Keystroke-based User Identification on Smart Phones

Saira Zahid[1], Muhammad Shahzad[1], Syed Ali Khayam[1,2], Muddassar Farooq[1]

[1]Next Generation Intelligent Networks Research Center (nexGIN RC)
National University of Computer and Emerging Sciences (FAST–NUCES)
Islamabad 44000, Pakistan
[2]School of Electrical Engineering and Computer Sciences (SEECS)
National University of Sciences and Technology (NUST)
Islamabad 44000, Pakistan
{saira.zahid, muhammad.shahzad, muddassar.farooq}@nexginrc.org, ali.khayam@seecs.edu.pk

Abstract. Smart phones are now being used to store users' identities and sensitive information/data. Therefore, it is important to authenticate legitimate users of a smart phone and to block imposters. In this paper, we demonstrate that keystroke dynamics of a smart phone user can be translated into a viable feature set for accurate user identification. To this end, we collect and analyze keystroke data of 25 diverse smart phone users. Based on this analysis, we select six distinguishing keystroke features that can be used for user identification. We show that these keystroke features for different users are diffused and therefore a fuzzy classifier is well-suited to cluster and classify them. We then optimize the front-end fuzzy classifier using Particle Swarm Optimizer (PSO) and Genetic Algorithm (GA) as back-end dynamic optimizers to adapt to variations in usage patterns. Finally, we provide a novel keystroke dynamics based PIN verification mode to ensure information security on smart phones. The results of our experiments show that the proposed user identification system has an average error rate of 2% after the detection mode and the error rate of rejecting legitimate users is dropped to zero after the PIN verification mode. We also compare error rates (in terms of detecting both legitimate users and imposters) of our proposed classifier with 5 existing state-of-the-art techniques for user identification on desktop computers. Our results show that the proposed technique consistently and considerably outperforms existing schemes.

1 Introduction

Smart phones[1] are pervasively and ubiquitously integrating into our home and work environments. In particular, due to the enhanced capabilities available on contemporary smart phones, users (in addition to personal and professional contacts' information) now store sensitive information – emails, credit card numbers, passwords, corporate secrets, etc. – on mobile phones which makes them an attractive target for imposters [10]. Stolen mobile phones can be used for identity theft which can then be exploited for malicious and/or unlawful activities. Various surveys conducted recently show that

[1] We use the terms smart phone and mobile phone interchangeably throughout this paper because our proposed system, with the support of OS vendors, can be deployed on both types of phones.

in case of mobile phone theft, users (instead of being worried about the cost of the stolen phone) are becoming more concerned with misuse of information and services on the stolen phones [1]. Therefore, it is important to develop intelligent user identification schemes for mobile phones.

Despite its need and importance, user identification on mobile phones has received little attention in research literature. User identification systems for mobile phones are usually based on secret PIN numbers [28]. These identification techniques are borrowed from desktop computers' domain and have not been very effective on mobile phones [7],[28]. For instance, freely available tools empower intruders, who have physical access to the Subscriber's Identity Module (SIM) and know the Personal Identification Number (PIN), to reverse engineer the International Mobile Subscriber Identity (IMSI) and the secret key of GSM mobile phone users [20]. Similarly, token-based authentication schemes developed for desktops are not suitable for mobile phones because[10]: (1) they cannot be efficiently implemented on resource-constrained devices [10], and (2) loss of a token in essence means loss of the device [26]. Biometric hardware for mobile phones are now being developed to overcome the shortcomings of token-based authentication [26]. A common drawback of these authentication paradigms is that they perform one-time identity check at the beginning of a session that allows imposters to access the smart phones once a session is logged in.

In this paper, we propose a robust approach to identify a legitimate user of a mobile phone by learning his/her "in-session" keystroke dynamics. The scheme requires no additional hardware or software resources and is user-friendly as it requires minimum user intervention after installation. While keystroke-based user identification was actively pursued in the desktop computer's domain [27],[18],[15],[19], its suitability for mobile phones has not been explored, except the preliminary work reported in [8],[16]. To use keystroke information for user identification, we collect and analyze keystroke data of 25 diverse mobile phone users including researchers, students and professionals from varying age groups. Based on our analysis, we select six distinguishing keystroke features that can be used for user identification. Two of these features – *Key hold time* (how long a key is pressed) and *Error Rate* (number of times backspace is pressed) – are borrowed from the desktop domain. We also customize a set of four features to capture the unique switching behavior across multiplexed mobile phone keys using: (1) *Horizontal Digraph*: time to switch between horizontally adjacent keys, (2) *Vertical Digraph*: time to switch between vertically adjacent keys, (3) *Non-Adjacent Horizontal Digraph*: time to switch between non-adjacent horizontal keys, and (4) *Non-Adjacent Vertical Digraph*: time to switch between non-adjacent vertical keys.

We reveal that, while these keystroke features differ across users, leveraging them for accurate user identification on mobile phones is significantly more challenging than on a desktop computer because on a majority of contemporary mobile phones: (1) different keys are multiplexed on a small keypad, (2) the variable and discontinuous keystroke usage of a mobile phone user results in a highly diffused (overlapping) and time-varying feature space that makes it difficult to cluster and classify different users, and (3) an imposter can get access to a mobile phone at anytime so techniques that rely on static, application-specific or keyword-specific authentication are not feasible. These challenges are aggravated by the fact that most of the mobile OS vendors do not provide

any mechanism/API for key logging. In view of these challenges, we set two accuracy objectives for the proposed technique: (1) correctly identify imposters and legitimate users using keystroke dynamics[2], and (2) identify an imposter within a small number of key hits to ensure timely information security. In addition to being accurate, an effective user authentication scheme for mobile phones must: (1) be able to continuously adapt to varying usage patterns of a phone user, (2) utilize a classifier that provides high classification accuracy for a diffused features space, and (3) have low-complexity so that it can be deployed on resource-constrained mobile phone sets.

To meet the above requirements, we propose a keystroke-based user identification system which operates in three sequential modes.

Learning Mode. In this mode, we train a fuzzy classifier which maps the diffused feature space of a mobile phone user to his/her profile. Moreover, it utilizes a hybrid of bio-inspired optimizers Particle Swarm Optimization (PSO) [17] and Genetic Algorithm (GA) [12] at the back-end for continuous evolution of the fuzzy system in order to cope with the varying usage pattern of the user[3].

Imposter Detection Mode. In this mode, the trained classifier is used to classify real-time keystroke measurements to classify a user as legitimate or imposter.

Verification Mode. This mode is only invoked if a user is potentially identified as an imposter in the detection mode or the user wants to transmit documents from a mobile phone. In the verification mode, the potential imposter is asked to type a remembered 8-character PIN of the legitimate user. The system then uses the keystroke dynamics model to analyze the typing behavior of the potential imposter. This mode makes it difficult for an imposter to have illegitimate access by hacking the PIN only; therefore, it serves as the last line of defence after an imposter has breached all other security layers.

Performance evaluation on the collected dataset shows that the proposed hybrid PSO-GA based fuzzy classifier, when trained using a mere 250 keystrokes, achieves an average error rate of approximately 2% after the detection mode and close to zero FRR after the verification mode. We compare the accuracy of our system with five other state-of-the-art keystroke-based user identification techniques and show that our proposed system provides significantly better accuracy in detecting legitimate users and imposters.

The rest of the paper is organized as follows. Section 2 briefly describes the related work. We discuss our dataset in Section 3 and explain feature selection along with a study of existing desktop schemes on these features in Section 4. In Section 5, we first investigate the feasibility of existing desktop classification schemes for mobile phones, and then we discuss the architecture of our proposed user identification system. In Sections 6, we analyze the performance of our proposed system for varying parameters. The limitations of the proposed system and potential countermeasures to overcome these limitations are detailed in Section 7. Finally, we conclude our paper with an outlook to our future research.

[2] Throughout the paper we define accuracy in terms of error in detecting an imposter – False Alarm Rate (FAR) – and error in detecting a legitimate user – False Rejection Rate (FRR).

[3] PSO and GAs are well-known for providing efficient and online solutions to dynamic and time-varying optimization problems [11],[5].

2 Related Work

The idea of using keystroke dynamics for user authentication is not new as there have been a number of prior studies in this area for desktop computers. Most of these prior studies have focused on static or context-independent dynamic analysis using the inter-keystroke latency timing method for desktop keyboards only. From the earliest studies in 1980 [6], the focus has been on the analysis of delays between two consecutive keystrokes – also called digraph. Later studies [15],[21] further enhanced the work, identifying additional statistical analysis methods that provided more reliable results. This section briefly summarizes some of the prominent research on keystroke based user identification.

One of the earlier works in the area of keystroke dynamics was accomplished by Umphress and Williams [27] in 1985. They used digraphs as the underlying keystroke biometric. However, they were only able to achieve a FAR of 6%. In 1987, Williams and Leggett [18] further extended the work by: (1) increasing the number of users in the study, (2) reducing experimental variables, and (3) discarding inappropriate digraphs according to latency and frequency. They managed to reduce the FAR to 5%.

Another extension of the above work was conducted in 1990 by Leggett et al. [19]. While the results of the static procedure of entering a reference and testing profiles achieved the same 5% FAR, they were the first ones to utilize the concept of keystroke dynamics for doing verification in a dynamic environment. They were able to achieve FAR of 12.8% and FRR of 11.1% using statistical theory. In a study by Joyce and Gupta [15], the username was compared to the particular profile for that user. The login had four components – username, password, first name and last name. Digraphs were then calculated and basic statistical method of means, variances and standard deviations were used to determine a match. Using this method, the FAR was just 0.25% but the FRR was 16.67%. Bleha et al. [4], in 1990, used a different statistical method: the Bayes classification algorithm. The verification system gave results of 8.1% for FRR and 2.8% for the FAR. Regarding features' set, no significant additions occurred until 1997 when Obaidat and Sadoun [22] introduced key hold times as another feature of interest. Currently, the most common and widely-known application that uses keystroke dynamics technology is BioPassword [13]. To the best of our knowledge, BioPassword is the only product available in the market that has relatively wide usage.

These studies, however, have focused their research only on desktop computers. Except for [8],[16], no work has been done on user identification using keystroke dynamics on mobile phones. Clarke et al. [8] have used neural networks to classify a user using key hold time and inter-key latency or digraph. They performed three sets of experiments on mobile phone emulators: (1) on PIN verification, (2) on specific text, and (3) on phone number entry. They achieved FARs of 3%, 15% and 18% respectively for these three experiments, however FRRs were 40%, 28% and 29%, respectively.

3 Data Acquisition

As a first step towards developing a robust mobile phone user identification system, we developed an application to log mobile keystroke data. We decided to develop the application for Symbian OS 3rd Edition because: (1) it had a relatively large customer base

Table 1. Feature Table of 25 Mobile Users of this Study (c_v is coefficient of variation)

Users	Nokia model model	Social status status	Total key hit key hit profiles	Total key key hits	Key Hold Time Time		Horizontal Digraph Digraph		Vertical Digraph Digraph		Non-adjacent Vertical Digraph		Non-adjacent Horizontal Digraph		Error (%) (%)
					μ (ms)	c_v	μ (sec)	c_v	μ (sec)	c_v	μ (sec)	c_v	μ (sec)	c_v	
u1	N73	manager	17	4113	61.1	0.08	0.12	2.08	0.15	1.80	0.19	1.21	0.05	1.11	1.25
u2	N95	researcher	48	11901	83.1	0.01	0.32	0.87	0.33	0.87	0.24	1.79	0.09	1.23	3.07
u3	N81	student	12	2939	81.4	0.04	0.35	0.31	0.42	0.45	0.39	1.43	0.07	1.36	2.63
u4	6650	engineer	21	5011	131	0.02	0.26	0.88	0.34	0.38	0.18	1.77	0.13	0.82	0.93
u5	6120	teenager	34	8255	103	0.08	0.21	2.09	0.12	2.33	0.43	1.25	0.11	1.31	2.38
u6	N79	businessman	20	4919	25.3	0.16	0.32	1.68	0.17	3.05	0.53	0.45	0.14	1.23	8.47
u7	N73	student	30	7283	45.1	0.21	0.11	5.72	0.23	2.82	0.12	5.33	0.17	1.11	8.59
u8	6124	manager	33	8209	95.9	0.03	0.12	1.50	0.11	3.09	0.53	0.43	0.09	0.79	5.79
u9	N95	engineer	37	9211	83.2	0.04	0.31	2.45	0.61	0.39	0.32	1.03	0.05	1.09	4.10
u10	N82	advertiser	53	13193	76.6	0.04	0.21	0.62	0.42	0.59	0.52	1.46	0.08	1.23	6.55
u11	E51	student	27	6501	32.1	0.12	0.33	0.87	0.56	1.33	0.32	0.68	0.04	1.01	8.14
u12	6120	researcher	37	9028	67.3	0.03	0.18	1.66	0.82	0.42	0.54	0.62	0.11	0.92	7.85
u13	N81	student	41	10001	11.3	0.16	0.22	1.59	0.28	0.82	0.75	0.88	0.14	1.22	6.02
u14	N77	student	53	13022	35.6	0.07	0.24	1.37	0.48	1.56	0.35	1.22	0.19	0.86	2.47
u15	E65	engineer	55	13713	61.5	0.05	0.33	0.93	0.41	0.95	0.12	1.83	0.16	1.31	3.22
u16	N76	senior citizen	19	4744	15.9	0.13	0.71	0.33	0.86	0.75	0.43	1.55	0.13	0.87	2.37
u17	N81	manager	12	2900	42.1	0.07	0.54	1.40	0.28	1.25	0.65	0.35	0.08	2.01	11.2
u18	6121	engineer	48	11793	57.6	0.07	0.18	1.72	0.48	1.56	0.45	1.20	0.13	1.71	1.35
u19	6120	student	17	4011	21.7	0.01	0.21	3.38	0.19	2.94	0.19	1.21	0.07	1.52	1.21
u20	N73	researcher	47	11529	33.4	0.15	0.15	1.53	0.32	0.81	0.43	1.23	0.15	1.08	2.53
u21	N81	researcher	20	4992	76.3	0.02	0.66	0.51	0.64	0.67	0.64	0.35	0.15	0.57	1.33
u22	N73	director	27	6721	23.3	0.12	0.21	1.04	0.24	3.25	0.24	2.79	0.11	1.14	6.23
u23	N95	student	41	10132	68.2	0.08	0.63	0.61	0.53	0.66	0.15	2.26	0.14	1.16	8.43
u24	N81	researcher	39	9531	79.7	0.05	0.44	0.72	0.31	0.74	0.32	1.71	0.09	1.11	3.11
u25	6120	teenager	33	8193	17.5	0.38	0.25	0.64	0.45	1.66	0.74	0.31	0.07	1.23	2.31

in our social network, and (2) it provides developers with Application Programming Interfaces (APIs) to capture key events[4]. The application runs in the background so that a user can continue using his/her mobile phone uninterruptedly. All keys pressed by a user are logged along with the press/release times of the keys [5]. In addition to the regular keys, we also log left soft key, right soft key, left arrow key, right arrow key, up arrow key, down arrow key, joystick key, menu key, call dial key, call end key, back space key, camera key, volume up key, volume down key, * key, and # key. A text file containing all logged key events is stored in the phone memory and is periodically uploaded to our development server. The application was digitally signed by Symbian testing team [`www.symbiansigned.org`] before deployment.

Despite security and privacy concerns shown by most volunteers, we were able to convince 25 mobile phone users to volunteer for this study. The subjects of our study have different socioeconomic backgrounds (see Table 1) that provides good diversity in our dataset; we have teenagers, corporate executives, researchers, students, software developers and even senior citizens in our list of volunteers.

Another distinguishing feature of this dataset is that it is not a one prototype model dataset and has been collected from a diverse set of Nokia mobile phones. We have N-series, E-series and 6xxx series mobile phones all of which have multiplexed keypad.

[4] We have used Symbian C++ instead of JAVA because Symbian smart phones do not allow JAVA midlets to capture the background key events [2].

[5] We used Active Objects to realize this functionality [3].

This diversity in phone sets is important to ensure that our system design and evaluation spans across a wide range of modern mobile phones.

For all the analysis provided later in the paper, we use a dataset of 25 users spanning over 7 days. We quantify the keystrokes into a profile of 250 key-hits each, which we call a 'Key hit profile'. A justification for this profile size is given in Section 6. Table 1 shows that people from different walks of life have different number of key hit profiles in accordance with their social status. We observe that students, teenagers and professionals use keyboard of mobile phones aggressively while senior citizens and managers use keyboard of mobile phone less frequently. For instance, users u10, u14 and u15 have more than 50 key hit profiles while users u1, u3, u16, u17 and u19 make less than 20 key hit profiles over the same period of 7 days.

After successfully collecting the dataset, we started the next phase of our research – systematically analyzing our raw data to extract useful features for user identification. We observed that some people tend to type faster with less errors as compared to others, while some others type very slowly which is uniquely linked to their social status and age as shown in Table 1. Based on this preliminary analysis, we observed that if we can identify a keystroke dynamics feature set that covers all aspects of a persons' unique typing pattern, we can actually identify the mobile phone user. Therefore, we extracted 6 features to correctly identify a user – 2 of these features have been borrowed from the desktop world while the remaining 4 are customized for mobile phones' multiplexed keypads. A detailed discussion of this feature set is provided in the next section.

4 Feature Selection and Study of Desktop-based Schemes

In this section, we first analyze three well-known features that have been used for user identification on desktop/laptop computers. We then customize these features for mobile phones. Finally, we evaluate the accuracies of existing keystroke-based user identification schemes in identifying mobile users.

4.1 Feature Selection

After collecting data of the mobile phone users, we extracted three features from this data – key hold time, digraph, and error rate. These features have been used for keystroke-based user identification for desktop/laptop computers [18],[15]. However, their usability to identify a legitimate user on mobile phones has not been explored before. These features are defined as:

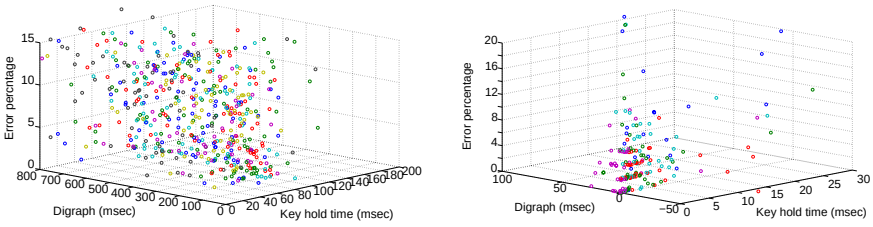
Key hold time. The time difference between pressing a key and releasing it;

Digraph time. The time difference between releasing one key and pressing the next one; and

Error rate. The number of times backspace key is pressed.

We observed that identifying a user based on these three features is less challenging on desktops because of a relatively distinguished feature vector for each user. As an example, we installed a key-logging application on the laptops of 6 users for a period of 5 days. The plot of these three features extracted from the desktop key logging data

of 6 users is shown in Figure 1(b). It can be observed that the features' set on desktops is well segregated and poses a relatively simple classification problem. However, once we extracted the same three features from the mobile phone data of 25 users, their feature vectors are extremely diffused as shown in Figure 1(a). Keystroke-based user identification problem is more challenging on mobile phones because they generally have multiplexed keys in a 4×3 matrix. In order to make the data less diffused, we split



(a) Plot showing high diffusion in mobile features

(b) Plot showing less diffusion in desktop features

Fig. 1. Plot of features' variation for mobile and desktop

the feature "digraph" into four types of digraphs as follows:
Horizontal Digraph (D_h^a). This is the time elapsed between releasing a key and pressing the adjacent key in the horizontal line of keys, e.g. the time between key 1 and key 2, key 5 and key 6, key 0 and key * etc.;
Vertical Digraph (D_v^a). This is the time elapsed between releasing a key and pressing the adjacent key in the vertical line of keys, e.g. the time between key 1 and key 4, key 5 and key 8, key # and key 9 etc.; and
Non-adjacent Horizontal Digraph (D_h^{na}). This is the time elapsed between releasing a key and pressing the next in the horizontal line such that the keys are separated by another key, e.g. time between key 1 and key 3, key 4 and key 6, key * and key # etc.
Non-adjacent Vertical Digraph (D_v^{na}). This is the time elapsed between releasing a key and pressing the next in the vertical line such that the keys are separated by another key, e.g. the time between key 1 and key 7, key 0 and key 5, key 3 and key 9 etc.

Once we extracted these features, we calculated the coefficient of variation for each feature to determine variation and randomness in the features' data. From Table 1, we can observe that the coefficient of variation of the key hold time feature for 25 different users is very small (order of 10^{-2}) which highlights that users normally press keys for approximately the same length of time. However, this parameter is significantly higher for digraph times. The coefficient of variation of more than 1 shows large randomness in the data. Therefore, in order to correctly classify a user based on this collective feature set, we need a classifier that can identify classification boundaries for this highly varying data which results in diffused usage patterns for different users.

4.2 Accuracy Evaluation of Existing Techniques

As a next logical step, we investigate the accuracy of existing classification schemes, developed for desktop computers, on the mobile phones dataset. To this end, we evaluate five prominent classifiers proposed in [25],[23],[14],[29],[9],[24]. These classifiers are quite diverse. Naive Bayes [25] is a probabilistic classifier; while Back Propagation Neural Network (BPNN) [23] and Radial Basis Function Network (RBFN) [14] belong to the category of neural networks. In comparison, Kstar [9] is a statistical classifier and J48 [24] is a decision tree classifier. In order to remove any implementation related bias, we have performed our experiments in WEKA [29].

Ideally, we need a classifier that classifies a user as legitimate or imposter with 100% accuracy. In our current accuracy evaluation setup, the errors are of two types: (1) *False Acceptance Rate (FAR)* is defined as the probability that an imposter is classified as a legitimate user, and (2) *False Rejection Rate (FRR)* is defined as the probability that a legitimate user is classified as an imposter.

The results of our experiments are tabulated in Table 4. We can see that the existing classifiers provide an FAR of 30-40% which is not acceptable. Similarly, FRR of most of the classifiers is approximately 30% or more and this again confirms that their accuracies are not acceptable for real-world deployments because such a high FRR will simply frustrate legitimate users.

Table 2. A comparative study of techniques on the basis of key hold time, digraph and error percentage

	Naive Bayes		**BPNN**		**RBFN**		**Kstar**		**J48**	
Users	FAR	FRR	FAR	FRR	FAR	FRR	FAR	FRR	FAR	FRR
u1	51.1	**6.31**	56.2	12.4	33.2	9.31	**13.2**	22.1	44.3	39.2
u2	32.4	17.9	31.5	58.4	28.4	**11.9**	**11.2**	38.4	31.2	67.4
u3	42.1	11.6	45.3	19.6	**22.5**	**11.2**	34.2	11.5	44.3	26.4
u4	56.9	**7.28**	31.4	11.3	58.3	12.4	49.8	19.5	**21.3**	32.4
u5	33.1	36.6	44.2	24.5	45.2	**21.4**	**32.1**	31.3	33.4	25.8
u6	44.6	17.8	53.4	20.5	48.9	**11.5**	37.6	18.4	**24.6**	32.4
u7	40.2	**21.3**	45.6	18.9	36.7	21.4	43.1	33.4	**21.2**	24.8
u8	29.8	58.2	37.5	23.6	68.9	**18.9**	44.6	26.5	**24.6**	32.1
u9	27.3	62.7	40.4	44.2	44.1	31.3	**24.6**	**21.3**	44.3	38.5
u10	**24.5**	63.2	36.7	72.4	30.9	**43.2**	27.6	53.2	42.1	79.6
u11	41.6	18.9	42.1	19.6	23.5	**12.3**	23.2	31.2	**21.7**	34.5
u12	33.1	37.3	42.1	**28.5**	33.2	33.5	**31.2**	43.2	43.8	73.5
u13	32.1	53.4	42.6	61.3	**19.5**	54.3	24.6	**34.2**	**19.5**	75.2
u14	**22.5**	63.5	28.9	23.1	33.5	31.3	26.6	**21.5**	43.5	39.3
u15	21.5	38.8	33.4	78.9	20.4	59.6	21.3	**31.2**	**12.4**	81.2
u16	43.1	35.8	56.7	19.6	67.5	**15.6**	52.3	33.4	**21.7**	30.4
u17	49.6	**11.9**	61.3	12.4	39.4	13.7	**34.6**	28.5	41.2	23.2
u18	29.8	63.4	31.2	73.2	34.5	35.6	28.7	39.2	**23.5**	**34.6**
u19	52.4	**4.16**	64.7	13.2	**37.4**	15.8	38.6	30.5	47.7	32.4
u20	29.8	**13.2**	**22.5**	38.6	33.3	66.7	27.9	35.4	33.2	28.3
u21	39.8	19.7	53.7	**19.2**	**28.5**	19.8	32.3	31.2	31.4	25.3
u22	39.1	35.6	39.6	44.2	22.1	33.1	**19.4**	31.3	**19.4**	**28.5**
u23	30.9	**23.3**	28.6	26.7	21.8	32.1	**12.5**	43.2	23.4	55.3
u24	33.5	**21.3**	41.4	21.4	31.2	24.1	18.4	22.3	**16.4**	39.2
u25	42.5	19.7	29.6	19.6	38.2	22.4	21.3	38.2	**19.4**	**18.3**
average	36.9	30.5	41.6	32.2	36.0	**26.5**	**29.2**	30.8	29.9	40.7
standard deviation	**9.50**	19.9	11.1	20.8	13.5	15.7	11.0	**9.22**	10.9	19.2

4.3 Discussion

The main reason for poor FAR and FRR performance of these classifiers is that they are unable to cope with the variation in the feature set that was highlighted by the coefficient of variation parameter in the previous section. Thus an important outcome of this pilot study is that we need to design a classifier for our user identification and authentication system that should meet the following requirements: (1) it should provide low ($< 5\%$) FAR, (2) it should also provide low ($< 5\%$) FRR, (3) it must have small detection time to correctly classify a user, (4) the system must be deployable on real mobile phones, (5) it should continuously adapt to the variation in the feature set, and (6) it should have low run-time complexity.

Requirement (1) ensures that an imposter does not go undetected. Once this requirement is combined with requirement (3), we reduce the identification delay on a mobile phone. Requirement (2) is important because if our system starts rejecting the legitimate users, it will lead to their frustration and annoyance and the system will loose its usability appeal. Finally requirement (6) is important because a highly complex system can not be deployed on resource constrained mobile phones.

The following section develops a classifier that can simultaneously meet all of the above requirements.

5 A Tri-Mode System for Mobile User Identification

Based on the results of the last section, we propose a tri-mode system for mobile user identification. To simultaneously cater for the requirements set above, the system operates in three sequential modes.

Learning Mode. This mode can be sub-divided into two modes: initial (static) learning and continuous (dynamic) learning. In the static learning phase, a keystroke profile of a user is collected and a feed-forward classifier is trained on this profile. The dynamic learning phase executes continuously to track and learn changes in the user's behavior. These changes are fed back into the feed-forward detector to allow it to adapt to variations in user behavior.

Detection Mode. During the detection mode, the classifier tuned by the learning module is used to differentiate between legitimate users and imposters. If the detector raises an alarm during this mode, the system moves to the verification mode.

Verification Mode. In the verification mode, a user is asked to type a remembered 8-character PIN. In the verification mode, we not only compare the typed characters with the stored PIN but also match how the pin has been typed. In the worst case, when an imposter already knows the PIN the imposter would still have to enter the PIN using the legitimate user's keystroke dynamics. This mode acts as a final line of defence against an imposter who has successfully breached every other protection layer.

In subsequent sections, we explain the algorithms used in each of the modes described above.

5.1 Algorithms in Learning and Detection Modes

Previous results showed that, due to the variation in the feature-set of different users, standard machine learning classifiers cannot provide acceptable error rates for the present

problem of keystroke-based mobile user identification. Specifically, variation in the features' set results in a diffused dataset and consequently it is not possible to assign crisp classification boundaries to different users. A study of existing classifiers reveals that classifiers based upon *fuzzy logic* [30] are well-suited for this problem. Fuzzy classifiers can provide acceptable accuracies on diffused datasets because they assign a given data point a degree of membership to all available classes. The primary task of fuzzy classification is to determine the boundaries of the decision regions based on the training datapoints. Once the class-labeled decision regions in the feature space are determined, classification of an unknown point is achieved by simply identifying the region in which the unknown point resides. Since fuzzy logic assigns each data point a degree of membership to different decision regions instead of a single association to a decision region, we expect a fuzzy classifier to provide an accurate and efficient learning mechanism for the diffused mobile phone feature-set. The remainder of this section develops and evaluates a fuzzy classifier for mobile user classification.

Initial Learning using a Feed-Forward Fuzzy Classifier We are working on a two-class classification problem as we need to distinguish between a legitimate user and an imposter. A fuzzy system is based on a database, rulebase, and a fuzzy inference system. The database is composed of linguistic variables, fuzzy partitions, and membership functions. We now describe our fuzzy clustering algorithm and then evaluate its accuracy on the mobile keystrokes dataset.

In order to determine an initial rule base for fuzzy system, we define the centroid of a cluster in the form of $(x_1, x_2, \ldots, x_z)$, where x_1, x_2, $\ldots$, x_z are the values of the first, second, $\ldots$, z^{th} feature, respectively, where z is the dimension of the feature vector. It is mentioned earlier that we use $z = 6$ features. For a given data point, we search its value in the corresponding fuzzy sets, determine its degree of membership to each fuzzy partition and then assign the point to the partition with the maximum degree of membership. To determine the consequent of a rule, we find the density of the cluster of the centroid for which we are defining an antecedent of the rule. If a cluster has $high$, $medium$ or low density then the output belongs to the fuzzy partitions $high$, $medium$ or low, respectively, in the consequent of the rule. We repeat this procedure for all training data points to define a rule-base using the centroids of all the clusters.

To give a preliminary indication of the accuracy of the first phase of our proposed system, the FAR and FRR values of the fuzzy classifier are shown in Table 3. FAR and FRR of approximately 18.6% and 19.0%, respectively – much better compared with existing classifiers in Table 4 – are still far from acceptable. These accuracy results do not meet the requirements that we have set for our system. In our performance evaluation, we observed that the main accuracy limiting factor for the fuzzy classifier was the dynamically changing keystroke behavior of mobile users. Thus the performance of the feed-forward fuzzy classifier can be improved if we use an online dynamic optimizer that can dynamically track and feedback the changing feature trends into the fuzzy system.

Prior work has shown that Particle Swarm Optimizers (PSO) and Genetic Algorithms (GAs) have the capability to adapt with changes in data [11],[5]. Therefore, in

Table 3. A comparative study of the feasible techniques

	RBFN		Fuzzy		PSO-Fuzzy		GA-Fuzzy		PSO-GA Fuzzy	
Users	FAR	FRR	FAR	FRR	FAR	FRR	FAR	FRR	FAR	FRR
u1	33.2	9.31	18.1	19.3	8.34	7.55	8.32	8.54	**2.13**	**1.76**
u2	28.4	11.9	17.3	21.6	9.63	6.43	8.73	8.11	**1.61**	**0.82**
u3	22.5	11.2	21.3	17.5	7.43	9.22	6.34	7.63	**2.14**	**1.71**
u4	58.3	12.4	17.6	16.2	7.92	8.74	8.91	6.19	**1.19**	**1.56**
u5	45.2	21.4	18.3	15.3	8.73	6.12	7.43	8.11	**1.87**	**2.01**
u6	48.9	11.5	17.1	18.2	9.54	9.01	8.63	7.23	**2.01**	**2.33**
u7	36.7	21.4	18.9	19.9	6.94	5.91	7.23	3.71	**1.46**	**2.15**
u8	68.9	18.9	21.6	21.4	7.22	6.42	8.34	9.73	**2.14**	**1.61**
u9	44.1	31.3	19.3	19.8	9.02	6.71	9.84	7.29	**3.34**	**1.14**
u10	30.9	43.2	17.3	21.2	11.4	9.13	10.1	8.92	**1.73**	**1.28**
u11	23.5	12.3	18.3	22.1	7.44	8.21	9.23	9.31	**2.43**	**1.86**
u12	33.2	33.5	16.2	17.2	5.22	9.42	9.31	8.22	**1.71**	**1.92**
u13	19.5	54.3	19.7	18.1	8.23	6.12	8.34	9.31	**3.44**	**1.81**
u14	33.5	31.3	17.3	16.4	9.15	9.84	8.91	8.34	**1.29**	**1.38**
u15	20.4	59.6	17.3	16.3	7.97	8.92	7.25	8.81	**3.37**	**1.95**
u16	67.5	15.6	18.1	16.9	6.95	7.01	6.33	9.42	**2.31**	**2.11**
u17	39.4	13.7	19.2	14.5	9.12	9.21	8.93	7.71	**1.82**	**2.04**
u18	34.5	35.6	19.1	22.4	10.1	7.01	11.3	8.73	**1.01**	**1.72**
u19	37.4	15.8	15.1	26.8	6.02	5.17	9.61	7.29	**1.21**	**1.04**
u20	33.3	66.7	17.5	18.1	9.14	5.95	7.21	6.32	**2.04**	**1.33**
u21	28.5	19.8	22.1	17.2	7.05	5.11	8.87	9.82	**1.41**	**2.38**
u22	22.1	33.1	19.2	15.5	6.21	9.31	9.94	9.63	**2.12**	**2.24**
u23	21.8	32.1	19.8	22.6	9.11	8.01	12.1	8.24	**2.02**	**2.92**
u24	31.2	24.1	16.6	21.8	6.22	6.16	10.4	7.31	**3.11**	**1.14**
u25	38.2	22.4	22.1	19.3	8.34	8.91	8.22	4.72	**2.97**	**1.19**
Avg	36.0	26.5	18.6	19.0	8.09	7.58	8.79	7.94	**2.07**	**1.73**
standard deviation	13.5	15.7	1.86	3.00	1.47	1.55	1.46	1.56	**0.73**	**0.47**

subsequent sections, we study the effect of incorporating a PSO- and GA-based optimizer into the fuzzy classifier.

Continuous Learning using Dynamic Optimizers As mentioned above, after an initial rule base is generated, we use Particle Swarm Optimization (PSO) and Genetic Algorithms (GAs) to adapt to dynamically varying user keystoke behavior.

Particle Swarm Optimizer (PSO). The main idea of PSO [17] is to use a swarm of agents that is spread on the landscape of search space, and these agents, through local interactions, try to find an optimal solution to the problem. The characteristic that makes PSO successful is the communication between the agents which allows agents to converge to the best location. Table 3 tabulates the results of our fuzzy classifier optimized using PSO. It can be seen that the FAR and FRR values have improved significantly to approximately 8% (averaged). However, even after this improvement, a system with error rate of around 8% is not usable in the real-world.

Genetic Algorithm (GA). Genetic algorithms are well-known for providing acceptable solutions to dynamic optimization problems [5]. Unlike PSO, GA does not utilize feedback explicitly; rather, it uses genetic operators of selection, crossover and mutation to find the best solution. Use of GA reduces error rate to approximately 8% (averaged) which is almost the same as that obtained from fuzzy classifier optimized using PSO.

Hybrid PSO-GA. PSO utilizes the concept of feedback and GA uses the diversity achieved by randomness. Both of these optimizers have improved FAR and FRR con-

siderably. If we can somehow use the concept of feedback and randomness together, theoretically the accuracy of our fuzzy classifier should be improved. For this scenario, we use PSO and GA together for optimizing the database and rulebase of the feed-forward fuzzy classifier. The results of the fuzzy classifier optimized by a hybrid PSO-GA optimizer are tabulated in Table 3. It can be seen that the FAR and FRR have improved substantially to approximately 2%; as a result, our hybrid system is able to meet the accuracy requirements set earlier.

Another important thing to mention here is the standard deviation of the results. The standard deviation of our proposed hybrid PSO-GA-Fuzzy system is only 0.73% for FAR and 0.47% for FRR which is negligible. We have repeated the experiments for our scheme 500 times and the confidence interval of our results is 95% using t-distribution. This shows that the results produced by our system are statistically significant and the variation in the results is quite low.

5.2 Algorithm in Verification Mode

If the detection mode raises an alarm, the system moves to the verification mode. In this mode, we ask the suspicious user to enter a remembered 8-character PIN. During the PIN entry process, we observe his/her keystroke patterns and conclude whether or not the current user is an imposter. In this mode, the system extracts three features – key hold time, digraph (irrespective of the position of keys) and error rate – from the key log of entering the PIN. Note that here we use only three features because we have empirically determined that these features are sufficient to achieve close to 0% error.

We have also empirically determined if a potential imposter passes the test twice in three attempts, we declare him/her a legitimate user. We have arrived at this configuration by running a controlled experiment. We asked 10 of our colleagues to enter their PINs 30 times for training. After training, we asked all of these 10 colleagues to enter their passwords 5 times. We observed that each of them has been able to enter his/her password with correct behavior at least two out of the first three attempts. Later, we selected three imposters for each of those 10 colleagues and informed them about the correct passwords of legitimate users. We again requested imposters to enter the password 5 times and it was interesting to note that none of them was able to enter the password with matching behavior even once.

For PIN verification, we have designed a simple, efficient and accurate classifier specifically for keystroke dynamics. Our classifier dynamically assigns an impression coefficient (iC) to a user on the basis of his/her PIN typing keystroke pattern. We argue that a legitimate user is less likely to commit a mistake while entering his/her PIN; therefore, committing a mistake during the PIN entry process counts negatively towards the possibility that the current user is the legitimate user. We calculate the difference between the key hold times of keys of current profile with the key hold times of all the corresponding keys of the standard profiles of a user and then sum up all these differences to find the overall difference in the key hold time. Similarly, we find an overall difference in the digraph time. Finally, we sum overall key hold time difference and digraph difference to define the impression coefficient of PIN entering behavior. If a user commits a mistake during the PIN entry process, we penalize him/her for each error by adding l milliseconds to the overall difference value.

The overall difference is compared with a threshold value that is also dynamically calculated. If iC of a user is larger than this threshold value, we classify the user as an imposter otherwise he/she is a legitimate user. It is important to emphasize that we do not train our system with imposters' profiles. The mathematical explanation of our proposed classifier is given below:

The size of the PIN is s characters, the number of keys pressed by the user to enter s characters is represented by t, and the number of training profiles is represented by n. P^k is a matrix consisting of n rows corresponding to n training profiles and t columns corresponding to t key hold times. P^{uk} is a row vector of t columns; each column corresponds to a key hold time for a key press in an unknown profile. D^k, similarly, is a matrix of dimensions $n \times t-1$ for digraph times obtained from training profiles and D^{uk} is a row vector of $t-1$ columns representing the digraph times of an unknown user. The mathematical model is given in following equations:

$$iC = \sum_{i=1}^{n} \left(\sum_{j=1}^{t} \left| p_{ij}^{k} - p_{j}^{uk} \right| + \sum_{j=1}^{t-1} \left| d_{ij}^{k} - d_{j}^{uk} \right| \right) + e \times l, \tag{1}$$

$$\mu = \frac{\sum_{m=1}^{n} \left\{ \sum_{i=1}^{n} \left(\sum_{j=1}^{t} \left| p_{ij}^{k} - p_{mj}^{k} \right| + \sum_{j=1}^{t-1} \left| d_{ij}^{k} - d_{mj}^{k} \right| \right) \right\}}{n}, \tag{2}$$

$$\sigma = \left[\sum_{m=1}^{n} \left\{ \sum_{i=1}^{n} \left(\sum_{j=1}^{t} \left| p_{ij}^{k} - p_{mj}^{k} \right| + \sum_{j=1}^{t-1} \left| d_{ij}^{k} - d_{mj}^{k} \right| \right) - \mu \right\}^{2} / n \right]^{\frac{1}{2}}, \tag{3}$$

where $p_{ij}^{k}, p_{mj}^{k} \in P^k$; $p_{j}^{uk} \in P^{uk}$, $d_{ij}^{k}, d_{mj}^{k} \in D^k$, $d_{j}^{uk} \in D^{uk}$. Moreover, if $iC \geq \mu + a\sigma$ then the user is classified as an imposter; otherwise the user is classified as a legitimate user. We have empirically determined that values of $l = 5$ and $a = 3$ to provide approximately 0% error. The following section evaluates the accuracy of the proposed tri-mode system for varying system parameters.

6 Performance Evaluation

In this section, we first evaluate the accuracy of the proposed system for a fixed training profile. We then systematically evaluate the system's performance for different parameters. Specifically, we chronologically answer the following questions: (1) What is the accuracy of the system for a fixed profile?, (2) What is the impact of number of profiles (i.e., number of keys used for training) on the accuracy of our system?, (3) What is the relationship between the size of a profile and the accuracy of our system?, (4) What is the average user identification delay in terms of mobile phone usage (we report it in number of SMS)?, (5) How much damage an imposter can do in 250 keystrokes?, and (6) What are the training and testing times of our system?

What is the accuracy of the system for a fixed profile? The accuracy of our system can be viewed from the Table 4. It can be seen that our tri-mode system achieves 0% FRR and approximately 2% FAR after the verification mode. 0% FRR indicates that

Table 4. Accuracy results after detection mode and verification mode for a fixed profile size of 250 keystrokes

	After Detection Mode		After Verification Mode			After Detection Mode		After Verification Mode			After Detection Mode		After Verification Mode	
Users	FAR	FRR	FAR	FRR	**Users**	FAR	FRR	FAR	FRR	**Users**	FAR	FRR	FAR	FRR
u1	2.13	1.76	2.13	0	**u2**	1.61	0.82	1.61	0	**u3**	2.14	1.71	2.14	0
u4	1.19	1.56	1.19	0	**u5**	1.87	2.01	1.87	0	**u6**	2.01	2.33	2.01	0
u7	1.46	2.15	1.46	0	**u8**	2.14	1.61	2.14	0	**u9**	3.34	1.14	3.34	0
u10	1.73	1.28	1.73	0	**u11**	2.43	1.86	2.43	0	**u12**	1.71	1.92	1.71	0
u13	3.44	1.81	3.44	0	**u14**	1.29	1.38	1.29	0	**u15**	3.37	1.95	3.37	0
u16	2.31	2.11	2.31	0	**u17**	1.82	2.04	1.82	0	**u18**	1.01	1.72	1.01	0
u19	1.21	1.04	1.21	0	**u20**	2.04	1.33	2.04	0	**u21**	1.41	2.38	1.41	0
u22	2.12	2.24	2.12	0	**u23**	2.02	2.92	2.02	0	**u24**	3.11	1.14	3.11	0
u25	2.97	1.19	2.97	0	**Avg**	2.07	1.73	2.07	0	**SD**	0.73	0.47	0.73	0

our system is completely user friendly and never rejects a legitimate user. It also has a very low FAR compared with other techniques.

What is the impact of number of profiles on the accuracy of our system? Scalability analysis is important to determine the minimum number of profiles/keystrokes required to achieve acceptable accuracy. We take the users with the most number of profiles (u10, u14, and u15) for our scalability analysis and tabulate the results in Table 5. Note that each profile is made up of 250 keys. The results in Table 5 suggest a gradual, almost linear decrease in FAR and FRR as we increase the number of training profiles up to 50. This shows that as the number of training profiles increases, the accuracy of our system increases.

What is the relationship between the size of a profile and accuracy of our system? For the same users (u10, u14, and u15,) we now take 50 profiles of each user and study the relationship between FAR and FRR and the size of a profile. These results are also tabulated in Table 5. It is obvious from Table 5 that FAR and FRR values degrade for small size profiles, however, for a profile of 250 keys, the error rates on the average are 2%. It can also be seen that increasing the size of profile from 250 to 350 keys further improves the detection accuracy but the improvement is not much significant; therefore, we use a profile of 250 keys. Note that increasing the size of profile not only increases the detection accuracy but also the time required to make a profile. Our aim is to get reasonable detection accuracy with as small a profile size as possible. (Profile size of 250 keys satisfies the criteria.)

Table 5. Relationship of number of training profiles and size of a profile with error rates

	Number of profiles (Profile Size = 250)								**Size of profile** (Number of Profiles = 50)									
	20		**30**		**40**		**50**		**150**		**200**		**250**		**300**		**350**	
Users	FAR	FRR	FAR	FRR	FAR	FRR	FAR	FRR	FAR	FRR	FAR	FRR	FAR	FRR	FAR	FRR	FAR	FRR
u10	2.32	1.99	2.01	1.51	1.93	1.35	**1.74**	**1.29**	11.2	7.28	4.98	3.45	1.74	1.29	1.45	1.11	**1.10**	**1.01**
u14	3.21	2.21	1.97	2.01	1.77	1.78	**1.30**	**1.40**	9.21	8.12	4.11	4.01	1.30	1.40	1.03	1.21	**0.97**	**1.11**
u15	5.89	3.13	5.11	2.72	4.01	2.11	**3.39**	**1.98**	17.8	11.5	9.62	6.22	3.39	1.98	2.87	1.23	**1.91**	**0.99**

What is the user identification delay? Table 6 shows the average number of SMS a user types in a single profile. Remember our system tries to classify a user after every 250 keystrokes using the keystrokes features' set. It can be seen from Table 6 that on the

Table 6. User identification delay

Users	Average SMSs / Profile	Users	Average SMSs / Profile	Users	Average SMSs / Profile	Users	Average SMSs / Profile	Users	Average SMSs / Profile	Users	Average SMSs / Profile	Users	Average SMSs / Profile
u1	1.06	**u2**	1.44	**u3**	1.25	**u4**	1.09	**u5**	1.26	**u6**	0.65	**u7**	1.03
u8	1.03	**u9**	1.11	**u10**	1.34	**u11**	1.04	**u12**	1.27	**u13**	1.32	**u14**	1.39
u15	1.33	**u16**	0.16	**u17**	1.17	**u18**	1.15	**u19**	1.00	**u20**	1.43	**u21**	1.25
		u22	1.22	**u23**	1.24			**u24**	1.13	**u25**	1.30		

average a profile of 250 keystrokes is generated once a user sends just 1 SMS. So our detection delay is bounded by the time within which a user sends an SMS. Detection delay is not a significant problem if the objective of an imposter is to steal the mobile phone. However, this delay will become very crucial if an imposter wants to steal information from the mobile phone. We invoke the verification mode of our system to disallow an imposter to transmit data from the mobile phone.

How much damage can an imposter do in 250 keystrokes? We have done an interesting study in which we requested 4 of our colleagues in the virology lab to act as imposters on a trained mobile phone to get an understanding of how much data they can read in a given document. (Remember that they cannot upload or copy the data because of the verification mode). We downloaded the current paper (18 pages long) on a smart phone and told the imposters the exact directory path of the paper. (Remember that it is the best case scenario for an imposter; otherwise, he needs to press more keys to search a document). We asked them to find the line "and the system will loose its usability appeal" in the paper that happens to be on page 9. We tabulate the number of keys pressed by each of them in Table 7 to locate the required information. It is interesting to note that, even in this best case scenario, only one of them was able to locate the information within 250 keystrokes. We have also done an interesting study to understand that how far different imposters can scan the given document in 250 keystrokes. **i4** appears to be a smart imposter who manages to reach page 14 in 250 keystrokes. Most of the users pressed 250 keystrokes in 8 to 15 minutes.

Table 7. Analysis showing the number of keystrokes to perform the task

Imposters	Number of Keys to Perform the Task	Page # After 250 Keys	Imposters	Number of Keys to Perform the Task	Page # After 250 Keys
i1	332	7	**i2**	442	6
i3	297	8	**i4**	189	14

What are the training and testing times of our system? We now analyze the training and testing times of different classifiers in Table 8. The training time of our classifier is 28 seconds, but our testing time is just 520 milliseconds. Thus, while the system's run-time complexity is comparable to other existing algorithms, its training complexity is significantly higher. The main source of this complexity are the back-end dynamic optimizers used in our system. However, we emphasize that training complexity needs to be incurred very infrequently after every 5 training profiles (it might take few hours).

Table 8. The processing overheads of classifiers on an old 233MHz, 32MB RAM computer

Algorithm	Train (*secs*)	Test (*secs*)	Algorithm	Train (*secs*)	Test (*secs*)	Algorithm	Train (*secs*)	Test (*secs*)	Algorithm	Train (*secs*)	Test (*secs*)
PSO-GA Fuzzy	28	0.52	**Naive Bayes**	0	0.52	**BPNN**	4.8	2.0	**RBFN**	0.41	0.42
			Kstar	8	0.21	**J48**	0.23	0.22			

Moreover, unlike desktop computers, mobile phones remain idle much of the time and the retraining can be performed during these inactivity periods.

7 Limitations and Potential Countermeasures

We now highlight the important limitations and countermeasures of our system.

Identification delay period. Our system can detect an imposter after observing a minimum of 250 keystrokes. The identification delay is hence a function of the imposters keyboard usage. We argue that an imposter's keyboard usage can belong to one of the following two types: (1) he/she wants to get access to the sensitive information/documents on the phone, and (2) he/she wants to steal the mobile phone. In the first case, the imposter must try to quickly get access to the sensitive information and, as a result, the time to generate a profile of 250 keystrokes, as mentioned before, will reduce to 10-15 minutes. If the imposter is of the second type, then our system will detect him/her once he/she tries to login through our PIN verification procedure.

Accuracy is sensitive to the number of profiles. Another shortcoming of our approach is that it requires a cold start of 30 or more profiles to accurately learn the behavior of a user. In this time period, the system might suffer from relatively high FAR and FRR which are still comparable with the existing techniques (see Tables 4 and 5). But our system provides significantly better FAR and FRR after collecting just one week of training data, which we believe is quite reasonable.

Portability to full keyboard smart phones. We have not tested our prototype on BlackBerry category of phones which have full non-multiplexed keyboard. While we believe that the results of our system will scale to these phones, we are currently soliciting volunteers with full keyboard Nokia phones for testing and evaluation.

Relatively large training time. Our systems takes 28 seconds on the average once we retrain it after every 5 profiles. During these 28 seconds after every few hours, the response time of the mobile phone degrades which might result in some annoyance to the user. We argue that this cost is worth the benefit of very low FAR and FRR values of our system. Moreover, as suggested earlier, the retraining module can be customized to execute during inactivity periods.

Resilience to reinstalling OS. A savvy imposter may reinstall the OS on the phone, thus circumventing our system. This is a common limitation for all host-based intrusion detection systems. A solution to this problem is OS virtualization which is computationally infeasible on contemporary mobile phones.

Table 9. Improvement shown by Hybrid PSO-GA Fuzzy system over the other classifiers

Algorithm	%Improvement		Algorithm	%Improvement		Algorithm	%Improvement		Algorithm	%Improvement	
	FAR	FRR		FAR	FRR		FAR	FRR		FAR	FRR
Naive Bayes	94.4	94.3	**BPNN**	95.0	94.6	**RBFN**	94.2	93.5	**Kstar**	92.9	94.4
J48	93.1	95.7	**Fuzzy**	88.8	89.1	**PSO-Fuzzy**	74.4	77.2	**GA-Fuzzy**	76.4	78.2

8 Conclusion and Future Work

We proposed a user identification system that monitors the keystroke dynamics of a mobile phone user to differentiate legitimate users from imposters. We have used a custom dataset of 25 diverse mobile phone users to show that the proposed system can provide an error rate of less than 2% after detection mode and an FRR of close to zero after PIN verification mode. We have also compared our approach with 5 state-of-the-art existing techniques for keystroke-based user identification. Table 9 shows percentage improvement of our scheme compared to the existing schemes on our dataset. In future, we plan to port our solution on full-keyboard mobile phones and evaluate its accuracy on them. We also intend to incorporate our system in a Symbian mobile phone and then evaluate its accuracy on-line under real-world usage circumstances.

References

1. Red herring mobiles scream for help: Uk-based mobile security company adds security to mobile phones, October 2006.
2. S60 3rd Edition. Getting Started With C++ Application Development Using CodeWarrior IDE.
3. S. Babin and A. Pranata. *Developing Software for Symbian OS: A Beginner's Guide to Creating Symbian OS V9 Smartphone Applications in C++*. John Wiley & Sons, 2007.
4. S. Bleha, C. Slivinsky, and B. Hussien. Computer-access security systems using keystroke dynamics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(12):1217–1222, 1990.
5. J. Branke. *Evolutionary Optimization in Dynamic Environments*. Kluwer Academic Publishers, 2002.
6. SK Card, TP Moran, and A. Newell. Computer text-editing: An information-processing analysis of a routine cognitive skill. 1987.
7. N.L. Clarke and S.M. Furnell. Authentication of users on mobile telephones–A survey of attitudes and practices. *Computers & Security*, 24(7):519–527, 2005.
8. N.L. Clarke and S.M. Furnell. Authenticating mobile phone users using keystroke analysis. *International Journal of Information Security*, 6(1):1–14, 2007.
9. J.G. Cleary and L.E. Trigg. Kˆ*: An Instance-based Learner Using an Entropic Distance Measure. In *Machine Learning-International Workshop then Conference*, pages 108–114. Morgan Kaufman Publishers, Inc., 1995.
10. M.D. Corner and B.D. Noble. Zero-interaction authentication. In *Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 1–11. ACM New York, NY, USA, 2002.
11. A.P. Engelbrecht. *Fundamentals of computational swarm intelligence*. John Wiley & Sons, 2006.
12. D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1989.

13. http://www.biopassword.com. BioPassword Inc.
14. Y.S. Hwang and S.Y. Bang. An Efficient Method to Construct a Radial Basis Function Neural Network Classifier. *Neural Networks*, 10(8):1495–1503, 1997.
15. R. Joyce and G. Gupta. Identity authentication based on keystroke latencies. *Communications of the ACM*, 33(2):168–176, 1990.
16. S. Karatzouni and N. Clarke. Keystroke Analysis for Thumb-based Keyboards on Mobile Devices. *International Federation for Information Processing-Publications-IFIP*, 232:253, 2007.
17. J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, 1995.
18. J. Leggett and G. Williams. Verifying identity via keystroke characteristics. *International Journal of Man-Machine Studies*, 28(1):67–76, 1988.
19. J. Leggett, G. Williams, M. Usnick, and M. Longnecker. Dynamic identity verification via keystroke characteristics. *International Journal of Man-Machine Studies*, 35(6):859–870, 1991.
20. P. Lilley. *Hacked, Attacked & Abused: Digital Crime Exposed*. Kogan Page Ltd, 2002.
21. D. Mahar, R. Napier, M. Wagner, W. Laverty, RD Henderson, and M. Hiron. Optimizing digraph-latency based biometric typist verification systems: inter and intra typist differences in digraph latency distributions. *International journal of human-computer studies*, 43(4):579–592, 1995.
22. M.S. Obaidat and B. Sadoun. Keystroke Dynamics based Authentication Biometrics. *Systems, man and cybernatics*, 27(2):261–269, 1997.
23. J.D. Paola and R. Schowengerdt. A detailed comparison of backpropagation neural network and maximum-likelihood classifiers for urban land use classification. *IEEE Transactions on Geoscience and Remote Sensing*, 33(4):981–996, 1995.
24. J.R. Quinlan. Bagging, Boosting, and C4. 5. In *Proceedings of the NCAI*, pages 725–730, 1996.
25. I. Rish. An empirical study of the naive Bayes classifier. In *Proceedings of IJCAI-01 Workshop on Empirical Methods in Artificial Intelligence*, volume 335, 2001.
26. D. Sims. Biometric recognition: our hands, eyes, and faces give us away. *Computer Graphics and Applications, IEEE*, 14(5):14–15, 1994.
27. D. Umphress and G. Williams. Identity Verification Through Keyboard Characteristics. *International Journal of Man-Machine Studies*, 23(3):263–273, 1985.
28. X. Wang, M.H. Heydari, and H. Lin. An intrusion-tolerant password authentication system. In *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*, pages 110–118, 2003.
29. I.H. Witten, University of Waikato, and Dept. of Computer Science. *WEKA Practical Machine Learning Tools and Techniques with Java Implementations*. Dept. of Computer Science, University of Waikato, 1999.
30. L.A. Zadeh. Fuzzy sets. *Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems: Selected Papers*, 1996.