Keystroke Biometric System Test Taker Setup and Data Collection

Kurt Doller, Sarika Chebiyam, Smita Ranjan, Elyse Little-Torres, and Robert Zack

Seidenberg School of CSIS, Pace University, White Plains, NY, 10606, USA
{kd54446w, sc53019w, sr79540n, et37640n}@pace.edu, robert.zack@stpart.com

Abstract

Pace University has been conducting keystroke biometric research for seven years. The system under development has the capability of identifying or authenticating users from typing characteristics and patterns. It consists of three primary components: a keystroke entry system that collects data over the Internet, a feature extractor, and a pattern classifier. This paper is focused on the keystroke entry system component and enhancements made to it to support the biometric authentication of students taking tests over the Internet. The keystroke entry system is enhanced to operate in stealth or background mode so that a user's typing characteristics and patterns can be captured anonymously while the user participates in an online test. The system compares test and enrollment samples to authenticate test takers.

1. Introduction

Keystroke biometric systems measure typing characteristics believed to be unique to an individual and difficult to duplicate. When incorporated into systems such as Internet based test taker applications, keystroke biometrics can have important security benefits and can provide an additional layer of security to a system. This work enhances the keystroke entry system (KES) component of Pace University's Keystroke Biometric Authentication System (BAS).. Pace's Keystroke Biometric System [2, 3], developed over the last seven years, has been designed for both identification (1-of-n response) and for authentication (binary response, yes you are the person you claim to be or no you are not).

This paper discusses the function and operation of the enhanced KES on-line test taker system. The KES is a Web-based application that provides web interfaces to the Instructors for entering student & course data, and to the students for answering questions. It output is a set of text files containing the raw keystroke data for each student and each question. To participate, a test taker must operate a computer that runs the Microsoft Internet Explorer browser capable of displaying HTLM web pages and has a minimum Java runtime environment of 1.4 installed. The server where this Web-based application is hosted should be able to run PHP scripts. In addition, the test taker's computer must be connected to the Internet and be equipped with a laptop or desktop keyboard. Firewall and anti-virus software must permit the java applet to run. This applet, the only locally installed software component required to run the KES, collects raw keystroke data during the testing session.

Feature measurements are extracted from the raw data and processed by a biometric classifier that uses the *k*-Nearest Neighbor classifier. The collected data samples are processed and compared against the archived enrollment samples to make an authentication decision. The test taker is either matched (accepted as the person claiming to be taking the test) or not matched (rejected).

The sections of this paper are organized as follows. Section 2 presents background information necessary to understand the system, section 3 describes the system and focuses on the new additions and enhancements, section 4 discusses the software development methodology, section 5 the Results produced at the end of our study and sections 6, 7 present the conclusions and recommendations for future work.

2. Background

The version of the KES used in this study was significantly enhanced over previous versions. This version provides the instructor with an interface to setup and customize questions used for testing the knowledge of the student and for obtaining keystroke biometric samples. Depending on the desire of the instructor or on the design requirement of related studies, the students may or may not be informed that their keystrokes are being captured for the purpose of biometric authentication.

This study is limited to the laptop free-text and desktop free-text input modes discussed in [2]. The test taker is presented with a question from a list of choices that the instructor provides in the online test setup. Users can write anything they want in answer to the question. A sample is considered complete if at least 300 keystrokes, approximately five lines of typewritten

text, are collected. [We have found the 300 keystrokes us usually sufficient, so please change this.] The test taker knows that a Java applet is installed on their computer to capture the answers to the online test, but otherwise may be unaware of its biometric sample collection function. Privacy issues do not come into play for this academic purpose when the user is not aware that her/his keystrokes are also captured for biometric authentication.

Potential applications of this system are numerous. Work place environments can implement the KES system to authenticate employees for compliance and other testing scenarios. In academia, the application for online test taking has been discussed and described in [2]. This study attempts to extend previous authentication research into a real world application.

3. The Enhanced Keystroke Entry System

The Keystroke Entry System that we have developed and enhanced comprises of the following components as shown below in our System Overview Diagram.

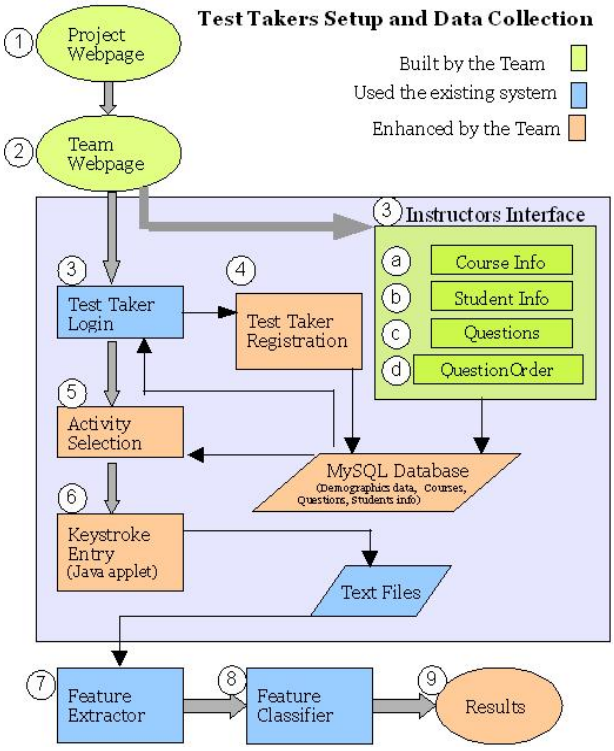

Figure 1. System Overview Diagram

The system is initiated by the Instructor, by entering the course information, student ids, names, test questions and the order in which they have to be answered, into the database tables using the Professor's interface. Then, the students login to the Test Takers Interface using the project webpage and register as new users by providing their demographic data, which is stored into the database and retrieved later when the user logs in to take the test. At this step, the student's first name, last name, semester and course id should match with those entered by the professor in the database for validation. To take the test, the registered students login to the interface and answer the questions which will be displayed from the database in an order chosen by the professor.

Once the test taker specifies the keyboard type he/she is using, the questions are displayed one at a time in an order chosen by the Professor. Each question is displayed by a java applet which monitors the text entry screen where all the keystrokes entered by the student are captured without his/her knowledge into separate text files on the server. These text files are analyzed and compared by feeding into the BAS system's Feature Extractor and Feature Classifier to identify the keystroke patterns of each test taker and thus authenticate them. The following sections explain the functioning of the system in detail.

3.1 Instructors Interface

The instructors interface [5] or professor's Interface is an enhancement to the KES system. It allows the instructor to add or edit the questions he wants to ask the test takers during a testing session.

Earlier versions of the applications required the instructor to directly edit the PHP scripts / HTML pages to perform this function which would have been a cumbersome and error-prone task. The new interface allows the instructor to easily and dynamically enter course information, student details and test questions for use in the on-line test and biometric sample collection.

The instructor arrives at the Instructor's interface (3) via the Project web page (1) and Team web page (2). in Figure 1. Then in step 3a the Instructor is presented with the screen in Figure 2 where he/she enters the course information.

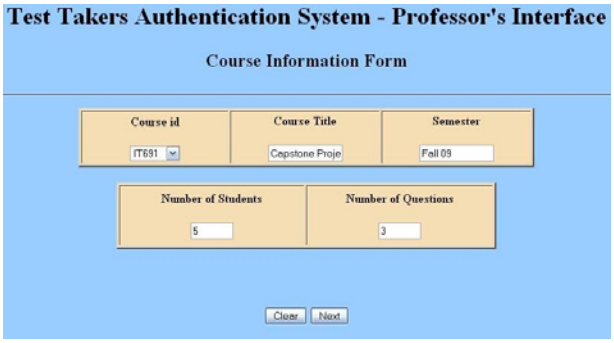

Figure 2. Instructor's interface - Course information screen

When the Instructor clicks on Next, the course ifnormation is entered into a database table called *course_ids*. Then the control goes to the screen which will present the student information form as displayed in Figure 3. In step 3b, the instructor makes a one-time entry of the student names and identification numbers. These values are stored in a table labeled *student_info*, which will be used to quickly accept or reject the students when they login.

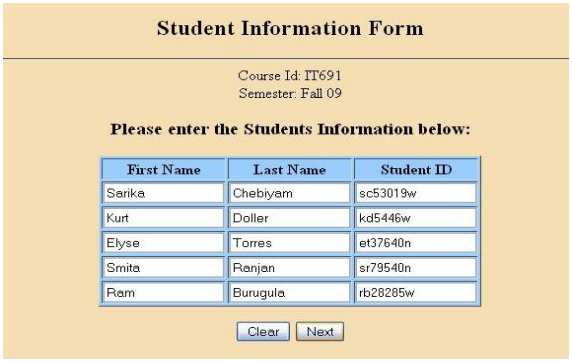

Figure3. Instructors interface – Student information screen

Next the instructor is presented with a question list entry form in step 3c as displayed in Figure 4, where the instructor can enter the list of questions for for the course. These questions are stored in the database table *course_questions.*

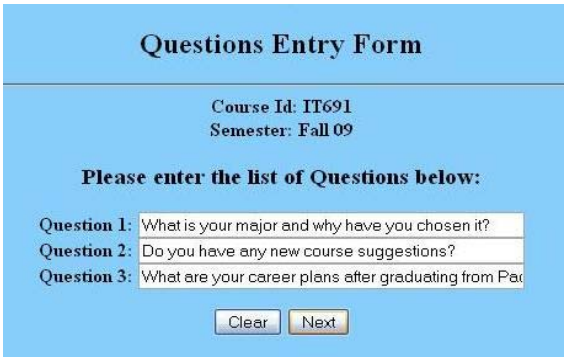

Figure 4. Instructors interface – Questions screen

Next the instructor is presented with a question-order selection form in step 3d as displayed in Figure 5. Here the instructor can choose a particular question order for each student individually or the questions can be randomly shuffled for the test takers during the testing session checking to see that no question is repeated.

Figure 5. Instructors interface Question Order screen

At the end of this process a conformation screen is displayed. Now all the information entered by the instructor is stored in the respective database tables and the questions will be displayed to the test takers as they login.

3.2 Test Taker Setup / Data Collection

The test taker enters the system via the Project web page and Team page in steps 1 and 2 respectively as shown in System Overview diagram in Figure 1. In step 3 the test taker is presented with the Login screen as shown in Figure 6 below.

Figure 6. Test Taker's Interface - Login Screen

Here the Test taker enters his/her First name, Last name, Semester, and Course ID. If the test taker has already registered with the KES before, then the test taker is presented with the Keyboard selection screen in step 5 as displayed in Figure 8. If this is the first time the test taker is using the KES, then he/she is presented with the Demographics form in step 4 as displayed in Figure 7 below, where he/she enters various personal information.

Note that the test taker has to be also registered by the Instructor for the specific semester and course ID

they entered in the login screen. If they try to login to the KES without being registered by the Instructor for that course and semester apriori, an error message screen is displayed asking them to contact the professor or retry loging in.

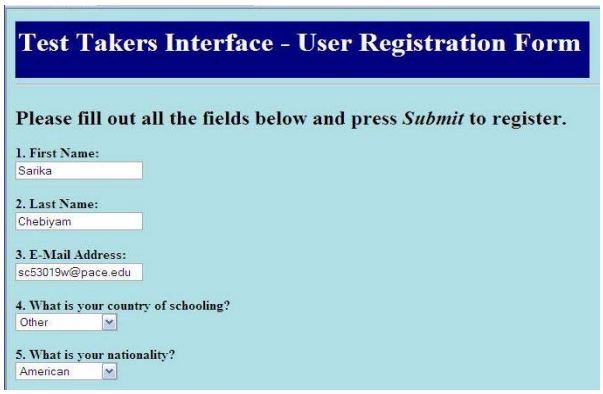

Figure 7. User Registration Form

After the test taker enters the demographic data in step 4, that data is then written to the demographics table and the user is thanked for their enrollment. Then the user is redirected to the login screen as shown in Figure 6 to attempt logging in to KES again.

After successfully logging in to the KES this time, the test taker is then presented with the Keyboard selection screen in step 5 as shown in the Figure 8 below. Here the test taker has to specify the type of keyboard they are using, laptop or desktop.

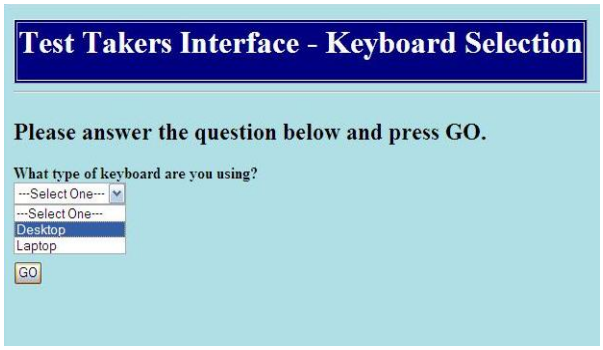

Figure8. Keyboard Selection screen

After selecting the type of keyboard once on this screen, it takes the user to the keystroke Java applet to answer the questions one after the other in the order chosen by the instructor as in the screen shown in Figure 9 below.

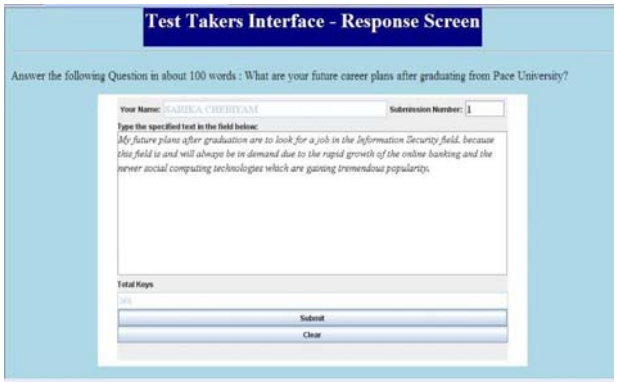

Figure 9. Display Question Screen

Here the test taker is asked to answer several questions specified by the Professor. The user must have at least Java Runtime Environment (JRE) 1.4 to launch the applet. The application is tested to work with Microsoft Internet Explorer. Other browsers are not supported at this time and functionality of using them may be limited.

Each question is displayed by a Java applet. There are eight pieces of information sent to, and required by, the applet: first name; last name; experiment style (e.g., free text); sequence number for the selected experiment style (respective counter field value); keyboard style; awareness [3]; URL of the script that sends the next question; and the URL of the script that saves the keystroke raw data into a file. Awareness refers to whether the user knows he/she is working with a stealth or visible KES. If the Java applet does not receive these eight values, or if the user does not have a Java Runtime Environment (JRE) equal to or later version 1.4, the applet will not launch. [3].

KES Applet

All versions of the KES applet require collection of enrollment data from the user. This data becomes part of the training data used for comparison against test samples collected during an on-line testing session. For enrollment, the user has the choice of copying text passages or writing a letter as free text. This study uses free text mode only. In the old system, during test taking process, the user could see the on-screen statistics, including the total number of keys pressed, the current character pressed and current key press and release timings (Figure 10).

When integrated into an on-line test, the enhanced applet removes queues and indicators seen in the keystroke entry process to enable stealth or anonymous operation. The enrollment process and visible operation of the KES are the same. However, the new system checks for the student identity in two ways, one is by

comparing from the registration information and another from the information entered by the professor through his interface. For anonymous or stealth operation, the user is not aware that biometric samples are being collected.

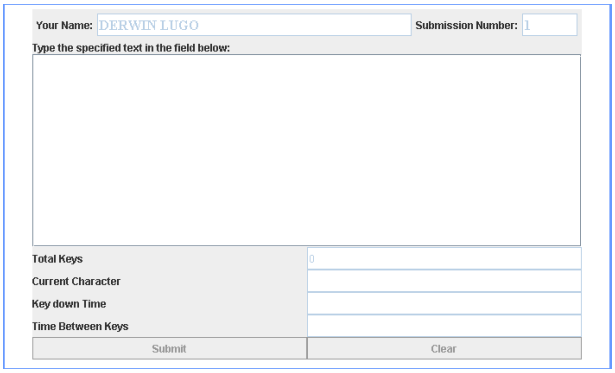

Figure 10. Legacy Java applet before any keystrokes have been entered [3]

Depending on the sample being collected, the system counts the number of keystrokes. When the user correctly completes the task and clicks submit, a PHP script is invoked, which writes the raw data information to a text file and updates the user's counter field database. This action is transparent to the user. The text file includes the actual keystrokes entered by the user, the durations or amount of time each key was pressed, and the latency, also known as the elapsed time between adjacent keystrokes. For ease of locating the raw data files, each condition, laptop and desktop, are included in the file name itself directory on the server. The researcher should become familiar with the data storage locations as the raw files should be copied to a work area in separate storage for feature extraction and classification activities.

Figure 10 depicts the keystroke entry applet when operating in a non-stealth data collection mode. The sequence numbers incremented after each valid sample is accepted. The user could enter another sample or click the back button to return to the activity selection page [3].

Stealth Data-Capture Mode

Enhancements to the test-taker applet enable keystroke data to be captured anonymously and in stealth mode. The test taker's keystrokes are collected in the background without queues or user awareness. When the test taker first accesses the online test, they are prompted to login by entering their first and last name as they did when they enrolled. It is important the login name match the enrollment name.

Once the user is logged into the system, the window with the online test appears. The stealth Java applet is running, but all queues and onscreen statistics are hidden from the display. The applet overhead is small and the user should be unaware that keystroke biometric samples are being collected.

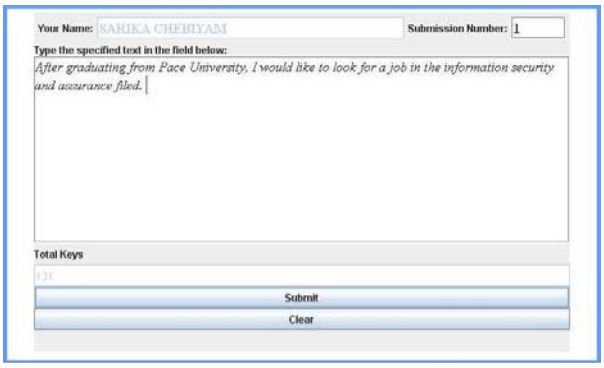

Figure 11. Test Applet with keystroke data invisible

If the test taker or the user does not enter the required 300 keystrokes, the system will display an error message requesting them to continue typing until the required keystrokes are met. Keystroke characteristics are stored in separate text files named after their selection criteria

These files must be downloaded from the server to a storage location designated for processing by the BAS system. The BAS system is a Java based application detailed in [2, 3]. Features are extracted from the raw data and classified by BAS.

Code Refactoring

A significant part of the effort in the development of this new KES system also involved refactoring the old code so that it is much simpler and easier to maintain and improve in future. Both the old code and new code consists of the same types of files – HTML, PHP, and Java. However, the old code has 36 files – 29 HTML files, 5 PHP files and 2 Java files, whereas as the new code has only 23 files – 9 HTML, 13 PHP, and 2 Java. – and that includes the new Professor's Interface functionality. The number of HTML files were significantly reduced because in the new system, the questions are retrieved and displayed from the database tables instead of a separate html file for each question as was the case in the old system.

Also, the old code has URLs / absolute paths hard coded in 40 places, whereas the new code has the URL specified in only one config file, kbs_globals.php. Every PHP script, HTML, or Java file will read the URL from this config file either directly or indirectly. Moreover, there are no absolute paths in any file; all the

absolute paths were changed to relative paths. This makes moving this application to a different server and/or different directory location much easier in future.

3.3 Feature Extraction and Classification

The feature extraction program reads all of the raw data text files from the processing storage location. Storage locations can be local or network based and should be separate from the primary storage location used by the applet. The demographics file created during test taker enrollment is also required to run the BAS system, although the contents of it are reserved for future use. The demographics file should be stored in a separate directory from the raw data. The BAS Biofeature application generates a features file from the raw data collected by the Java applet. After processing, each test taker's information becomes a row in the features file. The features file is used by the BAS classifier component.

The BAS System uses the *k*-Nearest Neighbor classifier. As part of the processing the multi-class input data is dichotomized into two classes. The test taker samples are decided to be within-class or between-class by the classifier. With-class samples are decided by the classifier to be "you are authenticated". Between-class samples are decided by the classifier to be "you are not authenticated" [2].

The BAS classifier component consists of Java-based components. The GUI allows the researcher to dichotomize all of the data or a portion of the data. This is useful for data sets with large numbers of samples [2]. Similar to the Feature Classifier, the BAS program uses the "train-on-one/test-on-another" to attain results.

The researcher specifies training and testing feature files for an experiment, then clicks "Apply Dichotomy Model" to perform the dichotomy transformation (Figure 12).

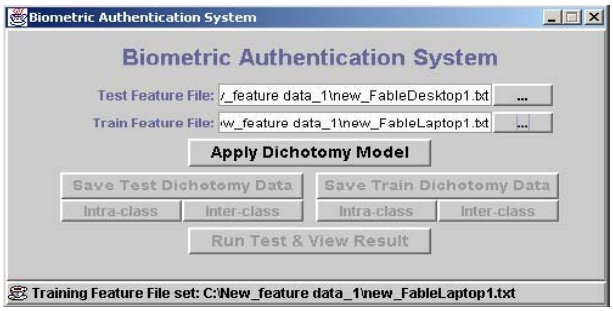

Figure 12. Biometric Authentication System (BAS)

After all features have been extracted into a data or "features" file, the data is ready to be classified. The matching process uses Euclidian distances of all the collected features.

4 Software Development Methodology

Our Methodology incorporated agile project development techniques and Extreme programming. Deliverables were framed around two week iterations and a task backlog. In order to gain velocity, deliverables that could not be met within an iteration cycle were further decomposed into tasks that could be accomplished in the iteration cycle.

Project communications between team members took the form of frequent conference calls, at least once per week. This level of communication helped keep the project on-track and maintain clarity and understanding of the deliverables. Communications between team members and customers were primarily done with email and phone calls on an as needed basis. The project status reports were uploaded onto the team web page[4] every week for the customers and the to keep track of our progress regularly.

5 Student Authentication Results

Our efforts are the result of taking test samples submitted by users over several weeks. The test samples were requested on a weekly basis. The test samples consist of both laptop and desktop free text entries in which users would answer one of several questions displayed. The original 677 keystroke requirement was later modified to 300 which earlier studies showed sufficient to attain reasonable authentication accuracy. Users were requested to submit a different sample of free text each week. The results were captured and stored on the Pace Utopia server for analysis. The raw data was then downloaded from the Utopia server and used to generate FAR and FRR scores along with performance results.

The False Acceptance Rate is the rate at which a negative result is classified as positive. The False Rejection Rate is the rate at which negative results are classified as positive. In our case the FAR and FRR rates would be the rate at which individuals were incorrectly accepted or rejected. Once the files were downloaded we used the standalone biofeature to process the raw data. The results were then split into two files for training and testing. As soon as this was completed the BAS application was run to process the training and testing files. Lastly we used the calculate accuracy tool to in order to accumulate results for 1, 3, 5, 7, 9 NN (nearest neighbor).

Our testing consisted of 5 samples each from 5 individuals using a laptop and 5 more samples from individuals using a desktop. The testing performed was one-within class test sample, a difference between feature vectors of the same individual, which resulted in

a correct within-class match (FRR=0/1 or 0% and Performance=1/1 or 100%). And the same result was shown for kNN or 1, 3, 5, 7, and 9. Table 1 shows this result from free text laptop. The table 2 representation shows the desktop free text result.

Table 1.Testing one within-class laptop sample.

Test Sizes	Train Sizes	FRR	FAR	Performance	Test Subject | AVG(Sample)	Train Subject | AVG(Sample)	kNN
1-0	113-833	.00% (0/1)	---	100.00% (1/1)	1 | 2.00	10 | 4.40	1
1-0	113-833	.00% (0/1)	---	.00% (0/1)	1 | 2.00	10 | 4.40	3
1-0	113-833	.00% (0/1)	---	.00% (0/1)	1 | 2.00	10 | 4.40	5
1-0	113-833	.00% (0/1)	---	.00% (0/1)	1 | 2.00	10 | 4.40	7
1-0	113-833	.00% (0/1)	---	.00% (0/1)	1 | 2.00	10 | 4.40	9

Table 2. Testing one within-class desktop sample.

Test Sizes	Train Sizes	FRR	FAR	Performance	Test Subject | AVG(Sample)	Train Subject | AVG(Sample)	kNN
1-0	17-119	.00% (0/1)	---	100.00% (1/1)	1 | 2.00	8 | 2.12	1
1-0	17-119	.00% (0/1)	---	.00% (0/1)	1 | 2.00	8 | 2.12	3
1-0	17-119	.00% (0/1)	---	.00% (0/1)	1 | 2.00	8 | 2.12	5
1-0	17-119	.00% (0/1)	---	.00% (0/1)	1 | 2.00	8 | 2.12	7
1-0	17-119	.00% (0/1)	---	.00% (0/1)	1 | 2.00	8 | 2.12	9

The testing we performed used one pair of samples from one student and mimics what would occur in an actual class environment taking an online test. Our results show that the software will perform the testing one-student at a time.

6 Conclusions

Keystroke biometrics is relatively new in its application and testing phase. The potential uses of this technology especially in the area of security are promising. It can be used to identify and authenticate individuals, which could prove useful in both corporate and academic environments. With the advent of online courses and the misuse of e-mail becoming more prevalent the need for such a type of technology becomes evident.

BAS can accurately authenticate individuals taking online tests using the enhanced KES. The system can authenticate a user with high accuracy if the user completes the enrollment process and answers the questions in the test-taker applet using the same type of keyboard. The authentication system consists of three components: data collection, feature extraction and classification.

The KES system previously described [2, 3] has been enhanced and consists of the following components (see Figure 1):

1. The instructors interface is a web-based application written in PHP which allows the instructor to setup the on-line test and provide the test questions. These questions are presented to the test takers taking the online test (Figure 9).
2. The Test-Takers interface is web –based application consisting of HTML pages, PHP scripts and Java applets. This interface requires a test taker to register. If the required Java applet is not present, the user will be prompted to install it. Once user setup is complete, the user can take the test.
3. Once registered, the test taker logs into the test. A modified Java applet, operating anonymously in Stealth mode, captures keystrokes at predefined points in the testing session and produces a raw feature file that is subsequently analyzed by the BAS system.

In our research, we captured keystroke data using an online test-taking web interface. The interface was modified so the keystroke data could be obtained in a stealth mode without the knowledge of the test taker. The original 677-keystroke requirement was changed to 300, which proved sufficient to provide meaningful results to authenticate the test takers.

A detailed User Manual [6] document step-by-step instructions for using the system.

7 Recommendations

Several further enhancements can be done to the Keystroke entry system to make it more useful in the future. One of them is to research the potential to imitate user's keystrokes. This type of investigation could provide insight into vulnerabilities of the system and could bring potential problems to the surface to be corrected. The Pace studies have focused on identifying and authenticating the users until now. It is important to also investigate more into the potential security issues that may or may not exist within the keystroke system. Provision of a secure login for the instructor and the test takers by linking this system with the pace university Blackboard or the PacePortal will address the security issues to a great extent. Another recommendation is to send the keystroke raw data from the Java applet to the server periodically during the typing instead of waiting till the test taker presses the "submit" button at the end. This way, the BAS system can authenticate using the intermediate data and can alert the Professor

immediately if it cannot authenticate the user. The idea is that the chance of finding a cheating student is higher if you catch while he/she is still writing the exam, instead of waiting all the way to the end. A third recommendation is to enhance this application so that it can support touch-screen based keypads such as Smart Phones, or the new All-in-one touch screen computers. In future, to make it more developer friendly, it would be nice if the system is hosted on a Linux server with SSH capability.

References :

[1] Admit One Security Inc. (1998-2008), *AdmitOne Security Suite: Risk based Authentication for the web*, <http://www.admitonesecurity.com/> (15 November 2008)

[2] C.C. Tappert, M. Villani, and S. Cha, "Keystroke Biometric Identification and Authentication on Long-Text Input," chapter in *Behavioral Biometrics for Human Identification: Intelligent Applications*, Edited by Liang Wang and Xin Geng, 2009.

[3] M. Villani, C.C. Tappert, G. Ngo, J. Simone, H. St. Fort, and S. Cha, "Keystroke Biometric Recognition Studies on Long-Text Input under Ideal and Application-Oriented Conditions," Proc. CVPR 2006 Workshop on Biometrics, New York, NY, 2006.

[4]TeamWebpage- http://utopia.csis.pace.edu/cs691/2009-2010/team4/team4/index.html

[5]Instrustors-Interface - http://utopia.csis.pace.edu/cs691/2009-2010/team4/team4/ProfessorInterface/courseInfo.php

[6] Kurt Doller, Sarika Chebiyam, Elyse Little-Torres, "User Manual: Keystroke Biometric Authentication System – Test Taker Setup and Data Collection", School of CSIS, Pace University, Fall 2009