variance is greater in the earlier part of the dataset, the user model trained and tested using later data will be more accurate than the one trained and tested with earlier data. The experimental procedure is illustrated in Fig. 8.1.

1. Divide the dataset in 3 equal blocks (Step 1 in Fig. 8.1). Each block is representative of the user's profile at a specific point in time. The objective of this experiment is to compare the performance of user models developed on each of the three blocks. Note that a user model is trained and tested on data from the same block.

2. For each block B^i_j, $1 \leq i \leq 31$ and $1 \leq j \leq 3$ where i is the user and j is the block number:

   (a) Sample genuine data - Sample alternate strokes to create genuine train set $G^i{}_{trainj}$ and test set $G^i{}_{testj}$ (Step 2a in Fig. 8.1). Sampling alternate strokes evenly samples the entire block.

   (b) Sample impostor data - We sampled equal number of genuine and impostor data for both train and test sets. To sample n impostor strokes randomly sample *n/30* strokes from each impostor user to create impostor train set $I^k{}_{train}$ and test set $I^k{}_{test}$. Random sampling of impostor data generates the broadest range of impostor profiles.

   (c) Append $G^i{}_{trainj}$ and $G^i{}_{testj}$ with $I^k{}_{train,}$ and $I^k{}_{test}$ respectively to create the complete train and test sets.

   (d) As this is a supervised learning process, the complete train set is used to train the classifier and generate the user model which is then evaluated on the corresponding test set. Calculate the Equal Error Rate of the evaluated test set.

   (e) Cross-validate the results by repeating Steps 2b-2d 10 times. This generates 10 different impostor train and test sets ($I^k{}_{train},\ I^k{}_{test}$) $1 \leq k \leq 10$. To determine the performance of a user model built using Block B^i_j, the mean performance of the user model on these 10 test sets is calculated.

   (f) Steps 2d and 2e are repeated using the following classification algorithms: Random Forest, Classification and Regression Tree (CART), Naive Bayes, Support Vector Machine (linear kernel), k-Nearest Neighbor (*k=10*) and Linear Perceptron. These have been used previously in other benchmark studies [13, 21].

All pre-processing and classification algorithms were implemented using Python's scikit-learn library [22]. We used Equal Error Rate (EER) from the Receiver Operating Characteristics (ROC) graph as our performance metric. The ROC is created by plotting True Positive Rate (TPR) against False Positive Rate (FPR) at different threshold points as shown in Fig. 8.1. True Positive Rate is the proportion of genuine samples classified correctly and False Positive Rate is the ratio of impostor samples being classified incorrectly.

The Equal Error Rate is the point on the curve where sum of TPR and FPR is 1. The lower the EER, the better the classifier. Ideally, EER equals 0 indicating perfect classification.

Results and Conclusions:

Table 2 shows the results of this experiment. We reiterate that in this experiment the genuine train data and genuine test data were both sampled from the same block, i.e. the same point in time.

- Note that Random Forest consistently provides the best results across the dataset.
- The three linear classifiers (Linear kernel SVM, Naive Bayes and Linear Perceptron) performed the worst.
- The results indicate that the classifier accuracy remains constant when user models are trained and tested using genuine data from any one block.
- We thus infer that the intra-user variance in touch profile has no perceivable effect on classifier performance.

Based on the lack of change in classifier performance over time (when genuine train and test data are sampled from the same block), we conclude that the user profile exhibits either of two characteristics:

1. It remains unchanged over time OR
2. It does change but the classifier is robust enough to handle the changes and maintain performance.

We show that inference 2 is correct in Section 9.

8.2 Addressing potential confounding factors

It could be claimed that the experimental setup in the above experiment does not provide strong evidence for user habituation because of two confounding factors:

1. Insufficient training data samples available to the classifier which may hamper user model development.

Table 2: A benchmark comparison of user models using various classification algorithms on different parts of the dataset.

Classifier	**Mean EER% (Std Dev)**		
	B1	B2	B3
Random Forest	10.4 (3.9)	10.5 (4.2)	10.4 (3.1)
CART	15.5 (5.1)	16.5 (5.9)	15.9 (5.0)
k-NN (k=10)	16.3 (5.4)	16.0 (5.5)	15.7 (5.0)
SVM (linear)	18.0 (8.4)	18.3 (7.6)	17.2(6.4)
Naïve Bayes	20.3 (7.1)	21.7 (7.9)	19.9 (7.2)
Linear Perceptron	34.5 (10.3)	34.2 (10.6)	33.9 (9.4)

2. The dataset was not divided into sufficiently small blocks to detect the habituation trend.

To address the first concern we repeated the experiment by dividing the dataset into two equal blocks as opposed to three blocks previously. This ensures that each block will have more samples.

To address the second concern we again repeated the experiment by dividing the dataset into six equal blocks. This ensures that each block will contain the data from a shorter temporal range compared to the original experiment.

This answers our Research Question RQ3 listed in Section 2: Since Random Forest algorithm had the highest accuracy over our entire dataset, the results of the experiment are shown only for this classification algorithm.

Results and Conclusions:

Tables 3a and 3b show the classifier performance when the data is split into two and six blocks respectively.

Neither experiment resulted in a user model with appreciable performance difference compared to the original experiment. This indicates that:

- We used a sufficient sample size for train and test sets to arrive at our conclusions.
- Using fewer samples to generate a user model does not help in detecting changes in intra-user variance.

9 Effect of change in user profile on classifier performance

We now determine which of the two inferences in Section 8.1 caused the classifier performance to remain unchanged over time:

Table 3: Classifier performance using Random Forest algorithm when genuine data is divided in (a) 2 blocks and (b) 6 blocks

(a) 2 blocks split

Block #	**Mean (StdDev)**
1	11.5 (3.8)
2	11.5 (3.6)

(b) 6 blocks split

Block #	**Mean (StdDev)**
1	10.0 (5.2)
2	11.7 (4.8)
3	10.0 (4.7)
4	11.1 (4.9)
5	9.0 (3.8)
6	10.3 (3.9)

1. Lack of change in the user profile.

2. Classifier robustness to changes in the user profile.

To do so we used the same experimental setup as Section 8.1 except that the genuine train and test sampling scheme (Step 2a) was changed:

This modified scheme is illustrated in Fig. 9.1. Unlike Section 8.1 where genuine train and test sets were created from the same block, this experiment uses a complete Block *i* as the genuine train set. It is then tested ***separately*** on each of the later blocks (Block *i+1, ..., 6*). We did not test the user model on any block prior to Block *i* because it is not relevant to the experiment's objective. The genuine set is divided into six equal blocks to keep the train/test set size the same as previous experiments. The impostor data is sampled as in the prior experiments.

As shown in Fig. 9.1, in iteration 1 a user model using Block 1 as train set is generated. It is then tested on Blocks 2-6 as 5 separate test sets. In iteration 2, Block 2 functions as the train set and Block 3-6 as the test sets and so on.The EER is calculated for each test set.

Results and Conclusions

Table 4 reports the mean EER values for each iteration of this experiment. The results show that:

1. In every iteration the user model trained on Block *i* offers the best performance on Block *i+1*. This indicates that, any block is most similar to the block immediately after it and the similarity decrease as the distance between them increases.

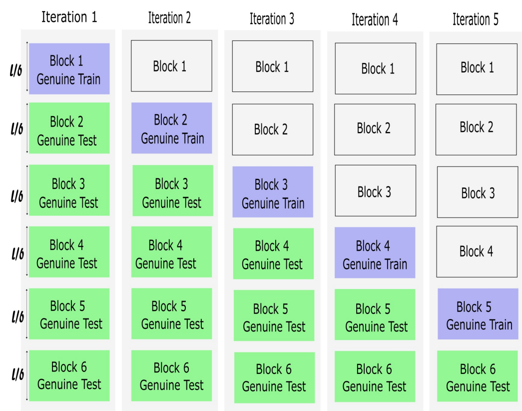


Figure 9.1: Experimental setup for Section 9 to determine the degradation in performance of a user model over time. White blocks correspond to unused data in every iteration.

Table 4: Degradation in performance as the user model becomes older. The user model is tested only on blocks later than the train block.

	Mean EER(%)				
Test on Block:	*Train on Block:*				
	1	*2*	*3*	*4*	*5*
2	16.35	-	-	-	-
3	22.57	16.22	-	-	-
4	28.95	20.50	18.77	-	-
5	27.66	22.46	21.27	15.48	-
6	31.60	26.32	22.34	23.19	17.20

2. The performance degradation over time is significant. The EER increases from ~16% to ~32% when the distance between the train and test sets is 600 strokes (Each block contains ~150 strokes).

3. This clearly shows that habituation causes the user profile to change significantly over time, i.e. the second inference is the correct one.

4. The user model performance always changes irrevocably. We infer this from the fact that the classifier performance always decreases over time and then never increases later on.

Based on the above conclusions, we now answer Research Question RQ1 listed in Section 2: it is clear that regularly retraining the classifier using the latest data is vital to maintaining the accuracy of the touch-based authentication system. Since the user profile changes irrevocably, only the latest data must be used for updating the user model.

10 What train set size works best?

Section 9 clearly shows that the user profile becomes outdated over time and degrades the classifier performance. Thus it is imperative to periodically update the user model with the latest genuine user data. In this section we determine the optimal amount of prior data that must be used to update the user model.

To do so we used the same experimental setup as Section 8.1. However, the genuine train and test set sampling strategy was modified as shown in Fig. 10.1:

1. The genuine data was divided into 12 equal blocks of size *l/12*. This provides the flexibility to increase the train set in relatively smaller steps.

2. For a given Block i as the test set, in iteration j where $6 \leq i \leq 12$, $1 \leq j \leq 5$:

   (a) Generate a user model using Blocks *i-1 to i-j* ***together*** as a train set as shown in Fig. 10.1.

   (b) Calculate EER for this user model.

Thus in each iteration the genuine train set size is enlarged by a block while the genuine test set remains the same. Step 2 is performed using Blocks 6-12 as test sets separately. It should be noted that the classifier train and test set always contained equal number of genuine and impostor strokes. The impostor data samples are sampled using the same sampling scheme as before (described in Section 8.1).

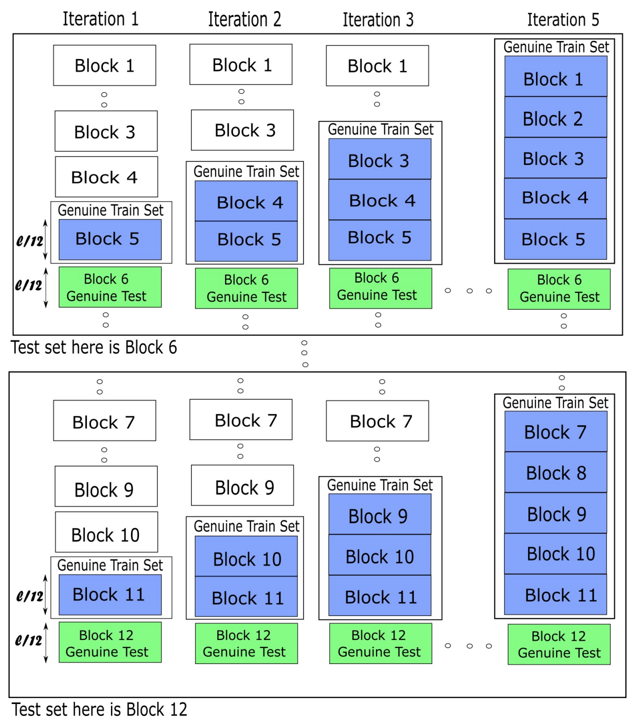


Figure 10.1: Experimental setup to determine the optimum train set size. The dataset is divided into 12 blocks. Initially, Block 6 is solely used as the test set followed by Blocks 7-12, each used separately as a test set. For a given test set, depending on the iteration number x, the previous x blocks are used to train and generate the user model.

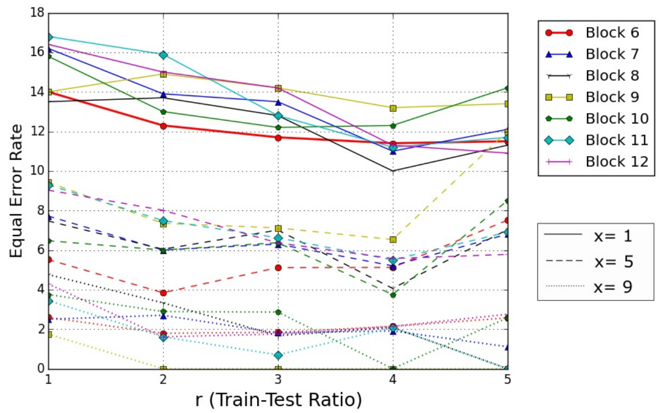


Figure 10.2: Median EER variation as r is increased. The solid lines indicate the results of Section 10 (x=1) while dashed (x=5) and dotted lines indicate the results of Section 11.

Say r denotes the train:test set size ratio such that $r = N$ implies a train:test size ratio of *N:1*. The optimum train set size can be determined by comparing the performance of the user models as r is increased. The upper limit of r is set to 5. This is because the dataset is divided in 12 equal blocks. A larger value of r would have meant that less than half of the dataset is available for testing purposes.

10.1 Results and Conclusions

The solid lines in Fig. 10.2 show the results for this experiment. Each solid line depicts the variation in Equal Error Rate as r increases.

Note that $r = 4$ gives best performance. Thus for this dataset, the best strategy is to use ~300 prior strokes for generating the user model when the test set contains ~75 strokes (each block contain $l/12$ strokes where $l = 912$). Using any further prior data has a detrimental effect on classifier accuracy.

These conclusions contradict those of Xu et. al. [12] who determined that using all prior data provides the best performance. However, we believe our results are empirically stronger and more correct due to the following reasons:

1. Xu et al. performed their analysis on data from only 3 genuine users. Our study uses 31 genuine users. This provides conclusions that can be more strongly extrapolated to the general population.

2. The impostor data in their study was collected by having 29 others users submit data in a single session. This yielded 200 strokes per impostor. In contrast, our dataset contains equal

r	Group size		
	x=1	*x=5*	*x=9*
1	16.81 (15.24) 9.1	9.27 (7.84) 8.1	4.67 (3.27) 7.23
2	15.68 (14.1) 8.4	7.60 (6.39) 7.23	3.85 (1.84) 5.15
3	14.77 (13.05) 7.9	7.23 (6.41) 6.39	3.98 (1.51) 5.71
4	11.48 (11.48) 9.0	6.83 (5.10) 7.19	3.68 (1.47) 5.61
5	12.15 (12.15) 8.2	8.91 (7.79) 6.9	4.12 (1.28) 6.42

Table 5: Classifier performance as r is increased when using non vote-based (x=1) and vote-based (x=5, 9) stroke reclassification. The results are reported as: "Mean (Median) Std deviation"

data (912 strokes each) from genuine and impostor users, both submitting data over 30 days.

3. While their dataset contains 30% more strokes, we argue that it is better to have more genuine users and a larger, more diverse impostor pool to obtain valid conclusions about habituation.

Based on the above conclusions, we now answer Research Question RQ2 listed in Section 2. Note that the optimal value of r may change if the test set size is varied. Furthermore, these conclusions are valid for the device and application used in this dataset (described in Section 5). As mentioned before, we used a 10" tablet running a custom image matching app. The conclusions from our described empirical techniques may be different for a different combination of software and hardware.

11 Quantifying the benefits of vote based reclassification and a benchmark reporting framework

11.1 Experimental Setup

In the previous experiments, the class predictions for every stroke generated by the classifier was used as-is to generate the EER. However, it is common practice in literature to take a majority vote on a group of strokes to determine their class [8, 13]. This is because a group of strokes is more reflective of the user's profile than the constituent strokes in isolation. Thus, in vote based reclassification, each stroke is reclassified by taking the predicted class of neighboring strokes into consideration. The benefits of this practice have not been quantified before and are presented in this section.

The experimental setup is identical to Section 10 except that stroke reclassification was introduced. This method is described below and illustrated in Fig. 11.1. Note that the stroke

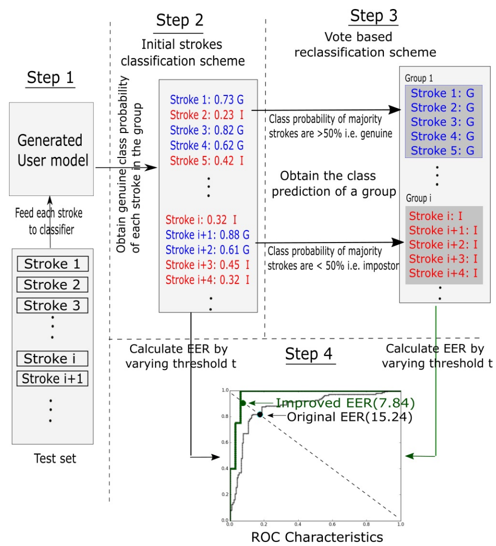


Figure 11.1: Evaluating the user model using traditional and proposed method where threshold t is set to 0.5.

reclassification is performed in Step 3b:

- **Step 1**: Each stroke in the test set is supplied to the classifier/user model.
- **Step 2**: The genuine class probability p of each stroke is determined.
- **Step 3**: The threshold t is varied in the range [0,1] in steps of 0.01
  - **a**. If probability $p > t$, the stroke is labeled as genuine, else as an impostor.
  - **b**. Group x consecutive strokes together. If $p > t$ for the majority of the strokes in a group, transform the predicted class of all strokes in that group to genuine, else to impostor.
  - **c**. Calculate FPR and TPR metrics.
- **Step 4**: Calculate EER

11.2 Results and Conclusions

Table 5 report the mean, median and standard deviation of performance for x = 1, 5, 9 as r is varied. $x = 1$ corresponds to when the reclassification is not performed. Each mean and median value in the table is based on the performance on the six test blocks. The individual performance of every test block is detailed in Fig. 10.2. The solid, dashed and dotted lines indicate where group size x is 1, 5 and 9 respectively. Each line depicts the variation in Equal Error Rate as r increases. The results shows that:

- Using a group of strokes and our optimum value of $r = 4$, we achieve a mean and median EER of 3.68% and 1.47%. This performance meets or exceeds that reported in related literature [13, 7]. Moreover, we report this benchmark using training and testing data that is from separate points in time. This is uncommon in related literature.
- We reiterate that these results have been attained using only the horizontal strokes from the dataset. We are optimistic that fusing the results with a vertical stroke-based classifier will further increase performance.
- Using a voting scheme with a group of 5 strokes enhances the median performance by 36-47%.
- Using a group of 10 strokes further increases the median performance by 78-86%.
- The voting scheme also reduces the standard deviation in EER. When taking with the fact that the median EER is always lower than the mean EER, it is clear that this strategy works well for the majority of users and a few users may be outliers who are difficult to classify.
- The accuracy of $x = 5, 9$ based method is more consistent over different values of r as compared to when $x = 1$ while showing a steady increase in performance up to $r = 4$.

The above results provide the answer to Research Question RQ4. We conclude that predicting the user's identity using individual stokes is an inefficient and inaccurate strategy. However, it should be noted that, increasing the number of strokes in a group will increase the time required to authenticate. Finally, in reference to Research Question RQ5, we attained a benchmark equal error rate of 3.68% on our dataset using a realistic train-test setup and vote-based reclassification.

12 Summary and Future work

In this part, we explored the presence of habituation in a touch based authentication dataset and its effects on the classifier performance. Through a series of empirical experiments, we concluded that the change in intra-user stroke variance over time has no perceivable effect on classifier accuracy. However, the user profile does change and, furthermore, changes irrevocably over time. This significantly degrades classifier accuracy and shows that the user model must be regularly updated using only the latest data to maintain classifier performance. We proceeded to evaluate the amount of data that leads to the best classifier performance. Our results indicate that, on our dataset, 300 strokes is the optimum number of strokes needed to develop the most accurate user model. Such a model suffers negligible performance degradation when used to test up to 75 strokes that temporally follow the training data. We then quantified the benefits of vote-based stroke reclassification. We show that this post-processing boosts the classifier performance by as much as 86% in comparison to non vote based classification. It also leads to more consistent classifier accuracy across all the users. We demonstrated an Equal Error Rate of 3.68% on our dataset using these techniques.

In the future, we plan to extend the study of habituation to other devices types and user postures. We will analyze the relationship between the classifier performance, group size and time to authenticate.

Part II

A cluster analysis based fusion algorithm to improve classification performance in touch based continual authentication system

13 Overview of Proposed System

This part broadly consist of the following major parts in chronological order which are explained in greater details in later sections.

- Part 1: Extract train and test datasets - The mechanism to sample train and test datasets is explained in step 1-3 of Fig. 16.1 and in Section 8.1. Train and test dataset are used for developing a user models and evaluate its performances respectively.
- Part 2: Determine the best similarity measure for clustering algorithm - The similarity measure between any data samples forms the core idea of any clustering algorithm. Better the similarity measure, better the performance of clustering algorithm. Different similarity measures are explained in Section 14 and Section 15.
- Part 3: Evaluate individual performances of supervised and non-supervised algorithms - Implement traditional supervised Random Forest classification algorithm and unsupervised clustering algorithms individually on train set to obtain their respective predictions on the test dataset. This is explained in step 5 of Fig. 16.1 and in Section 8.1. This is the first ever implementation of non-supervised clustering algorithm in continual authentication.
- Part 4: Fusion of predictions from different sources - The novelty of this paper lies within fusion of probabilistic predictions of traditional supervised random forest algorithm and clustering algorithm to improve the performance. This is explained in detail in step 6 of Fig 16.1 and in Section 17.

14 Clustering Theory

Clustering is an important data mining technique, that is used to divide data samples into different groups or classes, by minimizing intra-group similarity and maximizing inter-group similarity.

Cluster analysis is based on the principle that objects of the same class have greater similarity to each other and less similarity to objects of different class. One of the contribution in this section is the implementation of first ever non-supervised clustering algorithm in continual authentication on mobile devices. Under the condition of no apriori knowledge of the system, clustering methods are supposed to classify the data according to the type or class associations of that dataset. Formally, the clustering structure is represented as a set of subsets C = C_1, . . . , C_k of S, such that: S = $U_{i=1}^{k} = C_i$ and $C_i \bigcap C_j = \varnothing$ for i != j. Consequently, any instance in S belongs to exactly one and only one subset.

Thus clustering isn't a special algorithm by itself but a general task to group the data samples. There are broadly two ways to find the cohesion similarity between the data samples within the clusters and between the clusters:

- Distance Measure
- Similarity Measure

15 Cohesion similarity measures.

15.1 Distance Measure

Many clustering methods use distance measures to determine the similarity or dissimilarity between any pair of objects. It is useful to denote the distance between two instances X_i and X_j as: $d(X_i, X_j)$. A valid distance measure should be symmetric and obtains its minimum value (usually zero) in case of identical vectors. For our dataset we have used Mahalanobis distance measurement to find the similarity between two objects/samples[23]. We have explained Mahalanobis distance in detail in Section 7. However, we briefly discuss it here again. The Mahalanobis distance d_m between two vectors $\boldsymbol{X}$ and $\boldsymbol{Y}$ is defined as:

$$d_m(\mathbf{X}, \mathbf{Y}) = \sqrt{(\mathbf{X} - \mathbf{Y})^t S^{-1} (\mathbf{X} - \mathbf{Y})}$$

where S is the covariance matrix.

The non unitary covariance matrix S in Mahalanobis distance neutralizes the effect of differences in range and variance amongst features by compensating features with low variability and range. Euclidean distance assumes that all components of the vector contribute equally towards the distance. In contrast the non-unitary covariance matrix S in Mahalanobis distance neutralizes the effect of differences in range and variance amongst features by compensating features with low variability and range as shown in Fig. 7.1.

15.2 Similarity Measure

An alternative approach to distance similarity is to find the similarity function $S(X_i, X_j)$ that compares the two vectors X_i and X_j, provided the similarity function is symmetric i.e. $S(X_i, X_j)$ = $S(X_j, X_i)$. The higher the similarity, the more chances of belonging to the same cluster. The two most commonly used similarity functions are:

1. Cosine Similarity: The cosine similarity between vector X_i and X_j is formulated as follows [24]:

$$S(\mathbf{X_i}, \mathbf{X_j}) = \frac{X_i^T * X_j}{\| X_i \| \| X_j \|}$$

   where X^T is the transpose of vector X and $\| X \|$is the magnitude of vector X . In cosine similarity the angle between the two vectors are used as a similarity function. The value of a cosine similarity ranges from -1 to 1 inclusive i.e. [-1, 1] where -1 indicate exactly opposite vectors or most dissimilar vectors and 1 indicate the exactly same or exactly similar vectors. The values between -1 and 1 indicate the extent of similarity or dissimilarity between the two objects with 0 indicating orthogonal vectors.

2. Extended Jaccard Measure: The extended jaccard similarity between vector X_i and X_j is formulated as follows [25]:

$$S(\mathbf{X_i}, \mathbf{X_j}) = \frac{X_i^T * X_j}{\| X_i \|^2 + \| X_j \|^2 - X_i^T * X_j}$$

   where X^T is the transpose of vector X and $\| X \|$is the magnitude of vector X . The extended jaccard measure is very similar to the principal of jaccard coefficient. It measures the similarity of two sets by comparing the size of the overlap against the size of the two sets. If the two sets have only binary attributes then it reduces to the Jaccard Coefficient. For example the jaccard coefficient between two vectors $X_i(0, 1, 1, 0)$ and $X_j(1, 0, 1, 1)$ is $1/4$. This is because the intersection cardinality between the two vectors is 1 and union cardinality is 4. However, in our dataset we have 17 continuous attributes which we have described in Section 5. Therefore we use the extended jaccard coefficient.

16 Procedure to generate the optimal set of clusters for touch based continual authentication system

We used a hierarchical clustering mechanism to build the clusters using our dataset. This is a iterative process carried out as follows:

1. Assign each items to its own cluster. In our case each item is a separate data stroke. Given N data strokes each stroke is considered its own cluster initially.

2. Use one of the similarity measures described in Section 14, find the similarity between each pair of clusters. This results in N_{C_2}computations.

3. Find the most closest pair of clusters in the above computations and merge them into single cluster by computing their mean.

4. Recompute similarities between the new cluster and remaining old clusters.

5. Repeat step 2 to step 4 until we get desired number of clusters.

It should be noted that clustering is very sensitive to to the dimensionality of the dataset. In our dataset each stroke is a 17 feature vector. Therefore we reduced the dimensionality using Principal Component Analysis (PCA). It has been used widely for dimensionality reduction of large multidimensional datasets using eigenvalues and eigenvectors. The use of PCA allows the number of variables in a multivariate data set to be reduced, while retaining the present variation as much as possible in the data set [26]. It transforms the original set of observation into a new set of variables called principal components. Note that we do not need to keep all the principal components. We can reduce the dimensionality by only keeping top n number of principal components. This is done by following simple transformation function

$$T_n = XW_n$$

where T_n is the new set of variables having n principal component, X is the original set of observations having N features and W_n is the $N * n$ matrix whose columns are the eigenvectors of $X^T X$.

To build the most accurate cluster space representation we varied two parameters:

1. Number of clusters *c.*

2. A boolean parameter denoted by *w,* indicating whether the data is whiten or not. A whitening transformation is a linear transformation that transforms the vectors of known co-variance matrix into a new set of vectors that has unit variance and identity co-variance matrix.

The criteria used to achieve the best combination of parameters is the averaged SSE.The averaged sum of squared error is an internal quality check for clusters and it does not use any other information besides data. It usually measures the intra-cluster homogeneity and the the inter-cluster separability. Averaged SSE can be formulated as follows:

$$Avg - SSE = \frac{1}{N} \sum_{k=1}^{k} \sum_{X_i \in C_k} \parallel X_i - U_k \parallel$$

where N is the total number of instances in a dataset, k is the total number of clusters, C_k is the set of instances in cluster k and U_k is the mean vector of cluster k which can be calculated as

$$U_k = \frac{1}{N_k} \sum_{X_i \in C_k} X_i$$

The SSE should be minimized between the clusters to determine the optimal combination of PCA components, number of clusters and if data should be whitened. For any similarity measure we used we find the combination of the 3 parameters that give the least SSE averaged over all the user. Thereafter the best parameter combinations and similarity measure is used for all the users.

16.1 Experimental Setup to determine the best similarity measure & parameter combination

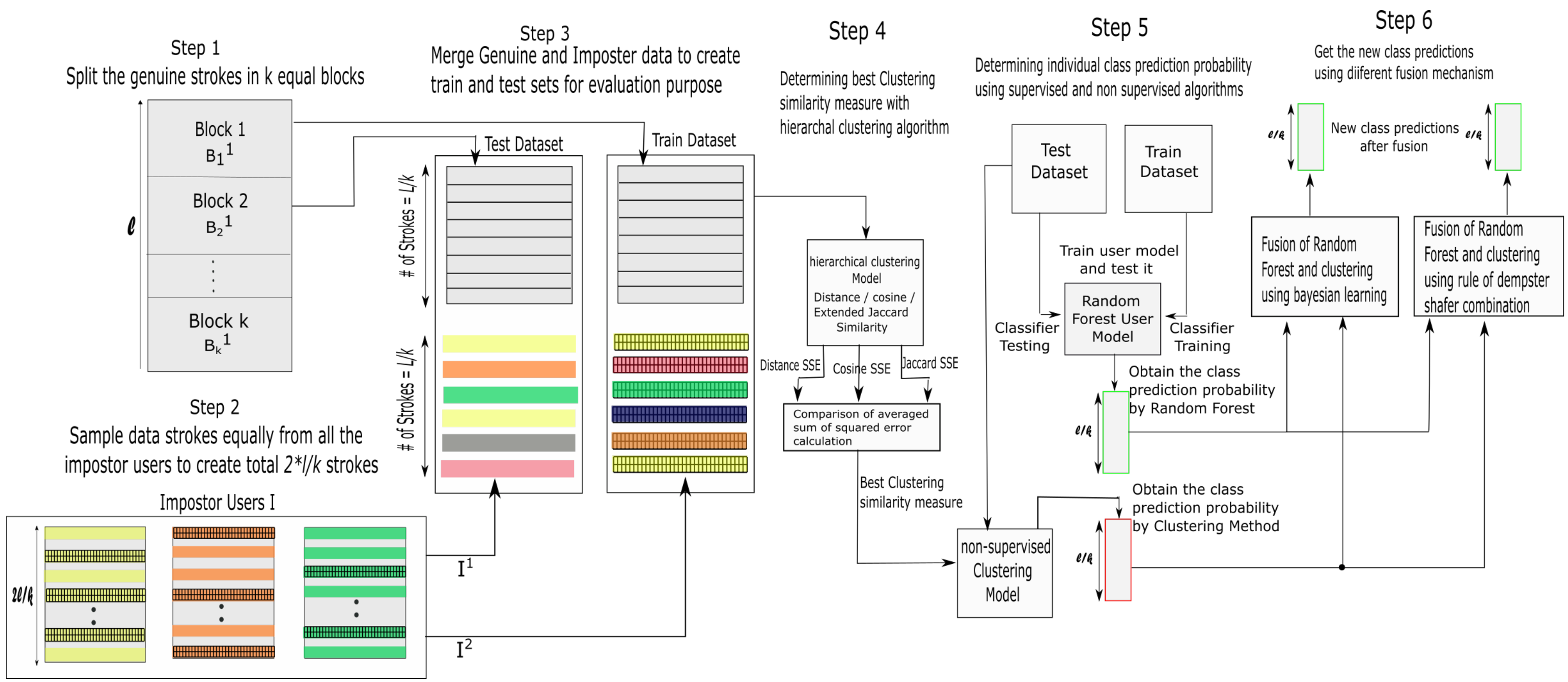


Figure 16.1: Experimental setup for evaluating different clustering similarity measure as well as different fusion mechanism. For simplicity, we show the process specifically for User 1 with Block 1 as genuine train dataset and Block 2 as genuine test dataset.

Steps 1-4 here are used to determine the best possible clustering parameters and steps 5-6 are used to test performances.

we resorted to developing a number of similarity measure-based user models using data from distinct parts of the dataset. The Experimental procedure is illustrated in Fig. 16.1 on Block 1 and Block 2.

1. **Step 1:** We divide the dataset of size l into k equal blocks. Each block is a representative of the user's profile at a specific point in time (Step 1 of Fig 16.1). For each block B_j^i, $1 \leq i \leq 31$ and $1 \leq j \leq k$ where i is the user and j is the block number: We split our dataset in 6 equal blocks. This is because:

   (a) The dataset used for evaluation is same as the dataset presented by Palaskar et. al. [27] and it is empirically confirmed that splitting the dataset in 6 blocks does not decay the performance.

   (b) This provides the flexibility to form more number of consecutive pairs of blocks for the evaluation and thus strengthening the results.

2. **Step 2**: Sample impostor data - We sampled equal number of genuine and impostor data for both train and test sets. To sample n impostor strokes, randomly sample $n/30$ strokes from each impostor user. In our experiment each genuine block is of size $l/6$ and we have used 2 genuine blocks each for train and test purposes. Therefore we sampled a total $2l/6$ impostor data strokes from all other users (Step 2 of Fig 16.1). Random sampling of impostor data generates the broadest range of impostor profiles.

3. **Step 3:** Append B_1^1 and B_2^1 with I^1 and I^2 respectively to create the complete train and test sets (Step 3 in Fig. 16.1). Fig. 16.1 explains the procedure for Block 1 and Block 2. However, note that the same procedure repeats for every consecutive pair of blocks.

4. **Step 4**: We build the clusters on train dataset individually for each user using three different similarity measures as described in Section 14. Our initial hypothesis for building the clusters is to separate genuine and impostor data strokes without any prior knowledge of their classes. Ideally we should get genuine strokes in one cluster and all impostor strokes in another cluster. To get the maximum accuracy clusters we vary parameters c and w for each clustering similarity explained in Section 6. Note that these set of combinations and similarity measure can be different for each user.

5. **Step 5:** Obtain the class prediction probability of each stroke in test dataset using Supervised (Random Forest algorithm) and Unsupervised (Clustering algorithm) individually as shown in step 5 of Fig16.1. The complete train set is used to train the classifier and generate the user model which is then evaluated on the corresponding test set.

Table 6: Comparison of similarity measures used to find clusters while keeping PCA = 6. Note thst only the result of 6 representative users are shown here.

User #	Distance Measure		Cosine Similarity		Extended Jaccard Measure	
	Best Combination	Avg SSE	Best Combination	Avg SSE	Best Combination	Avg SSE
1	c = *3*, w= *True*	9.18	c = *2*, w = *True*	2.68	c = *2*, w = *True*	5.67
2	c = *2* w = *True*	11.43	c = *3*, w = *True*	3.45	c = *3*, w = *False*	6.82
3	c = *2*, w = *True*	8.91	c = *2*, w = *False*	2.15	c = *2*, w = *False*	5.35
4	c = *3*, w = *True*	10.78	c = *3*, w = *True*	2.31	c = *3*, w = *True*	6.37
5	c = *3*, w = *True*	9.72	c = *3*, w = *True*	2.91	c = *3*, w = *False*	5.82
6	c = *2*, w =*True*	10.15	c = *2*, w = *False*	3.17	c = *2*, w = *True*	6.57

6. **Step 6**: Fusion of two results - Each set of predicted class probabilities from supervised and non supervised algorithms is then fed to two fusion mechanisms.

   (a) Fusion using Bayesian Learning, in which the two set of probabilities are fused using Bayes decision rule to form a new class prediction probabilities.

   (b) Fusion using simple rule of combination, in which at any moment of decision we use either of the prediction scheduler (random forest or clustering) depending on the constraints.

We explain the fusion mechanism in greater details in Section 17.

16.2 Results and Discussion

The results indicate the PCA with 6 principal components gives the optimum results on our dataset. Therefore we used 6 PCA transformed features to build the clusters. We compare the three similarity measures described in Section 14 using averaged SSE. However, note that the comparison of distance measure against the similarity measure is meaningless because of the differences in their nature. Therefore we must have to first revert the similarity measure statistics into a distance measure statistics using trigonometric cosine theorem properties, which can be stated as:

$$distances = \sqrt{2(1 - similarity)}$$

We find the corresponding distance measure for a given similarity measure and calculate averaged SSE for each clustering mechanism. Note that we calculate the averaged SSE for each genuine block as shown in Step 1 of Fig 16.1. However, for simplicity we have presented the average of averaged SSE over each block under different clustering mechanisms in Table 6.

Table 6 Shows the best parameter combinations for each clustering similarity and their averaged SSE on 6 strongly representative users from our 31 user dataset.

- Note that Cosine similarity measure gives us the optimum results with lowest average SSE.
- Each user has its own unique set of combination of parameters. This implies that clustering parameters of user x may not work for user y and vice versa.
- From our results we claim that the best clustering mechanism to use is cosine similarity. We have thus answered our Research Question 1 mentioned in Section 2.2, i.e which non-supervised clustering algorithm is most accurate and consistent over all the users in our dataset.

17 Fusing non-supervised and supervised algorithm

17.1 Fusion using Bayesian Learning

Bayesian learning is strongly based on the Bayes decision theory. Bayesian decision theory is a fundamental statistical approach to the problem of pattern classification. It is considered the ideal case in which the probability structure underlying the categories is known perfectly. Let us reconsider our problem of classifying two types data strokes: Genuine and Impostor. Suppose that our system finds it hard to predict what type of stroke will emerge next and that the sequence of categories of strokes appears to be random. In decision-theoretic terminology we would say that as each data stroke emerges is in one or the other of the two possible states: Either the stroke is a genuine or the stroke is impostor. Let p denote the state of nature, with $p = p_1$ for genuine and $p = p_2$ for impostor. Because the state of nature is so unpredictable, we consider p to be a variable that must be described probabilistically.

In the simplest condition we would say that next stroke is equally likely to be either genuine or impostor. More generally, we assume that there is some prior probability $P(p_1)$ that the next stroke is genuine, and some prior probability $P(p_2)$ if it is impostor. We know that in our condition there is no other category possible therefore$P(p_1) + P(p_2) = 1$. These prior probabilities reflect our prior knowledge of how likely we are to get a genuine or an impostor before the stroke actually appears.

In most circumstances, we are not asked to make decisions with so little information. In our condition for instance we use 17 different features to build the classifier and describe the properties of strokes. Different data strokes yields different feature readings. We consider this feature vector x to be a continuous set of random variable whose distribution depends on the state of nature and is expressed as $P(x|p)$. This is the class-conditional probability density function, which can also be reiterated as the probability density function for x given that the state of nature is p. The

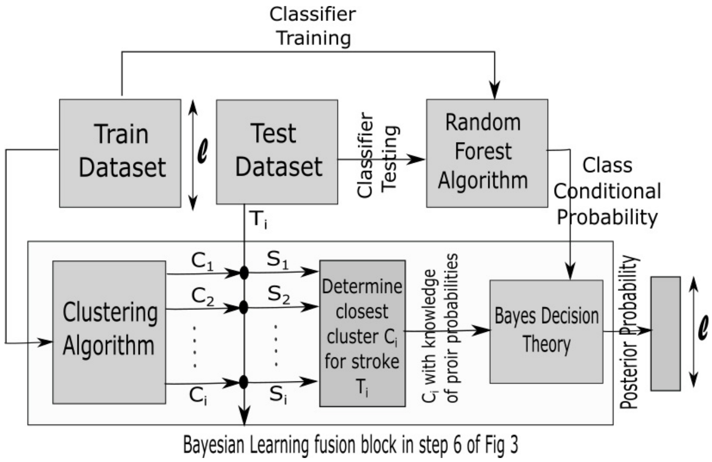


Figure 17.1: Bayesian Fusion mechanism of random forest and clustering algorithms

difference between $P(x|p_1)$ and $P(x|p_2)$ describes whether stroke belongs to genuine or impostor class.

When we have a knowledge about prior as well as class-conditional probabilities of a particular stroke, we can combine them using Bayes theorem for the decision making. The Bayes theorem can be formulated as follows:

$$P(p_i|x) = \frac{P(x|p_i) * P(p_i)}{P(x)}$$

where $P(p_i|x)$ is the joint probability function or the probability of category p given the feature vector x and $P(x)$ is the probability of a feature vector x to appear which is constant irrespective of its state of nature p.

Therefore in the Bayes theorem the decision rule becomes:

- If $P(p_1|x) > P(p_2|x)$ we chose $p_1$over $p_2$thus labeling the stroke as genuine and vice versa.

17.2 Experimental Setup:

The experimental setup for the fusion of Random Forest supervised classification and clustering based non-supervised classification is shown in Fig. 17.1. This is an explanation of Bayesian learning fusion block in step 6 of Fig 16.1.

- We pass the training dataset to the clustering algorithm and get the desired number of clusters shown by C_i in Fig 17.1. Each cluster has certain number of genuine and impostor strokes and therefore each cluster has a genuine probability density and impostor probability density.

Test Posture#	*f=50%*	*f=60%*	*f=70%*	*f=80%*	*f=90%*	*f=100%*
Trained on Posture # 1						
P # 1	*42*	*23*	*31*	*40*	*42*	*47*
P # 2	*76*	*51*	*57*	*58*	*74*	*81*
P # 3	*82*	*57*	*62*	*66*	*84*	*93*
Trained on Posture # 2						
P # 1	*80*	*55*	*61*	*64*	*75*	*84*
P # 2	*39*	*20*	*28*	*41*	*43*	*47*
P # 3	*71*	*48*	*54*	*60*	*72*	*74*
Trained on Posture # 3						
P # 1	*87*	*61*	*65*	*77*	*87*	*91*
P # 2	*73*	*51*	*57*	*60*	*75*	*79*
P # 3	*33*	*18*	*23*	*34*	*38*	*42*

Table 7: Improvement using Bayesian Fusion: Mis-predictions per 350 strokes represented in Median statistics

- Every stroke from test dataset is compared against the centroid of each cluster using the cosine similarity measure shown by S_i in Fig 17.1. The cluster that is closest to the test stroke T_i according to similarity measure is considered parent cluster of stroke T_i. since each cluster has a genuine and impostor probability density, therefore for any test stroke T_i depending on its parent cluster we determine the prior probabilities of each class. $P(p_1)$ = genuine probability density and $P(p_2)$ = impostor probability density.

- The class conditional probabilities $P(x|p_i)$ are calculated using traditional supervised random forest classifier user model. Finally we fuse them together using Bayes decision theory to produce the posterior probability which in turn will decide the final class of the test stroke T_i.

17.3 Results and Conclusion

Table 7 shows the fusion performance over traditional random forest classification. We used prior probabilities using clustering as our initial guess of what the type of stroke could be to strengthen our predictions. However, random forest is still our higher weighing evidence for prediction. Therefore we used Bayes decision theory if and only if the random forest confidence or class conditional probability is below certain degree. As shown in Table 7 parameter f represent range of confidence of random forest from 50% to certain f% e.g. if f is 70% then the class conditional probability

Test Posture#	*f=50%*	*f=60%*	*f=70%*	*f=80%*	*f=90%*	*f=100%*
Trained on Posture # 1						
P # 1	*42*	*17*	*22*	*38*	*51*	*88*
P # 2	*76*	*47*	*56*	*61*	*79*	*103*
P # 3	*82*	*42*	*49*	*58*	*78*	*97*
Trained on Posture # 2						
P # 1	*80*	*50*	*58*	*61*	*77*	*95*
P # 2	*39*	*15*	*22*	*34*	*47*	*79*
P # 3	*71*	*44*	*52*	*58*	*81*	*93*
Trained on Posture # 3						
P # 1	*87*	*56*	*61*	*73*	*81*	*102*
P # 2	*73*	*48*	*52*	*65*	*84*	*98*
P # 3	*33*	*12*	*19*	*29*	*45*	*62*

Table 8: Improvement using simple fusion: Mis-predictions per 350 strokes represented in Median statistics

lies between 50% to 70%. If f =X% it means we apply Bayes decision rule on T_i if and only if random forest class conditional probability is less than X% otherwise we use random forest prediction as our final prediction. When f =50%, it means we only use random forest as our final evidence without using prior probability knowledge whereas when f =100% we always use fusion irrespective of random forest class conditional probability.

- We now address our 3^{rd} research question 2 mentioned in Section2.2 i.e does fusion helps improving classification accuracy. Our results indicate that fusion mechanism improves the results by lowering the mis-prediction by as much as 50%.
- Note that in Bayesian fusion algorithm f =60% enhances the performance by as much as 50% over traditional random forest classification algorithm. This indicates that we should use fusion mechanism only when random forest confidence is between 50% to 60% otherwise use random forest as a final predictor.
- However, when we always use Bayesian fusion irrespective of random forest class conditional probability (f =100%) there is notable decline in the performance.
- Also note that when training and testing is done on the same posture orientation, the misprediction rate is far lower than any other combination This phenomenon is well explained by Syed et. al. [28].

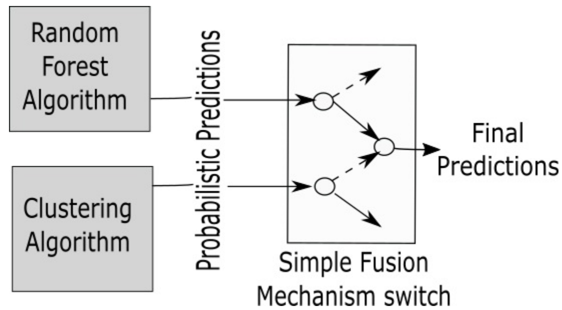


Figure 17.2: Simple fusion mechanism of random forest and clustering probabilistic predictions

17.4 Fusion using simple rule of combination

Section 17.1 implements Bayesian fusion, however it is just one of many methods of combinations. In this section we implement a simple fusion mechanism to combine two independent sets of probability mass assignments coming from random forest and clustering algorithms under specific situations. These two independent sources express their beliefs over the class predictions (genuine/impostor) such as giving hints or expressing preferences individually. In Section 17.1 we used the probabilistic predictions from clustering as our prior probability for Bayesian Learning. However, in this section we will treat probabilistic predictions from clustering as just another source of prediction besides random forest. Random forest and clustering algorithm have the same weightage in deciding the final predictions. This simple experimental setup is shown in Fig. 17.2 and implemented using following steps:

1. We implement step 1-5 shown in Fig. 16.1 to get the individual probabilistic class predictions from random forest and clustering algorithms.

2. It is then provided to the simple fusion mechanism which acts as switch as shown in Fig. 17.2. The switch mechanism chooses either random forest or clustering class prediction and label it as final class prediction for any specific stroke T_i. The switching between random forest and clustering predictions in simple fusion mechanism is based on random forest class prediction probability. If the random forest probability for a stroke T_i is less than $f\%$ then we only consider clustering prediction for stroke T_i over random forest in making final decision and vice versa.

17.5 Results and Conclusion

Table 8 compares the simple fusion algorithm performance with traditional random forest classification by number of mis-predictions for every 350 strokes. As mentioned earlier we used random forest and clustering as two independent sources of evidence.

As shown in Table 8 parameter f represent range of confidence of random forest from 50% to certain P% e.g. if f is 70% then the random forest class prediction probability lies between 50% to 70%. If f=P% it means we use clustering prediction as our final prediction for test stroke T_i if and only if random forest probability is less than P% otherwise we use random forest prediction as our final prediction. This is the only difference between Bayesian fusion and simple fusion mechanism. When f =50%, it means we only use random forest as our final evidence without acknowledging clustering algorithm prediction whereas when f =100% we always use clustering algorithm predictions irrespective of random forest classification results. Table 8 shows the performance of simple fusion algorithm when user model trained on posture x and tested on posture y.

- In simple fusion algorithm S = 60% gives us the optimum results with 59.5% improvement over traditional random forest algorithms. This indicates that we should use clustering predictions only when random forest class prediction probability is between 50% to 60%.
- From Table 8 it is clear that using only clustering algorithm for prediction decreases the classifier performance by a significant amount. Therefore it is imperative not to use predictions based on clustering solely.
- Note that simple fusion algorithm is more efficient way of fusing two sources of evidence compared to Bayesian fusion algorithm. Furthermore, these conclusions are valid for the device and application used in this dataset.
- Therefore we address our research question 3 mentioned in Section 2.2 and claim that simple fusion works better than Bayesian fusion for our dataset. However, these results may vary based on different combination of device and applications.

18 Summary and Future Work

In this part, we explored the presence of fusion mechanism in a touch based authentication dataset and its effects on reducing the classifier mis-predictions. Through a series of empirical experiments, we concluded that the hierarchical clustering with Cosine similarity measure is the best strategy to build the unsupervised clusters as it has lowest average SSE. However, note that we have also mentioned that number of clusters may vary based on individual touch profile. Furthermore, we have seen that using only clustering algorithm for future prediction actually degrades the classifier performance. Therefore we implemented different fusion mechanism viz. Bayesian fusion and simple fusion to fuse traditional supervised classifier algorithm in our case we used random forest with unsupervised clustering algorithm. We show that Bayesian fusion implementation when random forest classifier confidence is less than 60%, decreases the mis-classification rate by as much as 50%. We also show that for our dataset over 30 users averagely simple fusion mechanism works better than Bayesian fusion. Simple fusion mechanism enhances the overall prediction while reducing mis-classification rate by as low as ~60% where fusion occurs when random forest classifier confidence is less than 60%.

In the future, we plan to extend the study of different fusion mechanism to other devices types and user postures. Furthermore, as mentioned in Section 12 we are creating a belief based authentication system that modifies the classification threshold based on the time and decision of past strokes.

A Appendix

A.1 Benchmark analysis of classification algorithms

In Section 8, in order to determine the best performing classifier for our touchbased authentication system, we performed a benchmark analysis using our dataset. We tested six classification algorithms: CART, SVM, Random Forest, Naive Bayes, K-NN (k=10) and Multi-layer Linear Perceptron. This section of the appendix lists the EERs calculated at Step 6 of Fig. 8.1 for 30 users on all the blocks for each classification algorithm.

	Users														
	1			2			3			4			5		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.07	0.11	0.14	0.16	0.16	0.2	0.25	0.2	0.13	0.17	0.25	0.17	0.15	0.19	0.13
2	0.1	0.07	0.11	0.2	0.17	0.17	0.18	0.23	0.19	0.15	0.19	0.21	0.17	0.18	0.2
3	0.04	0.07	0.11	0.22	0.16	0.22	0.25	0.18	0.18	0.14	0.19	0.14	0.19	0.19	0.17
4	0.05	0.11	0.12	0.21	0.13	0.2	0.23	0.23	0.18	0.15	0.22	0.17	0.17	0.16	0.15
5	0.07	0.05	0.11	0.19	0.14	0.22	0.25	0.24	0.21	0.14	0.21	0.15	0.2	0.23	0.18
6	0.06	0.09	0.11	0.15	0.13	0.18	0.2	0.18	0.15	0.15	0.21	0.14	0.2	0.19	0.16
7	0.05	0.07	0.12	0.16	0.13	0.19	0.22	0.27	0.21	0.13	0.22	0.18	0.12	0.2	0.17
8	0.09	0.1	0.15	0.21	0.19	0.25	0.26	0.26	0.18	0.12	0.23	0.18	0.16	0.18	0.21
9	0.08	0.09	0.13	0.19	0.14	0.24	0.24	0.19	0.16	0.16	0.25	0.18	0.14	0.17	0.13
10	0.07	0.1	0.12	0.2	0.15	0.21	0.18	0.2	0.17	0.2	0.21	0.17	0.13	0.21	0.19

Table 9: EER for Decision Tree - User 1-5

	Users														
	6			7			8			9			10		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.16	0.15	0.11	0.07	0.09	0.08	0.02	0.04	0.01	0.15	0.1	0.18	0.17	0.2	0.15
2	0.17	0.14	0.13	0.05	0.1	0.11	0.03	0.02	0.02	0.09	0.09	0.2	0.19	0.13	0.13
3	0.15	0.12	0.11	0.05	0.07	0.13	0.02	0.03	0.02	0.09	0.09	0.14	0.19	0.19	0.15
4	0.12	0.07	0.14	0.05	0.1	0.09	0.02	0.03	0.02	0.14	0.09	0.15	0.16	0.18	0.14
5	0.17	0.14	0.11	0.05	0.07	0.11	0.02	0.03	0.05	0.09	0.09	0.24	0.19	0.14	0.12
6	0.13	0.13	0.09	0.05	0.09	0.12	0.06	0.05	0.02	0.08	0.1	0.15	0.21	0.15	0.16
7	0.18	0.11	0.15	0.04	0.07	0.14	0.04	0.04	0.03	0.11	0.09	0.2	0.17	0.13	0.17
8	0.17	0.12	0.13	0.07	0.09	0.16	0.03	0.03	0.03	0.09	0.07	0.14	0.17	0.19	0.2
9	0.16	0.17	0.1	0.05	0.07	0.09	0.04	0.05	0.05	0.09	0.07	0.18	0.16	0.19	0.18
10	0.15	0.09	0.15	0.07	0.11	0.13	0.02	0.02	0.03	0.1	0.09	0.17	0.13	0.15	0.14

Table 10: EER for Decision Tree - User 6-10

	Users														
	11			12			13			14			15		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.21	0.24	0.27	0.12	0.07	0.07	0.15	0.14	0.15	0.22	0.29	0.23	0.04	0.04	0.03
2	0.18	0.3	0.29	0.09	0.11	0.05	0.16	0.16	0.15	0.2	0.22	0.19	0.04	0.05	0.07
3	0.18	0.21	0.18	0.14	0.13	0.05	0.16	0.17	0.14	0.22	0.28	0.2	0.05	0.05	0.03
4	0.21	0.19	0.18	0.12	0.15	0.05	0.15	0.09	0.15	0.21	0.31	0.24	0.07	0.04	0.04
5	0.14	0.23	0.19	0.11	0.11	0.06	0.22	0.17	0.13	0.2	0.25	0.18	0.03	0.03	0.01
6	0.16	0.25	0.23	0.11	0.13	0.08	0.14	0.19	0.17	0.16	0.33	0.16	0.04	0.04	0.05
7	0.17	0.24	0.26	0.15	0.12	0.07	0.18	0.15	0.15	0.21	0.31	0.2	0.04	0.04	0.03
8	0.18	0.31	0.21	0.17	0.12	0.08	0.2	0.15	0.15	0.23	0.23	0.16	0.03	0.05	0.01
9	0.2	0.21	0.22	0.1	0.14	0.05	0.16	0.14	0.15	0.22	0.29	0.19	0.06	0.06	0.03
10	0.16	0.25	0.21	0.1	0.1	0.08	0.16	0.12	0.18	0.23	0.27	0.17	0.03	0.03	0.03

Table 11: EER for Decision Tree - User 11-15

	Users														
	16			17			18			19			20		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.25	0.19	0.17	0.14	0.11	0.16	0.14	0.14	0.14	0.11	0.17	0.29	0.18	0.15	0.15
2	0.26	0.18	0.1	0.17	0.12	0.11	0.15	0.18	0.2	0.15	0.15	0.23	0.19	0.22	0.16
3	0.29	0.11	0.13	0.14	0.11	0.14	0.11	0.13	0.13	0.07	0.14	0.2	0.18	0.19	0.19
4	0.22	0.14	0.13	0.18	0.19	0.18	0.17	0.18	0.15	0.12	0.2	0.2	0.16	0.19	0.13
5	0.26	0.2	0.16	0.18	0.13	0.17	0.12	0.12	0.16	0.12	0.13	0.2	0.13	0.21	0.17
6	0.24	0.2	0.13	0.14	0.12	0.13	0.14	0.17	0.14	0.13	0.22	0.24	0.21	0.2	0.16
7	0.25	0.16	0.14	0.12	0.17	0.12	0.14	0.15	0.16	0.14	0.16	0.24	0.21	0.2	0.13
8	0.22	0.17	0.11	0.16	0.12	0.14	0.12	0.18	0.22	0.13	0.15	0.23	0.2	0.22	0.15
9	0.2	0.14	0.15	0.13	0.15	0.18	0.15	0.15	0.15	0.14	0.17	0.21	0.13	0.22	0.17
10	0.25	0.14	0.15	0.17	0.12	0.17	0.11	0.15	0.16	0.18	0.16	0.22	0.22	0.22	0.15

Table 12: EER for Decision Tree - User 16-20

	Users														
	21			22			23			24			25		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.18	0.2	0.22	0.18	0.13	0.16	0.16	0.14	0.17	0.18	0.16	0.11	0.15	0.15	0.19
2	0.17	0.16	0.17	0.22	0.16	0.18	0.19	0.17	0.21	0.14	0.16	0.13	0.19	0.16	0.25
3	0.2	0.18	0.24	0.2	0.14	0.14	0.15	0.14	0.13	0.16	0.15	0.11	0.16	0.15	0.17
4	0.2	0.16	0.22	0.19	0.16	0.14	0.11	0.18	0.14	0.14	0.16	0.1	0.17	0.14	0.2
5	0.18	0.22	0.21	0.18	0.15	0.14	0.15	0.12	0.18	0.14	0.15	0.1	0.13	0.17	0.14
6	0.19	0.16	0.21	0.16	0.13	0.15	0.16	0.15	0.18	0.15	0.16	0.12	0.13	0.17	0.19
7	0.18	0.17	0.2	0.16	0.13	0.21	0.14	0.1	0.13	0.17	0.15	0.14	0.13	0.18	0.2
8	0.2	0.18	0.18	0.14	0.16	0.12	0.13	0.16	0.14	0.17	0.13	0.11	0.18	0.19	0.2
9	0.19	0.19	0.18	0.19	0.14	0.18	0.14	0.14	0.2	0.21	0.19	0.13	0.15	0.12	0.22
10	0.15	0.17	0.19	0.22	0.17	0.14	0.16	0.14	0.16	0.19	0.16	0.11	0.11	0.2	0.2

Table 13: EER for Decision Tree - User 21-25

	Users														
	26			27			28			29			30		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.23	0.26	0.18	0.17	0.2	0.19	0.21	0.3	0.24	0.15	0.2	0.18	0.17	0.23	0.19
2	0.16	0.23	0.2	0.18	0.15	0.18	0.2	0.22	0.22	0.17	0.22	0.17	0.21	0.24	0.22
3	0.2	0.22	0.27	0.17	0.18	0.22	0.23	0.24	0.2	0.19	0.23	0.18	0.2	0.21	0.19
4	0.26	0.2	0.21	0.16	0.19	0.19	0.18	0.25	0.25	0.19	0.15	0.18	0.2	0.28	0.25
5	0.25	0.2	0.2	0.13	0.19	0.17	0.19	0.23	0.25	0.17	0.18	0.17	0.22	0.24	0.19
6	0.2	0.24	0.21	0.12	0.21	0.17	0.2	0.24	0.19	0.17	0.16	0.17	0.22	0.29	0.23
7	0.18	0.26	0.22	0.15	0.2	0.21	0.21	0.22	0.16	0.17	0.21	0.19	0.17	0.24	0.19
8	0.22	0.22	0.21	0.18	0.14	0.19	0.24	0.3	0.22	0.22	0.21	0.18	0.23	0.24	0.24
9	0.23	0.28	0.19	0.15	0.19	0.23	0.17	0.23	0.24	0.15	0.22	0.16	0.22	0.23	0.22
10	0.19	0.23	0.27	0.23	0.21	0.22	0.2	0.25	0.22	0.15	0.21	0.15	0.23	0.29	0.2

Table 14: EER for Decision Tree - User 26-30

	Users														
	1			2			3			4			5		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.10	0.08	0.10	0.12	0.17	0.15	0.16	0.16	0.08	0.09	0.13	0.11	0.16	0.10	0.16
2	0.10	0.07	0.07	0.18	0.17	0.19	0.14	0.19	0.09	0.12	0.13	0.12	0.17	0.12	0.13
3	0.08	0.08	0.09	0.16	0.16	0.15	0.17	0.19	0.13	0.11	0.15	0.12	0.13	0.12	0.17
4	0.08	0.05	0.08	0.15	0.14	0.18	0.15	0.17	0.11	0.11	0.11	0.12	0.13	0.11	0.13
5	0.07	0.05	0.05	0.16	0.16	0.17	0.17	0.15	0.10	0.10	0.14	0.11	0.16	0.12	0.14
6	0.08	0.07	0.05	0.16	0.17	0.19	0.18	0.22	0.11	0.11	0.15	0.13	0.11	0.13	0.14
7	0.06	0.08	0.08	0.15	0.17	0.19	0.15	0.15	0.10	0.10	0.10	0.09	0.09	0.07	0.13
8	0.09	0.05	0.10	0.18	0.18	0.17	0.18	0.21	0.13	0.13	0.15	0.12	0.15	0.14	0.17
9	0.11	0.06	0.08	0.15	0.18	0.17	0.15	0.16	0.14	0.12	0.16	0.11	0.13	0.13	0.12
10	0.09	0.05	0.09	0.14	0.17	0.16	0.18	0.19	0.10	0.11	0.15	0.11	0.15	0.11	0.14

Table 15: EER for K- Nearest Neighbor (K=10) - User 1-5

	Users														
	6			7			8			9			10		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.13	0.12	0.11	0.08	0.08	0.08	0.03	0.04	0.02	0.12	0.11	0.18	0.15	0.16	0.16
2	0.16	0.12	0.11	0.09	0.11	0.10	0.03	0.03	0.03	0.09	0.07	0.15	0.08	0.10	0.12
3	0.14	0.11	0.11	0.07	0.07	0.08	0.04	0.04	0.04	0.12	0.08	0.22	0.13	0.16	0.14
4	0.13	0.08	0.10	0.07	0.08	0.09	0.02	0.03	0.03	0.09	0.12	0.19	0.13	0.15	0.14
5	0.12	0.08	0.11	0.08	0.06	0.08	0.03	0.04	0.03	0.07	0.11	0.19	0.15	0.17	0.15
6	0.12	0.07	0.10	0.08	0.08	0.09	0.03	0.04	0.03	0.09	0.11	0.19	0.13	0.17	0.15
7	0.11	0.09	0.10	0.08	0.07	0.09	0.03	0.03	0.04	0.10	0.09	0.15	0.12	0.17	0.17
8	0.17	0.10	0.10	0.09	0.10	0.10	0.03	0.04	0.03	0.10	0.10	0.14	0.16	0.17	0.13
9	0.19	0.11	0.12	0.12	0.11	0.10	0.04	0.03	0.04	0.12	0.09	0.18	0.15	0.15	0.14
10	0.13	0.08	0.10	0.07	0.08	0.09	0.04	0.03	0.04	0.08	0.11	0.18	0.15	0.16	0.14

Table 16: EER for K- Nearest Neighbor (K=10) - User 6-10

	Users														
	11			12			13			14			15		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.20	0.24	0.27	0.10	0.10	0.06	0.18	0.13	0.14	0.21	0.26	0.15	0.07	0.07	0.09
2	0.21	0.26	0.26	0.10	0.08	0.08	0.17	0.08	0.12	0.21	0.20	0.17	0.09	0.08	0.09
3	0.18	0.21	0.23	0.11	0.11	0.10	0.22	0.10	0.14	0.17	0.23	0.16	0.07	0.07	0.08
4	0.21	0.24	0.27	0.14	0.12	0.07	0.17	0.07	0.10	0.20	0.28	0.21	0.05	0.06	0.07
5	0.19	0.24	0.21	0.12	0.10	0.07	0.21	0.15	0.15	0.22	0.25	0.18	0.07	0.07	0.07
6	0.19	0.23	0.24	0.10	0.11	0.10	0.21	0.10	0.15	0.16	0.22	0.15	0.07	0.08	0.08
7	0.18	0.23	0.23	0.13	0.11	0.07	0.19	0.13	0.13	0.19	0.24	0.18	0.07	0.07	0.08
8	0.20	0.24	0.25	0.13	0.11	0.09	0.20	0.12	0.15	0.18	0.23	0.16	0.07	0.07	0.08
9	0.22	0.25	0.23	0.13	0.14	0.10	0.18	0.10	0.13	0.23	0.24	0.17	0.04	0.06	0.06
10	0.19	0.20	0.23	0.11	0.12	0.05	0.19	0.09	0.16	0.19	0.23	0.17	0.07	0.06	0.08

Table 17: EER for K Nearest Neighbor (K=10) - User 11-15

	Users														
	16			17			18			19			20		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.30	0.19	0.19	0.16	0.18	0.18	0.17	0.18	0.17	0.16	0.18	0.21	0.20	0.21	0.16
2	0.30	0.21	0.18	0.19	0.20	0.19	0.20	0.19	0.18	0.19	0.18	0.17	0.18	0.20	0.14
3	0.27	0.14	0.16	0.18	0.14	0.14	0.18	0.20	0.17	0.21	0.13	0.18	0.21	0.20	0.16
4	0.34	0.21	0.22	0.21	0.17	0.19	0.22	0.19	0.20	0.21	0.17	0.19	0.21	0.19	0.14
5	0.25	0.19	0.20	0.20	0.19	0.17	0.20	0.17	0.20	0.23	0.19	0.18	0.16	0.17	0.14
6	0.26	0.19	0.17	0.18	0.19	0.18	0.17	0.15	0.17	0.21	0.17	0.16	0.17	0.16	0.12
7	0.26	0.19	0.16	0.16	0.20	0.18	0.18	0.18	0.16	0.18	0.16	0.17	0.16	0.16	0.13
8	0.27	0.17	0.15	0.18	0.17	0.17	0.15	0.17	0.20	0.18	0.20	0.21	0.22	0.21	0.18
9	0.34	0.21	0.20	0.19	0.19	0.21	0.18	0.18	0.20	0.19	0.16	0.19	0.23	0.19	0.14
10	0.27	0.19	0.17	0.21	0.17	0.16	0.19	0.18	0.19	0.19	0.13	0.16	0.23	0.22	0.17

Table 18: EER for K Nearest Neighbor (K=10) - User 16-20

	Users														
	21			22			23			24			25		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.21	0.18	0.21	0.21	0.20	0.19	0.23	0.19	0.23	0.20	0.19	0.17	0.16	0.21	0.23
2	0.19	0.14	0.15	0.20	0.20	0.16	0.18	0.16	0.18	0.20	0.16	0.17	0.17	0.20	0.23
3	0.19	0.18	0.20	0.23	0.20	0.22	0.19	0.17	0.17	0.20	0.19	0.16	0.18	0.16	0.20
4	0.19	0.17	0.16	0.27	0.20	0.17	0.21	0.17	0.19	0.21	0.19	0.17	0.18	0.16	0.21
5	0.19	0.19	0.20	0.19	0.19	0.18	0.19	0.16	0.20	0.22	0.20	0.17	0.16	0.20	0.23
6	0.20	0.11	0.13	0.18	0.19	0.14	0.19	0.17	0.19	0.22	0.19	0.17	0.13	0.21	0.20
7	0.18	0.13	0.18	0.18	0.19	0.17	0.20	0.18	0.19	0.22	0.20	0.20	0.14	0.21	0.17
8	0.20	0.15	0.18	0.24	0.21	0.19	0.16	0.15	0.17	0.19	0.16	0.17	0.16	0.19	0.19
9	0.21	0.14	0.17	0.22	0.19	0.19	0.21	0.17	0.22	0.22	0.20	0.17	0.14	0.19	0.17
10	0.18	0.16	0.18	0.23	0.20	0.20	0.20	0.17	0.21	0.22	0.18	0.15	0.18	0.21	0.23

Table 19: EER for K Nearest Neighbor (K=10) - User 21-25

	Users														
	26			27			28			29			30		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.23	0.22	0.17	0.19	0.17	0.15	0.17	0.20	0.24	0.13	0.18	0.18	0.15	0.25	0.19
2	0.21	0.21	0.15	0.17	0.16	0.18	0.16	0.23	0.24	0.12	0.11	0.16	0.15	0.27	0.21
3	0.23	0.22	0.20	0.17	0.19	0.17	0.15	0.23	0.24	0.13	0.17	0.18	0.22	0.24	0.19
4	0.23	0.24	0.20	0.17	0.21	0.16	0.18	0.23	0.27	0.13	0.15	0.16	0.18	0.25	0.20
5	0.22	0.22	0.23	0.16	0.15	0.16	0.17	0.22	0.22	0.14	0.16	0.17	0.16	0.29	0.21
6	0.22	0.20	0.21	0.19	0.20	0.16	0.16	0.21	0.24	0.13	0.15	0.18	0.16	0.26	0.18
7	0.22	0.22	0.20	0.19	0.19	0.16	0.17	0.22	0.22	0.16	0.18	0.18	0.17	0.26	0.15
8	0.24	0.19	0.22	0.19	0.18	0.16	0.19	0.25	0.28	0.15	0.16	0.21	0.22	0.24	0.20
9	0.26	0.23	0.20	0.19	0.23	0.21	0.18	0.23	0.25	0.15	0.16	0.16	0.20	0.24	0.21
10	0.24	0.21	0.19	0.15	0.19	0.16	0.17	0.24	0.21	0.14	0.13	0.17	0.18	0.23	0.21

Table 20: EER for K Nearest Neighbor (K=10) - User 26-30

	Users														
	1			2			3			4			5		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.22	0.13	0.21	0.17	0.46	0.32	0.25	0.40	0.42	0.28	0.28	0.26	0.57	0.55	0.35
2	0.25	0.09	0.22	0.38	0.46	0.53	0.25	0.40	0.32	0.34	0.37	0.30	0.57	0.55	0.35
3	0.16	0.24	0.21	0.38	0.46	0.53	0.25	0.40	0.29	0.17	0.37	0.28	0.57	0.55	0.36
4	0.19	0.19	0.16	0.38	0.46	0.53	0.25	0.50	0.43	0.29	0.32	0.28	0.57	0.33	0.36
5	0.19	0.10	0.21	0.67	0.36	0.50	0.30	0.50	0.30	0.29	0.38	0.32	0.57	0.33	0.36
6	0.25	0.05	0.19	0.67	0.41	0.50	0.55	0.38	0.30	0.26	0.30	0.32	0.40	0.30	0.48
7	0.19	0.11	0.19	0.36	0.55	0.50	0.55	0.44	0.24	0.26	0.23	0.32	0.40	0.30	0.48
8	0.33	0.07	0.21	0.27	0.55	0.50	0.53	0.39	0.24	0.26	0.23	0.31	0.42	0.30	0.48
9	0.18	0.11	0.18	0.27	0.32	0.50	0.35	0.38	0.32	0.28	0.33	0.27	0.42	0.35	0.48
10	0.12	0.11	0.17	0.38	0.32	0.50	0.40	0.42	0.39	0.28	0.37	0.38	0.42	0.35	0.48

Table 21: EER for Linear Perceptron - User 1-5

	Users														
	6			7			8			9			10		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.21	0.17	0.20	0.21	0.34	0.36	0.05	0.01	0.05	0.17	0.09	0.17	0.43	0.33	0.36
2	0.21	0.29	0.22	0.21	0.27	0.26	0.03	0.03	0.06	0.25	0.17	0.11	0.42	0.67	0.25
3	0.30	0.17	0.21	0.22	0.27	0.26	0.04	0.05	0.03	0.29	0.16	0.11	0.40	0.67	0.25
4	0.20	0.30	0.24	0.22	0.37	0.41	0.05	0.10	0.26	0.36	0.09	0.50	0.36	0.67	0.39
5	0.20	0.26	0.30	0.22	0.29	0.37	0.08	0.03	0.24	0.40	0.24	0.11	0.36	0.36	0.39
6	0.20	0.31	0.26	0.30	0.23	0.33	0.08	0.09	0.20	0.38	0.08	0.42	0.23	0.36	0.23
7	0.20	0.21	0.28	0.30	0.23	0.40	0.06	0.03	0.04	0.38	0.27	0.42	0.23	0.36	0.23
8	0.33	0.20	0.22	0.30	0.23	0.36	0.04	0.04	0.08	0.25	0.10	0.42	0.41	0.36	0.42
9	0.35	0.20	0.21	0.30	0.36	0.23	0.05	0.11	0.21	0.18	0.10	0.21	0.32	0.36	0.42
10	0.28	0.18	0.21	0.32	0.28	0.23	0.04	0.05	0.17	0.24	0.25	0.44	0.33	0.36	0.25

Table 22: EER for Linear Perceptron - User 6-10

	Users														
	11			12			13			14			15		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.24	0.23	0.36	0.29	0.18	0.14	0.40	0.37	0.26	0.38	0.36	0.35	0.15	0.31	0.17
2	0.24	0.26	0.36	0.29	0.36	0.11	0.38	0.21	0.28	0.38	0.42	0.35	0.15	0.26	0.18
3	0.24	0.41	0.36	0.22	0.27	0.10	0.39	0.24	0.28	0.39	0.42	0.34	0.19	0.26	0.30
4	0.24	0.58	0.36	0.22	0.27	0.25	0.41	0.26	0.35	0.33	0.56	0.25	0.17	0.24	0.23
5	0.24	0.35	0.36	0.29	0.27	0.17	0.41	0.28	0.31	0.40	0.56	0.33	0.17	0.28	0.23
6	0.24	0.36	0.53	0.20	0.29	0.13	0.33	0.24	0.26	0.39	0.56	0.33	0.15	0.27	0.14
7	0.25	0.36	0.40	0.31	0.32	0.22	0.33	0.30	0.26	0.32	0.56	0.31	0.24	0.25	0.26
8	0.25	0.36	0.40	0.44	0.32	0.19	0.41	0.36	0.32	0.32	0.36	0.36	0.19	0.25	0.08
9	0.26	0.36	0.40	0.18	0.22	0.15	0.39	0.27	0.32	0.36	0.35	0.33	0.23	0.36	0.17
10	0.23	0.36	0.40	0.18	0.33	0.12	0.39	0.23	0.28	0.36	0.35	0.33	0.12	0.27	0.13

Table 23: EER for Linear Perceptron - User 11-15

	Users														
	16			17			18			19			20		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.13	0.30	0.21	0.29	0.25	0.29	0.26	0.41	0.28	0.60	0.43	0.36	0.53	0.44	0.54
2	0.29	0.22	0.35	0.31	0.29	0.29	0.26	0.33	0.28	0.60	0.35	0.38	0.31	0.44	0.29
3	0.37	0.25	0.26	0.31	0.50	0.29	0.26	0.24	0.56	0.60	0.46	0.36	0.31	0.50	0.23
4	0.37	0.25	0.26	0.32	0.50	0.29	0.26	0.50	0.67	0.60	0.31	0.34	0.31	0.43	0.23
5	0.46	0.23	0.33	0.32	0.40	0.33	0.26	0.33	0.67	0.60	0.31	0.34	0.31	0.43	0.27
6	0.41	0.22	0.33	0.32	0.40	0.33	0.26	0.33	0.65	0.47	0.32	0.34	0.31	0.50	0.25
7	0.38	0.22	0.22	0.32	0.40	0.33	0.26	0.50	0.50	0.44	0.60	0.33	0.37	0.50	0.26
8	0.40	0.29	0.22	0.32	0.28	0.50	0.26	0.60	0.35	0.44	0.36	0.41	0.42	0.50	0.27
9	0.30	0.29	0.22	0.32	0.28	0.31	0.67	0.28	0.35	0.44	0.36	0.34	0.42	0.54	0.27
10	0.30	0.20	0.35	0.27	0.29	0.36	0.67	0.28	0.60	0.43	0.36	0.34	0.34	0.54	0.27

Table 24: EER for Linear Perceptron - User 16-20

	Users														
	21			22			23			24			25		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.27	0.50	0.35	0.31	0.42	0.33	0.38	0.29	0.44	0.35	0.44	0.30	0.42	0.44	0.44
2	0.36	0.30	0.33	0.38	0.42	0.41	0.38	0.37	0.44	0.35	0.40	0.43	0.42	0.44	0.44
3	0.31	0.24	0.40	0.29	0.45	0.41	0.67	0.31	0.44	0.33	0.41	0.35	0.36	0.40	0.44
4	0.43	0.24	0.36	0.50	0.43	0.41	0.40	0.31	0.44	0.33	0.41	0.35	0.32	0.40	0.44
5	0.44	0.23	0.36	0.31	0.33	0.28	0.47	0.32	0.33	0.50	0.41	0.35	0.34	0.40	0.44
6	0.44	0.35	0.31	0.50	0.40	0.28	0.41	0.32	0.38	0.50	0.41	0.35	0.35	0.40	0.48
7	0.37	0.23	0.45	0.50	0.43	0.38	0.45	0.24	0.28	0.42	0.47	0.50	0.39	0.40	0.56
8	0.43	0.30	0.45	0.37	0.40	0.38	0.45	0.24	0.28	0.48	0.41	0.54	0.39	0.28	0.50
9	0.50	0.23	0.33	0.37	0.40	0.38	0.45	0.24	0.67	0.48	0.41	0.38	0.33	0.54	0.50
10	0.50	0.23	0.31	0.45	0.40	0.38	0.45	0.36	0.67	0.42	0.52	0.42	0.44	0.54	0.50

Table 25: EER for Linear Perceptron - User 21-25

	Users														
	26			27			28			29			30		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.50	0.56	0.47	0.28	0.26	0.24	0.38	0.25	0.20	0.50	0.32	0.35	0.40	0.67	0.38
2	0.60	0.56	0.47	0.32	0.42	0.24	0.38	0.23	0.20	0.50	0.32	0.35	0.40	0.32	0.38
3	0.60	0.53	0.39	0.33	0.42	0.24	0.28	0.29	0.37	0.50	0.32	0.35	0.50	0.32	0.33
4	0.60	0.50	0.39	0.33	0.42	0.24	0.28	0.29	0.37	0.50	0.32	0.45	0.50	0.54	0.29
5	0.60	0.50	0.39	0.33	0.42	0.36	0.33	0.34	0.37	0.50	0.32	0.45	0.33	0.54	0.29
6	0.57	0.50	0.60	0.40	0.42	0.35	0.25	0.34	0.37	0.32	0.32	0.45	0.33	0.47	0.29
7	0.47	0.47	0.43	0.32	0.33	0.35	0.37	0.37	0.33	0.32	0.28	0.45	0.33	0.36	0.33
8	0.47	0.47	0.43	0.25	0.38	0.35	0.33	0.37	0.33	0.32	0.33	0.45	0.33	0.36	0.33
9	0.56	0.47	0.43	0.25	0.38	0.35	0.38	0.37	0.50	0.32	0.33	0.67	0.33	0.36	0.33
10	0.56	0.47	0.43	0.26	0.38	0.38	0.25	0.50	0.50	0.32	0.21	0.37	0.35	0.38	0.33

Table 26: EER for Linear Perceptron - User 26-30

	Users														
	1			2			3			4			5		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.12	0.20	0.16	0.22	0.16	0.24	0.28	0.28	0.30	0.22	0.24	0.23	0.40	0.46	0.21
2	0.10	0.12	0.16	0.18	0.28	0.21	0.31	0.28	0.22	0.16	0.22	0.27	0.33	0.45	0.19
3	0.09	0.12	0.12	0.18	0.28	0.19	0.33	0.25	0.17	0.13	0.25	0.27	0.35	0.45	0.21
4	0.10	0.13	0.15	0.19	0.24	0.19	0.29	0.26	0.18	0.15	0.21	0.25	0.26	0.45	0.19
5	0.14	0.25	0.21	0.26	0.19	0.27	0.28	0.29	0.26	0.14	0.24	0.26	0.20	0.25	0.23
6	0.07	0.10	0.10	0.20	0.32	0.20	0.34	0.22	0.19	0.11	0.25	0.25	0.38	0.45	0.20
7	0.15	0.22	0.18	0.25	0.13	0.26	0.31	0.28	0.21	0.23	0.19	0.20	0.16	0.26	0.20
8	0.11	0.13	0.16	0.17	0.31	0.16	0.33	0.26	0.22	0.14	0.29	0.28	0.37	0.26	0.24
9	0.09	0.11	0.13	0.19	0.28	0.19	0.29	0.25	0.20	0.14	0.27	0.24	0.37	0.42	0.24
10	0.09	0.13	0.14	0.24	0.21	0.27	0.28	0.30	0.24	0.15	0.18	0.27	0.22	0.48	0.17

Table 27: EER for Naive Bayes Classfier - User 1-5

	Users														
	6			7			8			9			10		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.30	0.12	0.23	0.14	0.11	0.15	0.03	0.03	0.03	0.10	0.08	0.18	0.20	0.21	0.21
2	0.30	0.20	0.20	0.11	0.11	0.12	0.03	0.07	0.02	0.10	0.06	0.17	0.31	0.21	0.15
3	0.27	0.23	0.15	0.08	0.10	0.12	0.03	0.07	0.03	0.08	0.09	0.14	0.37	0.21	0.18
4	0.26	0.18	0.14	0.08	0.10	0.11	0.03	0.06	0.01	0.09	0.07	0.14	0.30	0.21	0.19
5	0.26	0.16	0.22	0.14	0.13	0.14	0.03	0.03	0.04	0.12	0.05	0.24	0.16	0.27	0.25
6	0.20	0.27	0.13	0.07	0.06	0.10	0.02	0.10	0.03	0.11	0.10	0.21	0.38	0.25	0.17
7	0.20	0.15	0.21	0.14	0.13	0.15	0.03	0.02	0.02	0.10	0.06	0.18	0.17	0.23	0.21
8	0.27	0.25	0.16	0.11	0.11	0.13	0.04	0.08	0.05	0.08	0.08	0.10	0.37	0.25	0.17
9	0.27	0.20	0.16	0.08	0.09	0.10	0.04	0.11	0.03	0.09	0.10	0.17	0.34	0.22	0.16
10	0.25	0.14	0.18	0.11	0.13	0.13	0.03	0.04	0.03	0.10	0.07	0.16	0.25	0.21	0.19

Table 28: EER for Naive Bayes Classifier - User 6-10

	Users														
	11			12			13			14			15		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.26	0.33	0.33	0.13	0.13	0.08	0.13	0.11	0.10	0.22	0.39	0.19	0.04	0.07	0.04
2	0.24	0.35	0.34	0.13	0.12	0.06	0.16	0.14	0.14	0.24	0.35	0.29	0.06	0.13	0.06
3	0.18	0.33	0.32	0.11	0.08	0.05	0.18	0.18	0.15	0.25	0.31	0.29	0.05	0.15	0.06
4	0.23	0.37	0.32	0.12	0.11	0.06	0.16	0.16	0.13	0.22	0.35	0.26	0.05	0.13	0.05
5	0.26	0.37	0.34	0.21	0.20	0.09	0.21	0.11	0.17	0.24	0.40	0.23	0.05	0.05	0.06
6	0.19	0.35	0.30	0.10	0.09	0.08	0.23	0.18	0.18	0.27	0.31	0.23	0.05	0.17	0.06
7	0.23	0.35	0.29	0.19	0.17	0.11	0.19	0.19	0.17	0.22	0.38	0.20	0.07	0.05	0.05
8	0.20	0.33	0.33	0.13	0.11	0.09	0.20	0.16	0.16	0.29	0.31	0.52	0.05	0.17	0.06
9	0.23	0.35	0.31	0.10	0.09	0.07	0.21	0.17	0.14	0.27	0.32	0.52	0.04	0.14	0.05
10	0.22	0.30	0.34	0.16	0.15	0.06	0.15	0.10	0.12	0.20	0.38	0.19	0.05	0.13	0.05

Table 29: EER for Naive Bayes Classifier - User 11-15

	Users														
	16			17			18			19			20		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.37	0.34	0.19	0.26	0.23	0.22	0.19	0.20	0.16	0.23	0.27	0.27	0.25	0.23	0.20
2	0.35	0.27	0.18	0.24	0.22	0.22	0.20	0.18	0.18	0.22	0.23	0.22	0.18	0.21	0.16
3	0.34	0.25	0.14	0.20	0.18	0.19	0.17	0.10	0.09	0.18	0.17	0.24	0.19	0.18	0.17
4	0.34	0.29	0.18	0.23	0.21	0.21	0.23	0.17	0.17	0.17	0.23	0.22	0.18	0.19	0.14
5	0.34	0.32	0.20	0.25	0.24	0.23	0.20	0.18	0.18	0.24	0.31	0.33	0.26	0.26	0.20
6	0.27	0.21	0.15	0.21	0.18	0.19	0.16	0.12	0.11	0.19	0.18	0.24	0.20	0.18	0.13
7	0.34	0.32	0.19	0.24	0.21	0.22	0.19	0.15	0.16	0.21	0.24	0.26	0.20	0.22	0.17
8	0.28	0.19	0.14	0.20	0.18	0.19	0.17	0.13	0.11	0.18	0.20	0.22	0.17	0.19	0.16
9	0.31	0.22	0.17	0.21	0.17	0.20	0.18	0.13	0.13	0.16	0.18	0.23	0.18	0.17	0.13
10	0.35	0.22	0.15	0.22	0.19	0.21	0.18	0.14	0.14	0.22	0.24	0.25	0.19	0.24	0.21

Table 30: EER for Naive Bayes Classifier - User 16-20

	Users														
	21			22			23			24			25		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.29	0.33	0.26	0.26	0.22	0.20	0.25	0.23	0.28	0.23	0.25	0.17	0.26	0.27	0.20
2	0.26	0.33	0.29	0.26	0.20	0.32	0.21	0.18	0.23	0.22	0.22	0.13	0.21	0.19	0.20
3	0.22	0.39	0.30	0.20	0.17	0.33	0.16	0.12	0.22	0.18	0.19	0.13	0.17	0.21	0.20
4	0.22	0.37	0.29	0.24	0.19	0.27	0.20	0.22	0.23	0.19	0.21	0.11	0.23	0.20	0.28
5	0.29	0.18	0.25	0.27	0.21	0.12	0.24	0.21	0.28	0.23	0.24	0.17	0.28	0.27	0.23
6	0.23	0.39	0.25	0.19	0.13	0.12	0.18	0.15	0.22	0.18	0.20	0.14	0.13	0.20	0.41
7	0.26	0.23	0.21	0.20	0.18	0.15	0.20	0.19	0.23	0.25	0.26	0.18	0.24	0.21	0.22
8	0.22	0.23	0.29	0.19	0.17	0.15	0.17	0.17	0.21	0.17	0.21	0.13	0.19	0.20	0.22
9	0.20	0.23	0.29	0.22	0.18	0.36	0.20	0.17	0.22	0.18	0.22	0.12	0.14	0.20	0.36
10	0.26	0.30	0.30	0.23	0.18	0.26	0.22	0.18	0.26	0.20	0.21	0.12	0.24	0.24	0.25

Table 31: EER for Naive Bayes Classifier- User 21-25

	Users														
	26			27			28			29			30		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.24	0.28	0.26	0.22	0.27	0.19	0.16	0.36	0.36	0.19	0.19	0.22	0.24	0.37	0.35
2	0.24	0.23	0.16	0.21	0.22	0.19	0.26	0.31	0.30	0.16	0.17	0.16	0.22	0.34	0.34
3	0.32	0.22	0.23	0.20	0.20	0.20	0.25	0.25	0.28	0.14	0.19	0.18	0.25	0.32	0.30
4	0.25	0.25	0.22	0.19	0.22	0.18	0.23	0.27	0.31	0.14	0.16	0.17	0.20	0.32	0.33
5	0.30	0.29	0.30	0.24	0.24	0.20	0.20	0.37	0.34	0.23	0.28	0.25	0.31	0.32	0.35
6	0.35	0.19	0.21	0.20	0.21	0.24	0.29	0.25	0.28	0.15	0.21	0.17	0.17	0.30	0.29
7	0.24	0.19	0.30	0.25	0.27	0.19	0.23	0.37	0.29	0.22	0.25	0.20	0.31	0.40	0.35
8	0.35	0.24	0.22	0.21	0.22	0.23	0.28	0.25	0.29	0.16	0.20	0.19	0.22	0.33	0.29
9	0.34	0.23	0.22	0.20	0.21	0.21	0.25	0.25	0.32	0.14	0.19	0.17	0.20	0.32	0.32
10	0.28	0.28	0.24	0.20	0.23	0.17	0.18	0.34	0.32	0.19	0.19	0.18	0.26	0.35	0.31

Table 32: EER for Naive Bayes Classifier - User 26-30

	Users														
	1			2			3			4			5		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.06	0.05	0.07	0.11	0.09	0.15	0.17	0.17	0.09	0.09	0.11	0.11	0.08	0.14	0.12
2	0.02	0.03	0.06	0.13	0.11	0.11	0.19	0.18	0.10	0.09	0.12	0.10	0.09	0.10	0.10
3	0.04	0.04	0.07	0.12	0.11	0.14	0.15	0.16	0.11	0.09	0.13	0.09	0.08	0.09	0.09
4	0.04	0.05	0.09	0.13	0.09	0.13	0.15	0.16	0.10	0.06	0.11	0.08	0.09	0.10	0.10
5	0.03	0.03	0.05	0.15	0.12	0.12	0.16	0.16	0.11	0.08	0.11	0.12	0.11	0.13	0.11
6	0.04	0.03	0.07	0.11	0.12	0.12	0.15	0.16	0.12	0.12	0.13	0.13	0.08	0.12	0.10
7	0.04	0.03	0.05	0.12	0.10	0.13	0.17	0.15	0.12	0.10	0.12	0.11	0.11	0.11	0.11
8	0.07	0.04	0.07	0.15	0.11	0.15	0.15	0.18	0.09	0.09	0.14	0.11	0.12	0.13	0.12
9	0.07	0.03	0.06	0.13	0.13	0.14	0.15	0.13	0.12	0.11	0.15	0.13	0.08	0.11	0.12
10	0.07	0.04	0.07	0.10	0.10	0.12	0.15	0.16	0.07	0.08	0.12	0.11	0.08	0.13	0.11

Table 33: EER for Random Forest Classfier - User 1-5

	Users														
	6			7			8			9			10		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.11	0.09	0.09	0.04	0.04	0.07	0.01	0.01	0.01	0.05	0.05	0.09	0.10	0.09	0.13
2	0.08	0.06	0.06	0.03	0.03	0.03	0.01	0.02	0.02	0.06	0.03	0.10	0.06	0.06	0.10
3	0.11	0.06	0.07	0.06	0.05	0.07	0.03	0.03	0.03	0.06	0.03	0.10	0.08	0.11	0.13
4	0.09	0.07	0.08	0.05	0.03	0.06	0.01	0.01	0.01	0.06	0.04	0.09	0.12	0.09	0.09
5	0.10	0.05	0.08	0.03	0.03	0.04	0.02	0.02	0.02	0.03	0.05	0.06	0.10	0.09	0.10
6	0.09	0.03	0.03	0.02	0.03	0.04	0.01	0.02	0.02	0.05	0.04	0.11	0.09	0.08	0.13
7	0.09	0.06	0.07	0.04	0.03	0.05	0.03	0.03	0.03	0.06	0.06	0.09	0.06	0.08	0.13
8	0.10	0.09	0.10	0.06	0.05	0.07	0.03	0.03	0.03	0.05	0.04	0.10	0.09	0.10	0.11
9	0.13	0.07	0.09	0.05	0.04	0.08	0.03	0.03	0.03	0.03	0.03	0.07	0.09	0.08	0.09
10	0.10	0.05	0.06	0.04	0.07	0.06	0.02	0.03	0.03	0.05	0.05	0.11	0.09	0.10	0.11

Table 34: EER for Random Forest Classifier - User 6-10

	Users														
	11			12			13			14			15		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.13	0.15	0.15	0.07	0.07	0.03	0.10	0.09	0.08	0.17	0.20	0.10	0.03	0.03	0.04
2	0.13	0.18	0.18	0.05	0.05	0.04	0.13	0.06	0.09	0.18	0.16	0.10	0.05	0.06	0.07
3	0.11	0.12	0.12	0.08	0.06	0.05	0.11	0.08	0.12	0.13	0.16	0.14	0.04	0.05	0.05
4	0.15	0.16	0.14	0.09	0.06	0.04	0.09	0.08	0.11	0.15	0.16	0.16	0.04	0.04	0.04
5	0.13	0.15	0.12	0.07	0.06	0.03	0.13	0.09	0.10	0.15	0.15	0.13	0.02	0.02	0.02
6	0.11	0.12	0.11	0.07	0.06	0.07	0.10	0.08	0.10	0.12	0.14	0.07	0.03	0.04	0.04
7	0.12	0.17	0.17	0.09	0.09	0.06	0.08	0.10	0.09	0.15	0.21	0.07	0.04	0.05	0.04
8	0.12	0.16	0.15	0.08	0.06	0.06	0.10	0.09	0.12	0.12	0.14	0.08	0.03	0.03	0.03
9	0.10	0.14	0.13	0.10	0.08	0.04	0.11	0.07	0.07	0.13	0.18	0.11	0.03	0.03	0.01
10	0.13	0.16	0.13	0.07	0.05	0.05	0.09	0.06	0.09	0.15	0.13	0.09	0.03	0.04	0.04

Table 35: EER for Naive Bayes Classifier - User 11-15

	Users														
	16			17			18			19			20		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.17	0.16	0.12	0.08	0.08	0.11	0.08	0.05	0.08	0.09	0.09	0.19	0.10	0.13	0.15
2	0.20	0.14	0.11	0.10	0.11	0.13	0.13	0.10	0.15	0.08	0.11	0.12	0.14	0.15	0.12
3	0.16	0.09	0.09	0.08	0.06	0.08	0.08	0.05	0.06	0.10	0.11	0.13	0.13	0.15	0.13
4	0.19	0.14	0.10	0.14	0.14	0.17	0.13	0.09	0.12	0.10	0.10	0.15	0.10	0.13	0.12
5	0.15	0.11	0.07	0.10	0.10	0.13	0.08	0.08	0.10	0.08	0.09	0.16	0.07	0.14	0.13
6	0.15	0.13	0.06	0.10	0.11	0.10	0.07	0.08	0.07	0.09	0.08	0.16	0.15	0.13	0.09
7	0.22	0.15	0.12	0.08	0.09	0.11	0.08	0.09	0.09	0.07	0.10	0.16	0.11	0.12	0.10
8	0.19	0.10	0.09	0.07	0.10	0.09	0.08	0.09	0.07	0.06	0.09	0.17	0.15	0.13	0.12
9	0.16	0.14	0.10	0.11	0.09	0.13	0.09	0.11	0.08	0.06	0.07	0.15	0.14	0.14	0.13
10	0.17	0.13	0.09	0.12	0.11	0.12	0.06	0.07	0.09	0.11	0.10	0.14	0.12	0.17	0.12

Table 36: EER for Random Forest Classifier - User 16-20

	Users														
	21			22			23			24			25		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.17	0.12	0.16	0.10	0.09	0.09	0.11	0.12	0.15	0.16	0.14	0.13	0.08	0.10	0.14
2	0.13	0.09	0.10	0.11	0.13	0.13	0.11	0.12	0.12	0.16	0.09	0.10	0.08	0.11	0.13
3	0.11	0.15	0.17	0.14	0.15	0.12	0.08	0.06	0.10	0.14	0.10	0.09	0.09	0.08	0.12
4	0.14	0.10	0.13	0.14	0.12	0.10	0.12	0.11	0.12	0.13	0.11	0.07	0.09	0.07	0.13
5	0.13	0.11	0.14	0.08	0.12	0.14	0.10	0.10	0.08	0.13	0.10	0.09	0.10	0.10	0.14
6	0.15	0.10	0.12	0.09	0.05	0.09	0.10	0.12	0.13	0.13	0.11	0.11	0.05	0.10	0.12
7	0.14	0.09	0.12	0.07	0.13	0.13	0.10	0.10	0.11	0.16	0.12	0.12	0.09	0.09	0.10
8	0.15	0.10	0.10	0.14	0.15	0.13	0.10	0.11	0.10	0.12	0.10	0.09	0.05	0.10	0.13
9	0.14	0.11	0.13	0.13	0.13	0.14	0.10	0.12	0.13	0.13	0.11	0.08	0.08	0.08	0.11
10	0.13	0.10	0.12	0.12	0.10	0.09	0.12	0.11	0.10	0.13	0.09	0.09	0.11	0.08	0.14

Table 37: EER for Random Forest Classifier- User 21-25

	Users														
	26			27			28			29			30		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.18	0.16	0.13	0.12	0.13	0.12	0.10	0.17	0.15	0.09	0.12	0.14	0.11	0.16	0.10
2	0.19	0.14	0.12	0.10	0.12	0.10	0.14	0.17	0.15	0.10	0.11	0.13	0.14	0.20	0.13
3	0.14	0.16	0.15	0.11	0.14	0.12	0.12	0.17	0.13	0.11	0.13	0.12	0.16	0.16	0.18
4	0.19	0.15	0.12	0.10	0.12	0.08	0.13	0.18	0.15	0.11	0.13	0.12	0.15	0.18	0.15
5	0.19	0.16	0.14	0.10	0.09	0.11	0.11	0.15	0.14	0.11	0.13	0.13	0.12	0.19	0.13
6	0.17	0.16	0.15	0.11	0.11	0.07	0.14	0.17	0.11	0.11	0.12	0.10	0.15	0.19	0.13
7	0.16	0.17	0.14	0.09	0.11	0.09	0.12	0.16	0.16	0.16	0.15	0.14	0.14	0.14	0.09
8	0.19	0.16	0.13	0.13	0.09	0.09	0.16	0.20	0.17	0.13	0.14	0.16	0.16	0.17	0.12
9	0.17	0.17	0.14	0.12	0.14	0.15	0.10	0.16	0.12	0.13	0.12	0.12	0.17	0.17	0.14
10	0.19	0.17	0.13	0.12	0.13	0.14	0.11	0.19	0.15	0.14	0.15	0.11	0.16	0.17	0.13

Table 38: EER for Random Forest Classifier - User 26-30

	Users														
	1			2			3			4			5		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.07	0.10	0.12	0.20	0.21	0.23	0.27	0.19	0.16	0.12	0.18	0.14	0.19	0.14	0.15
2	0.08	0.06	0.08	0.19	0.20	0.21	0.28	0.22	0.17	0.14	0.18	0.15	0.17	0.17	0.20
3	0.06	0.07	0.09	0.19	0.19	0.21	0.28	0.22	0.13	0.13	0.18	0.13	0.16	0.18	0.14
4	0.06	0.07	0.07	0.16	0.16	0.17	0.22	0.19	0.16	0.13	0.14	0.13	0.18	0.17	0.16
5	0.04	0.07	0.09	0.18	0.20	0.19	0.28	0.19	0.15	0.14	0.18	0.14	0.15	0.17	0.14
6	0.05	0.07	0.08	0.17	0.20	0.19	0.26	0.21	0.20	0.16	0.18	0.16	0.14	0.17	0.13
7	0.07	0.09	0.10	0.19	0.22	0.22	0.28	0.22	0.17	0.12	0.17	0.13	0.11	0.17	0.16
8	0.08	0.10	0.11	0.23	0.20	0.25	0.26	0.22	0.17	0.12	0.20	0.15	0.19	0.18	0.20
9	0.10	0.09	0.12	0.20	0.21	0.22	0.21	0.22	0.15	0.15	0.18	0.15	0.16	0.14	0.17
10	0.06	0.08	0.11	0.19	0.21	0.20	0.25	0.22	0.17	0.15	0.15	0.16	0.18	0.19	0.15

Table 39: EER for SVM Classfier - User 1-5

	Users														
	6			7			8			9			10		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.19	0.07	0.11	0.07	0.07	0.08	0.01	0.01	0.01	0.14	0.16	0.17	0.19	0.22	0.17
2	0.17	0.10	0.13	0.07	0.06	0.07	0.01	0.03	0.01	0.13	0.12	0.20	0.20	0.19	0.13
3	0.17	0.08	0.08	0.07	0.06	0.07	0.02	0.04	0.03	0.12	0.12	0.14	0.19	0.19	0.13
4	0.18	0.10	0.11	0.06	0.06	0.06	0.01	0.02	0.01	0.11	0.11	0.16	0.18	0.19	0.14
5	0.17	0.08	0.08	0.04	0.04	0.05	0.02	0.03	0.03	0.08	0.13	0.15	0.19	0.18	0.13
6	0.14	0.07	0.06	0.03	0.04	0.04	0.01	0.02	0.03	0.13	0.11	0.20	0.20	0.19	0.13
7	0.14	0.11	0.12	0.07	0.05	0.08	0.03	0.03	0.03	0.12	0.13	0.20	0.24	0.23	0.18
8	0.17	0.12	0.16	0.10	0.08	0.11	0.03	0.02	0.03	0.13	0.14	0.19	0.20	0.25	0.16
9	0.16	0.12	0.13	0.07	0.09	0.09	0.03	0.03	0.03	0.12	0.11	0.19	0.18	0.21	0.16
10	0.15	0.08	0.08	0.08	0.06	0.09	0.04	0.04	0.03	0.12	0.11	0.15	0.20	0.22	0.16

Table 40: EER for SVM Classifier - User 6-10

	Users														
	11			12			13			14			15		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.18	0.28	0.24	0.09	0.07	0.05	0.18	0.12	0.14	0.20	0.32	0.21	0.03	0.06	0.06
2	0.24	0.28	0.27	0.12	0.08	0.06	0.25	0.13	0.15	0.20	0.23	0.19	0.06	0.07	0.06
3	0.19	0.28	0.25	0.12	0.12	0.07	0.23	0.09	0.17	0.21	0.25	0.20	0.04	0.07	0.06
4	0.21	0.34	0.28	0.11	0.12	0.04	0.22	0.09	0.11	0.22	0.27	0.20	0.06	0.06	0.07
5	0.20	0.31	0.25	0.08	0.09	0.05	0.22	0.13	0.15	0.19	0.27	0.20	0.04	0.04	0.05
6	0.15	0.28	0.22	0.09	0.08	0.06	0.24	0.13	0.15	0.19	0.25	0.19	0.07	0.07	0.07
7	0.22	0.28	0.26	0.13	0.13	0.09	0.26	0.14	0.16	0.21	0.33	0.22	0.04	0.06	0.08
8	0.21	0.28	0.23	0.11	0.13	0.07	0.25	0.11	0.13	0.21	0.25	0.15	0.03	0.04	0.06
9	0.21	0.27	0.25	0.10	0.08	0.09	0.23	0.12	0.13	0.21	0.24	0.18	0.03	0.03	0.04
10	0.18	0.28	0.23	0.10	0.09	0.06	0.20	0.12	0.14	0.18	0.26	0.18	0.04	0.04	0.06

Table 41: EER for SVM Classifier - User 11-15

	Users														
	16			17			18			19			20		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.25	0.18	0.14	0.16	0.13	0.15	0.12	0.20	0.20	0.17	0.17	0.20	0.21	0.23	0.16
2	0.28	0.12	0.11	0.17	0.17	0.17	0.18	0.24	0.24	0.18	0.23	0.24	0.21	0.23	0.18
3	0.27	0.16	0.12	0.16	0.13	0.13	0.16	0.17	0.16	0.21	0.16	0.25	0.23	0.23	0.19
4	0.27	0.19	0.15	0.20	0.21	0.18	0.22	0.22	0.18	0.22	0.21	0.25	0.21	0.25	0.19
5	0.27	0.15	0.09	0.18	0.16	0.17	0.16	0.19	0.19	0.19	0.17	0.24	0.20	0.23	0.17
6	0.24	0.13	0.09	0.15	0.14	0.11	0.12	0.16	0.15	0.19	0.17	0.23	0.20	0.23	0.18
7	0.25	0.18	0.12	0.16	0.15	0.12	0.13	0.21	0.18	0.20	0.18	0.25	0.18	0.21	0.18
8	0.24	0.13	0.09	0.13	0.12	0.13	0.13	0.19	0.18	0.20	0.18	0.24	0.21	0.22	0.16
9	0.23	0.15	0.11	0.15	0.16	0.16	0.17	0.21	0.18	0.18	0.19	0.21	0.22	0.22	0.16
10	0.25	0.16	0.11	0.15	0.14	0.15	0.16	0.19	0.19	0.20	0.16	0.21	0.21	0.23	0.19

Table 42: EER for SVM Classifier - User 16-20

	Users														
	21			22			23			24			25		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.24	0.22	0.20	0.17	0.16	0.15	0.18	0.14	0.22	0.26	0.23	0.22	0.12	0.24	0.29
2	0.21	0.20	0.19	0.23	0.22	0.19	0.19	0.13	0.20	0.26	0.25	0.18	0.13	0.24	0.28
3	0.23	0.20	0.21	0.22	0.22	0.19	0.15	0.13	0.24	0.21	0.23	0.18	0.15	0.24	0.27
4	0.21	0.21	0.18	0.22	0.23	0.20	0.20	0.16	0.19	0.24	0.21	0.15	0.14	0.24	0.24
5	0.23	0.18	0.18	0.18	0.20	0.17	0.19	0.17	0.20	0.23	0.18	0.19	0.12	0.23	0.27
6	0.22	0.18	0.20	0.19	0.18	0.17	0.16	0.13	0.20	0.25	0.22	0.18	0.11	0.24	0.26
7	0.22	0.21	0.22	0.20	0.22	0.21	0.16	0.15	0.23	0.27	0.26	0.19	0.14	0.23	0.23
8	0.21	0.16	0.17	0.20	0.20	0.19	0.20	0.15	0.18	0.24	0.23	0.19	0.10	0.25	0.25
9	0.21	0.17	0.19	0.22	0.19	0.18	0.16	0.16	0.21	0.24	0.20	0.16	0.15	0.21	0.23
10	0.21	0.17	0.15	0.19	0.17	0.17	0.17	0.13	0.22	0.22	0.19	0.18	0.14	0.22	0.25

Table 43: EER for SVM Classifier- User 21-25

	Users														
	26			27			28			29			30		
	Block #			Block #			Block #			Block #			Block #		
Cross Validation #	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	0.51	0.34	0.27	0.19	0.26	0.29	0.12	0.23	0.24	0.14	0.18	0.23	0.22	0.29	0.24
2	0.47	0.33	0.24	0.19	0.25	0.26	0.21	0.25	0.27	0.16	0.19	0.19	0.25	0.30	0.29
3	0.51	0.33	0.26	0.19	0.23	0.24	0.18	0.26	0.24	0.14	0.19	0.21	0.27	0.34	0.30
4	0.48	0.29	0.23	0.19	0.24	0.19	0.17	0.25	0.24	0.14	0.17	0.18	0.25	0.32	0.29
5	0.48	0.36	0.27	0.18	0.18	0.19	0.17	0.23	0.21	0.12	0.15	0.21	0.22	0.32	0.29
6	0.48	0.32	0.26	0.16	0.22	0.23	0.16	0.22	0.21	0.16	0.21	0.20	0.23	0.36	0.28
7	0.52	0.36	0.32	0.22	0.26	0.26	0.21	0.31	0.27	0.18	0.19	0.24	0.23	0.35	0.27
8	0.52	0.33	0.24	0.22	0.26	0.26	0.16	0.29	0.30	0.16	0.22	0.22	0.30	0.34	0.27
9	0.51	0.33	0.24	0.21	0.24	0.25	0.15	0.22	0.25	0.16	0.19	0.20	0.29	0.33	0.32
10	0.47	0.33	0.25	0.21	0.26	0.27	0.14	0.26	0.20	0.16	0.18	0.17	0.23	0.30	0.25

Table 44: EER for SVM Classifier - User 26-30

A.2 Determine the Best Train:Test size ratio

It is imperative to periodically update the user model with the latest genuine user data. In section 10 we determine the optimal amount of prior data that must be used to update the user model.

We used the same experimental setup as Section 8.1. However, the genuine train and test set sampling strategy was modified as shown in Fig. 10.1:

1. The genuine data was divided into 12 equal blocks of size *l/12*. This provides the flexibility to increase the train set in relatively smaller steps.

2. For a given Block i as the test set, in iteration j where $6 \leq i \leq 12$, $1 \leq j \leq 5$:

   (a) Generate a user model using Blocks *i-1 to i-j* ***together*** as a train set as shown in Fig. 10.1.

   (b) Calculate EER for this user model.

Test Block #	Train Block #										
	1	2	3	4	5	6	7	8	9	10	11
1	-	-	-	-	-	-	-	-	-	-	-
2	9.09	-	-	-	-	-	-	-	-	-	-
3	12.21	8.18	-	-	-	-	-	-	-	-	-
4	15.77	15.68	12.08	-	-	-	-	-	-	-	-
5	14.24	10.59	11.16	9.75	-	-	-	-	-	-	-
6	16.95	11.49	12.50	8.01	5.52	-	-	-	-	-	-
7	16.53	13.28	15.02	13.98	12.74	7.72	-	-	-	-	-
8	14.86	18.15	16.44	17.32	11.86	11.10	7.47	-	-	-	-
9	13.52	23.87	17.70	16.49	17.60	17.67	19.30	9.43	-	-	-
10	14.18	14.54	17.43	20.94	16.56	15.69	13.33	14.54	6.47	-	-
11	17.57	21.85	17.55	13.81	16.74	15.07	17.63	10.30	15.70	11.33	-
12	22.98	17.00	16.89	15.92	17.06	14.89	14.10	11.91	12.68	12.72	10.3

Table 45: EER performance when blocks are divided in 12 equal blocks and Train : Test dataset ratio =1 : 1

Test Block #	Train Block #									
	1-2	2-3	3-4	4-5	5-6	6-7	7-8	8-9	9-10	10-11
1	-	-	-	-	-	-	-	-	-	-
2	-	-	-	-	-	-	-	-	-	-
3	8.66	-	-	-	-	-	-	-	-	-
4	12.86	6.33	-	-	-	-	-	-	-	-
5	12.13	7.75	9.41	-	-	-	-	-	-	-
6	15.70	8.23	7.21	3.84	-	-	-	-	-	-
7	15.77	12.47	12.51	8.30	6.09	-	-	-	-	-
8	11.58	12.26	11.46	8.87	10.62	6.04	-	-	-	-
9	16.23	18.12	14.52	10.04	14.94	12.43	7.35	-	-	-
10	8.98	10.37	10.73	14.79	17.53	13.57	9.24	6.16	-	-
11	18.28	13.91	13.03	15.92	12.96	12.60	12.74	7.34	8.49	-
12	20.83	13.46	14.24	13.48	16.77	10.38	11.48	8.31	9.77	12.72

Table 46: EER performance when blocks are divided in 12 equal blocks and Train : Test dataset ratio =2 : 1

Test Block #	Train Block #								
	1-3	2-4	3-5	4-6	5-7	6-8	7-9	8-10	9-11
1	-	-	-	-	-	-	-	-	-
2	-	-	-	-	-	-	-	-	-
3	-	-	-	-	-	-	-	-	-
4	7.50	-	-	-	-	-	-	-	-
5	7.42	6.92	-	-	-	-	-	-	-
6	6.90	7.83	5.11	-	-	-	-	-	-
7	12.67	7.82	8.29	6.37	-	-	-	-	-
8	9.87	8.09	7.01	6.99	7.01	-	-	-	-
9	16.34	12.55	9.84	12.89	12.83	8.12	-	-	-
10	8.98	9.05	11.13	11.96	10.61	8.51	6.37	-	-
11	16.52	12.60	11.51	11.67	16.10	9.05	8.18	7.62	-
12	14.11	12.69	13.47	14.08	12.09	10.07	8.62	8.21	7.36

Table 47: EER performance when blocks are divided in 12 equal blocks and Train : Test dataset ratio =3 : 1

Test Block #	Train Block #							
	1-4	2-5	3-6	4-7	5-8	6-9	7-10	8-11
1	-	-	-	-	-	-	-	-
2	-	-	-	-	-	-	-	-
3	-	-	-	-	-	-	-	-
4	-	-	-	-	-	-	-	-
5	5.92	-	-	-	-	-	-	-
6	4.61	6.12	-	-	-	-	-	-
7	10.71	5.79	5.23	-	-	-	-	-
8	7.41	6.58	6.68	4.06	-	-	-	-
9	13.21	11.74	10.48	11.38	7.53	-	-	-
10	6.41	8.55	11.31	7.89	8.35	3.57	-	-
11	12.31	11.19	12.65	11.90	9.78	7.02	6.83	-
12	11.71	10.87	11.58	12.31	10.27	5.87	7.88	7.66

Table 48: EER performance when blocks are divided in 12 equal blocks and Train : Test dataset ratio =4 : 1

Test Block #	Train Block #						
	1-5	2-6	3-7	4-8	5-9	6-10	7-11
1	-	-	-	-	-	-	-
2	-	-	-	-	-	-	-
3	-	-	-	-	-	-	-
4	-	-	-	-	-	-	-
5	-	-	-	-	-	-	-
6	6.71	-	-	-	-	-	-
7	11.74	7.84	-	-	-	-	-
8	9.37	8.52	6.81	-	-	-	-
9	14.14	11.35	8.48	11.29	-	-	-
10	7.81	8.42	9.81	13.16	9.49	-	-
11	15.23	10.26	10.42	12.23	14.17	8.35	-
12	13.10	11.62	11.36	13.21	12.81	9.27	8.13

Table 49: EER performance when blocks are divided in 12 equal blocks and Train : Test dataset ratio =3 : 1

A.3 Publications

A.3.1 Conference Publication

1. **Palaskar, N**.; Syed, Z.; Banerjee, S.; Tang, C., "*Empirical Techniques to Detect and Mitigate the Effects of Irrevocably Evolving User Profiles in Touch-based Authentication Systems,*", IEEE Int'l Symposium on High Assurance Systems Engineering (HASE), 2016.

A.3.2 Under Preparation

1. **Journal Article**: Empirical Techniques to Detect and Mitigate the Effects of Irrevocably Evolving User Profiles in Touch-based Authentication Systems.

2. **Conference Paper**: A Unique Naive Bayes and/or simple fusion of Random Forest and Clustering to improve Mis-Classification rates in Touch Based Authentication Systems.

A.3.3 Project Code Repository

All the codes are written in Python scripts and its is hosted at https://github.com/npalaska

References

[1] P. R. Center, "Cell phone and smartphone ownership demographics," 2014.

[2] Gartner, "Market share analysis: Mobile phones, worldwide, q2 2013," 2013.

[3] F. C. Commission, "Announcement of new initiatives to combat smartphone and data theft," 2012.

[4] "Business insider."

[5] iJailbreak, "ios 5 iphone lockscreen glitch allows you to bypass password protection and explore the phone.app!," 2012.

[6] J. A., "How to bypass android pattern lock screen," 2012.

[7] Z. Syed, J. Helmick, S. Banerjee, and B. Cukic, "Effect of user posture and device size on the performance of touch-based authentication systems," in *High Assurance Systems Engineering (HASE), 2015 IEEE 16th International Symposium on*, pp. 10–17, Jan 2015.

[8] M. Frank, R. Biedert, E. Ma, I. Martinovic, and D. Song, "Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication," *Information Forensics and Security, IEEE Transactions on*, vol. 8, pp. 136–148, Jan 2013.

[9] 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 SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, (New York, NY, USA), pp. 987–996, ACM, 2012.

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

[11] T. Feng, Z. Liu, K. A. Kwon, W. Shi, B. Carbunar, Y. Jiang, and N. Nguyen, "Continuous mobile authentication using touchscreen gestures," in *Homeland Security (HST), 2012 IEEE Conference on Technologies for*, pp. 451–456, Nov 2012.

[12] H. Xu, Y. Zhou, and M. R. Lyu, "Towards continuous and passive authentication via touch biometrics: An experimental study on smartphones," in *Symposium On Usable Privacy and Security, SOUPS*, vol. 14, pp. 187–198, 2014.

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

[14] C. Shen, Y. Zhang, Z. Cai, T. Yu, and X. Guan, "Touch-interaction behavior for continuous user authentication on smartphones," in *Biometrics (ICB), 2015 International Conference on*, pp. 157–162, May 2015.

[15] "Vision mobile. developer economics q3 2013 state of the developer nation," 2013.

[16] "Gartner. market share analysis: Mobile phones, worldwide, q2 2013, 2013," 2013.

[17] "Android sdk. get the android sdk," 2013.

[18] Z. Syed, S. Banerjee, Q. Cheng, and B. Cukic, "Effects of user habituation in keystroke dynamics on password security policy," in *High-Assurance Systems Engineering (HASE), 2011 IEEE 13th International Symposium on*, pp. 352–359, Nov 2011.

[19] T. Boult, "Robust distance measures for face-recognition supporting revocable biometric tokens," in *Automatic Face and Gesture Recognition, 2006. FGR 2006. 7th International Conference on*, pp. 560–566, April 2006.

[20] V. Perlibakas, "Distance measures for pca-based face recognition," *Pattern Recognition Letters*, vol. 25, no. 6, pp. 711 – 724, 2004.

[21] Z. Syed, S. Banerjee, Q. Cheng, and B. Cukic, "Effects of user habituation in keystroke dynamics on password security policy," in *High-Assurance Systems Engineering (HASE), 2011 IEEE 13th International Symposium on*, pp. 352–359, Nov 2011.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[23] N. Kato, M. Suzuki, S. Omachi, H. Aso, and Y. Nemoto, "A handwritten character recognition system using directional element feature and asymmetric mahalanobis distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, pp. 258–262, Mar 1999.

[24] L. P. Dinu and R. T. Ionescu, "A rank-based approach of cosine similarity with applications in automatic classification," in *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2012 14th International Symposium on*, pp. 260–264, Sept 2012.

[25] A. Strehl, J. Ghosh, and R. Mooney, "Impact of similarity measures on web-page clustering," in *Workshop on Artificial Intelligence for Web Search (AAAI 2000)*, pp. 58–64, 2000.

[26] B. Nie, J. Du, H. Liu, G. Xu, Z. Wang, Y. He, and B. Li, “Crowds’ classification using hierarchical cluster, rough sets, principal component analysis and its combination,” in *Computer Science-Technology and Applications, 2009. IFCSTA ’09. International Forum on*, vol. 1, pp. 287–290, Dec 2009.

[27] N. Palaskar, Z. Syed, S. Banerjee, and C. Tang, “Empirical techniques to detect and mitigate the effects of irrevocably evolving user profiles in touch-based authentication systems,” in *2016 IEEE 17th International Symposium on High Assurance Systems Engineering (HASE)*, pp. 9–16, Jan 2016.

[28] Z. Syed, J. Helmick, S. Banerjee, and B. Cukic, “Effect of user posture and device size on the performance of touch-based authentication systems,” in *High Assurance Systems Engineering (HASE), 2015 IEEE 16th International Symposium on*, pp. 10–17, Jan 2015.