Proceedings of Student-Faculty Research Day, CSIS, Pace University, May 3rd, 2013

Multi-Biometric System

Calvin Chan, Enpaksh Airon, Gowtham Narasimhaiah, Usha Aiyer, Ned Bakelman, and Vinnie Monaco
Seidenberg School of CSIS, Pace University, White Plains, NY 10606

Abstract

The 2008 federal Higher Education Opportunity Act requires institutions of higher learning to make greater access control efforts for the purposes of assuring that students of record are those actually accessing the systems and taking exams in online courses by adopting identification technologies as they become more ubiquitous. Previously, keyboard input and mouse movement data have to be analyzed to authenticate the computer users. Stylometry is added to the result in a combination of three biometric. The keystroke feature data can be availed by executing the keystroke feature extractor. These data are passed as input to the 'parser' program to get output in text form. These text form output is then fed as input to the stylometry feature extractor. The mouse feature data can be availed by executing the mouse feature extractor. The output obtained from execution of all the three feature extractors are then combined by executing the 'assembler' program, to form the multi-biometric feature data records on which the performance is then measured.

1. Introduction

The main application of interest in this study is to verify the identity of students taking online tests. These checks are becoming more important these days as the student enrollment for online classes are increasing. To address this, the current study focuses on developing a *multi-biometric system* that combines the keystroke, stylometry and mouse movement biometrics, to then measure the performance.

When a user types the characters, the keystroke rhythms are captured to develop a unique biometric template of the users' typing pattern, for future authentication.

Stylometry that assists in the study of linguistic style and analyzing texts can then be used for evidence of authenticity and identity.

By tracking the movement of the mouse trajectories through different arcs, it should also be possible to authenticate and verify the identity of the user to some extent, although this is rather weak biometrics. In particular, mouse movement biometrics can be used to enhance or augment keystroke and other biometrics like stylometry.

The target of developing such a system is to investigate several biometric system components for potential integration into a powerful cyber security system to provide a multi-level computational behavioral cognitive "fingerprint" of the person operating the computer. For example, keystroke and mouse components operate at the subconscious automatic motor control level, a stylometry component operates at the higher cognitive linguistic (character, word, syntax) level, and an intruder operational behavior component operates at the highest cognitive semantic level of intentional motivation.

For using such a continual authentication system on government or private company machines, key-logger software could be installed to transparently capture user input on all monitored PCs and the authentication processing performed on servers. However, because many employees like to use their PC for occasional personal use –email, banking, stock market transactions, etc. – there are obvious privacy concerns with a key-logger capturing all input, including account numbers and passwords.

Although the organizations might say they can monitor their machines as they like, the employees could have strong objections. To increase user acceptance and ameliorate privacy concerns, monitored machines should be clearly marked as such, and unmonitored machines could be made available for employee personal use during lunch and break times. Nevertheless, privacy concerns remain.

2. Scope and Description

The scope of this study is to combine several biometric systems such as keystroke, stylometry, and mouse movements feature data, and then measure its performance.

Keyboard and mouse data can be captured through the key-logger. If the text information is desired for stylometry analysis at the linguistic word and syntax level, it needs to be obtained from the keystroke data. This text data for stylometry analysis can be obtained by converting keystroke input into text, and it is possible to come close to getting it correct except for mouse editing. For example, the input of characters and spaces presents

no problem. Strikes of the "backspace" and "delete" keys, however, will delete data and these data must be appropriately deleted. The conversion process would keep track of where the cursor is currently located because it can then be moved, for example, by the "home", "end", and arrow keys prior to hits of keys that delete data. The input of this program would be a file of 'keystroke' data and the output a file of 'text'.

Browsing on the Internet, such as performing Google searches, usually involves both keyboard and mouse input. This browser data and mouse feature vectors are provided by customer and/or other teams, while the 'parser' program that gets developed during this study would need to provide the 'text' to obtain the stylometry feature vectors. Thus, the browser data would appear to be ideal for combining keystroke, mouse, and possibly stylometry information, which can then help to run experiments to measure performance.

3. Methodology

As this multi-biometric system needs to combine keystroke features, stylometry features and mouse movement features, it is developed using agile methodology. The iterations planned are biweekly in length.

To understand on the step-by-step requirements, biweekly meetings were scheduled by the lead with the customer, and team used Lync and Skype as the mode for voice and data communication. At the end of iteration, the milestone deliverables were showcased to customer, to receive constructive feedback and review comments. Besides, periodic status reports were sent to all stakeholders (i.e. instructor, customers) to update on the work progress.

4. Functional Flow

[I] Program-1: Keystroke Parser

Objective

Parse the keystroke feature file for reading the keys and its key-code values to then output a file of 'text' (which would be used as input by the stylometry feature extractor).

Algorithm

- Read one XML or batch of XMLs.
- Iterate keycode.
- Determine if keycode is function key such as "Shift."
- Determine if keycode is a character.
- Store keycode in data structure.
- Repeat step 2-5 until end of file.
- Write to file with output string in "textinput" attribute.

Design Considerations

- XML file is read and parsed within a given folder.
- An output file, once generated, is not overwritten.

Design Assumptions

- One key will be pressed at a time.
- Input Domain
  !@#$%^&*()
  Alpha Numeric
  :;-+"
  Delete, Home, End, Enter, and Space
- Alpha characters are case-sensitive in XML. Therefore the program is ignoring shift keys.

Class Design

Metric.java

- Main method instantiates Interface object.

Interface.java

- createInterface method displays a GUI dialog that allows to click the 'Open Folder' or 'Open File' button. Then it instantiates the FileHandler class.
- On click of the 'Open Folder' button, the windows' dialog gets displayed, by calling its ActionListener.
- If the user clicks on 'Open File' button, the appropriate windows' dialog gets displayed, by calling its ActionListener from the Interface class, where-in the user can select the required input XML data File to be opened from the dialog window.

FileHandler

- The parseFile method parses the data from the XML file to extract the key and key-code values.
- Post parseFile method parses Key-stroke block from the XML file (which has #1 character in it), it is then sent to the OutputBuilder class.
- The writeToFile method of the FileHandler class saves the Text to an output file, in the same folder where the input files reside.

OutputBuilder

- Vector data structure to hold data.
- Provides several output option such as XML file, GUI, etc.

Test Result: Keystroke Parser

Parser program assists to parse the chosen feature file and interpret the keystroke data in text form.

Figure-1

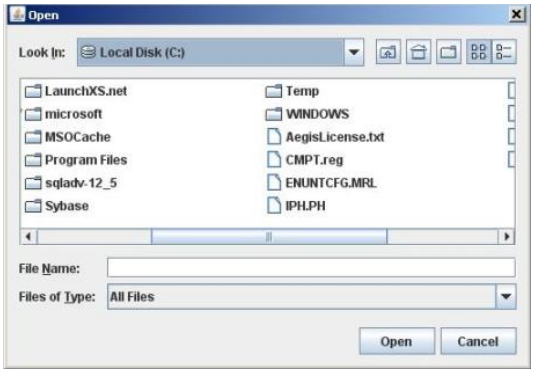

Figure-2

[II] Program-2: Assembler of feature files

Objective

To assemble the feature data from different biometric files (viz. keystroke, stylometry and mouse feature files) into one .CSV file.

- Select the biometric feature (keystroke, stylometry, mouse) for which the required files are to be opened.
- Choose files to be opened for each feature.
- Read each of the related feature records from the opened files, to concatenate the features per record and write into a single output file.
- Finally, the created output file would contain all the related data records from the three different feature files, delimited by a comma separator.

Class Design

Feature.java

- Main method instantiates Interface object

FileHandler.java

- Specifies the three input files (keystroke, mouse, stylometry feature vector files) and produce a single output features vector file.
- Reads the data records from each of these files, which have the same number of feature vectors per person with varying number of features. The input mouse, keystroke and stylometry feature files have data records that are comma delimited.
- Sets to four sections, the data records to be created in the Output file, such that the features are spliced together from each of the files per feature vector, and saves it in buffer. (For example if each file has 100 features from ten samples of John Smith, the output would have John Smith's ten samples each with 300 features.)
- Writes the data records with its sections from the buffer to the file, in the same format as in the input file, but combining each of the features per record delimited with a comma separator.

Interface.java

- Provides UI interface to select any of the 3 features.
- Provides UI interface for opening the files of keystroke, mouse, stylometry features.

 # Data Capture (Test result - Assembler)

This program basically has its input from the mouse, keystroke and stylometry feature files, to create a single output file that combines / concatenates data of related feature records. The output file has its data formatted with comma separated values.

Snapshot: One sample record from the combined o/p file

30
ALEX WEISMAN/?/?, ?, ?, Blah.Blah Blah Blah Blah, 1464, 0.6767011486137634, 0.124989916101354, 0.01619626171740155,

Each section is color coded above for better explanation, as listed below:
Section 1 (in Blue): Total record count in the output file.
Section 2 (in Pink): The related person data from feature files.
Section 3 (in purple): Total data count from all input feature files.
Section 4 (in green): Feature per data record combines from all input feature files.

Snapshot: To display feature files to be opened

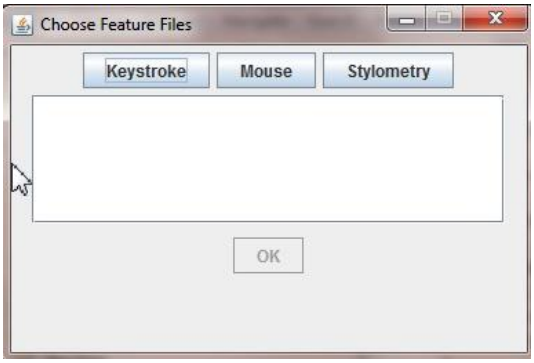

Figure-3

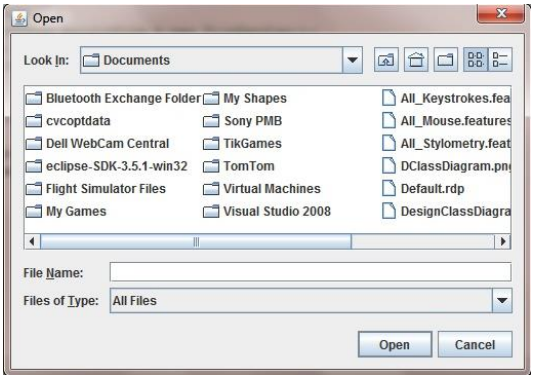

Figure-4

Figure-5

Snapshot: To display sample ouput .CSV file

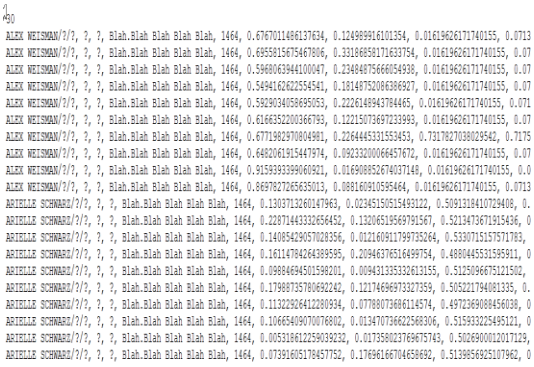
Figure-6

[III] Measure Performance

Experimenting steps

1. Create Test and Training files, by splitting the output obtained from 'assembler' program (such that first 5 records of a person is copied to 'Test' file, while next 5 records of that same person is copied to 'Training' file).
2. Input these two files to Biometric Authentication System, to generate the re-factored BAS output file.
3. This re-factored BAS file is then fed to the BAS Calculator which in turn generates an output HTML file.
4. For this output data, a ROC graph is plotted, to display the measure.

 # Result: Performance measure (Authentication system)

[i] Execution of BAS Accuracy Calculator

BAS Accuracy Calculator provides performance results.

Execute batch file (**Calculator.bat**)

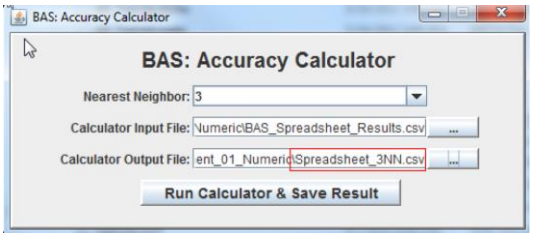

Figure-7

 # Result of BAS Accuracy Calculator (html)

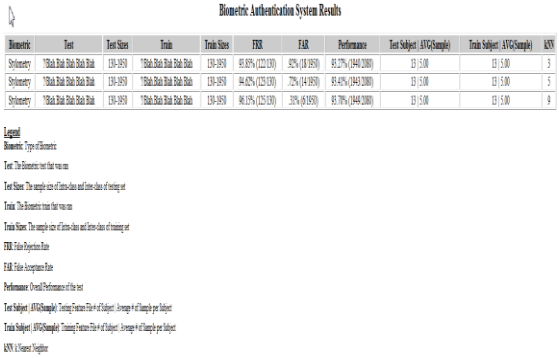
Figure-8

[ii] Execution of ROC data generator

The ROC Curve Data Generator creates Receiver Operator Characteristic (ROC) curves from non parametric classification data such as K nearest neighbor. This was pioneered at Pace University from faculty members Dr. Tappet and Dr. Cha and was the focus of Doctor Robert Zach's dissertation.

The ROC curves are a good way to show the tradeoffs between the False Acceptance Rate (FAR) and the False Rejection Rate (ERR) by indicating the Equal Error Rate (EER), all common measures found in biometrics.

Execute batch file (**ROC.bat**)

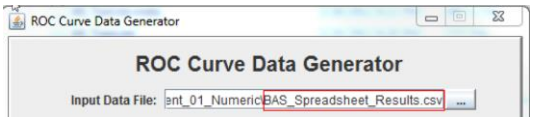

Figure-9

 # Result of the ROC data generator

(a) ROC graph - Keystroke feature

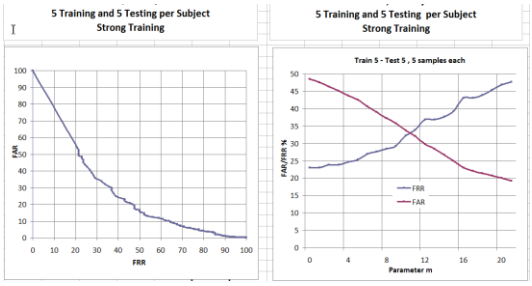

Figure-10

(b) ROC graph – Stylometry feature

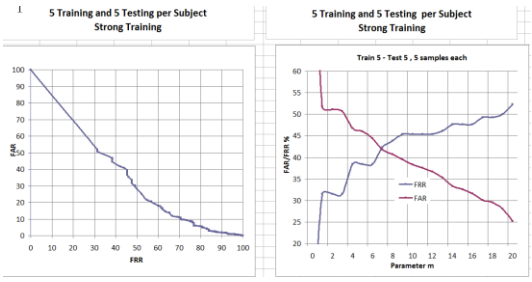

Figure-11

(c) ROC graph – Combo feature
(This includes Keystroke and Stylometry feature)

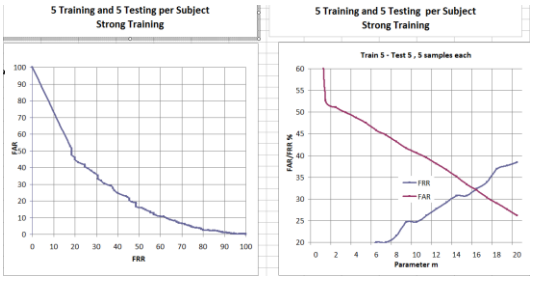

Figure-12

 # ROC Curve Derivation

Receiver operating characteristic (ROC) curves characterize the performance of a biometric system and show the trade-off between the False Accept Rate (FAR) and the False Reject Rate (FRR). In this study, the ROC curves were obtained by using a weighted procedure of the *k* nearest neighbors.

[IV] Program-4: Combining Biometric files

Objective

This Assembler program is used to combine the updated format of the Biometrics feature files (i.e. Keystroke, Stylometry and Mouse movements) into one single output .CSV file.

Execution

- Choose the biometric feature (i.e. keystroke, mouse or stylometry) for which the appropriate feature file(s) need to be opened.
- Choose files that need to be opened, for each selected feature.
- Repeat the above steps until at least two biometric feature vector files are selected.
- The program would then read the feature vectors from the opened biometric files. The feature file with least amount of specific vectors will be used as the base, while writing the combined vectors to a single output .CSV file delimited by a comma separator.

Source: File listing

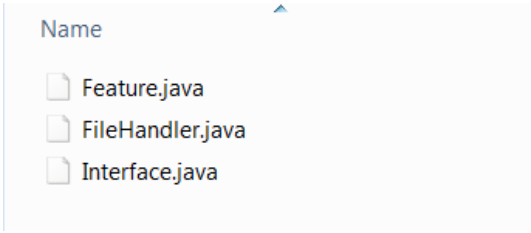

Figure-13

Feature.java

- Main method instantiates Interface object.

FileHandler.java

- Gets handle to a minimum of 2 input files (keystroke, mouse, stylometry feature files).
- Reads the feature vectors from each of these files. The input mouse, keystroke and stylometry feature files have vectors that are comma delimited. If the files have equal number of records then the features will be added, in the manner such that the file that is selected first from the UI dialog gets automatically treated as the base file. However, if the files do not have the same number of records then the file with the least number of records will be considered as the base file.
- Writes the feature vectors to an output file, in the same format as in the input file, but combining each of the feature vectors delimited with a comma separator.

Interface.java

- Provides UI interface to select the feature (i.e. keystroke or mouse or stylometry).
- Provides UI interface for opening the files of keystroke, mouse, stylometry features.

Execution Results

Program takes input from the chosen mouse, keystroke and stylometry feature vector files, to create a single output .CSV file that combines related feature vectors. The output file has its data formatted with comma separated values.

Snapshot: Input & Output file listing

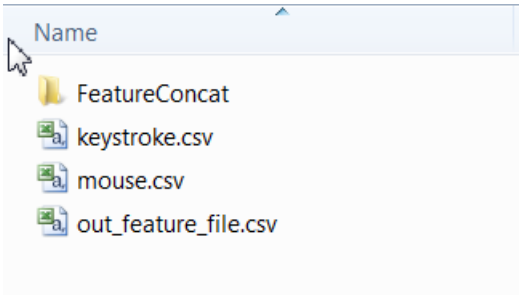

Figure-14

Snapshot: Execution steps

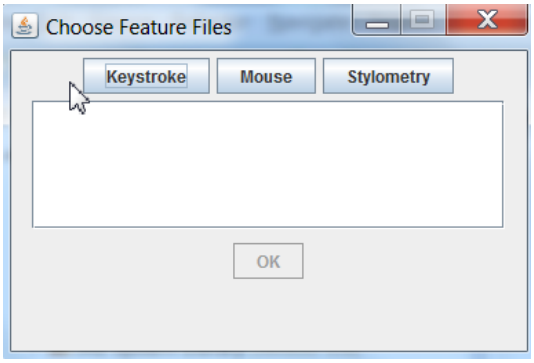

Figure-15

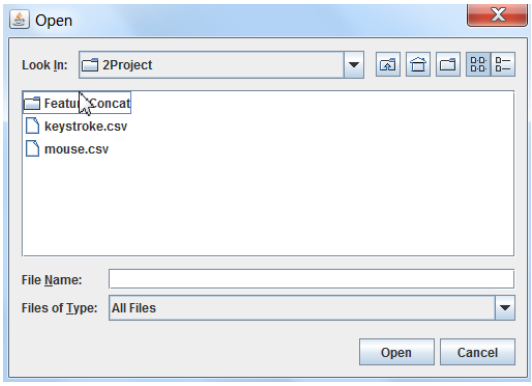

Figure-16

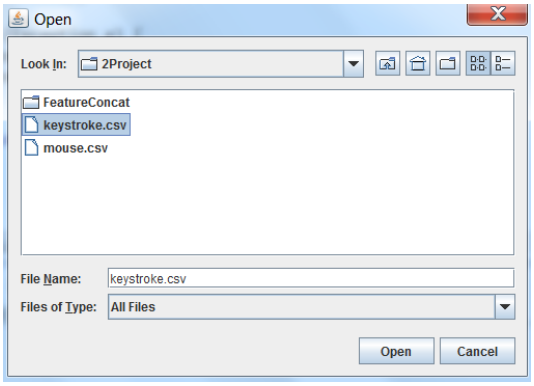

Figure-17

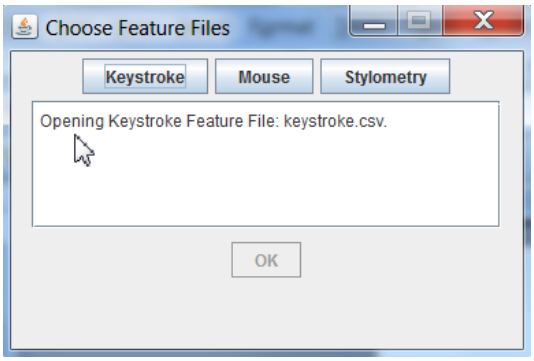

Figure-18

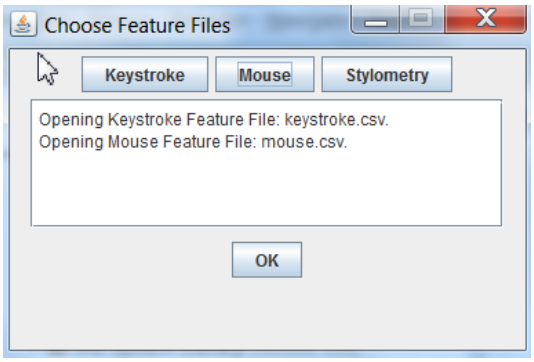

Figure-19

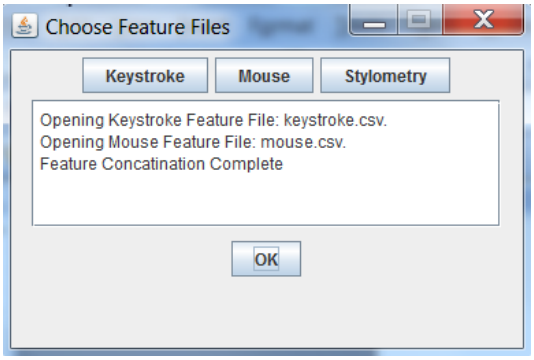

Figure-20

Snapshot: Sample Output .CSV file

Output .CSV file includes the combined feature vectors from all selected input files, and retains the same format as that of its input feature files.

```
ADAM FUSCO, 0.14917237427527197, 0.45547128546928123, 0.466132
ADAM FUSCO, 0.1707151263251986, 0.44999483861471954, 0.4768860
ADAM FUSCO, 0.1725370222887312, 0.4445973851465398, 0.48104804
ADAM FUSCO, 0.17523603738536023, 0.45093929567805713, 0.468949
ADAM FUSCO, 0.1779872052516918, 0.48589738160436596, 0.4710012
ADAM FUSCO, 0.18047468662206734, 0.46830253570753494, 0.454353
ADAM FUSCO, 0.1944562839929941, 0.48291595118120556, 0.4913429
ADAM FUSCO, 0.21499144267733242, 0.5366297239657563, 0.4949509
ADAM FUSCO, 0.2164131506548074, 0.5153571810428574, 0.48607587
ADAM FUSCO, 0.2167789545429072, 0.46311459915390657, 0.4794071
AL MIROFF, 0.08137456325648033, 0.4293897061175146, 0.38685047
AL MIROFF, 0.18727612865463272, 0.5095438734401793, 0.48108560
AL MIROFF, 0.1931142020413166, 0.49956743147052185, 0.53344828
AL MIROFF, 0.20800115211998443, 0.4696102100415164, 0.44839744
AL MIROFF, 0.21869106125017632, 0.47356346798840143, 0.5044922
AL MIROFF, 0.2202725660253587, 0.42790584191710884, 0.52135510
AL MIROFF, 0.22263569036835132, 0.4378891505936908, 0.52566734
AMAYAR AMAYAR, 0.2326364549806112, 0.5153180011178917, 0.52336
AMAYAR AMAYAR, 0.2370938651191464, 0.506132555173095, 0.482523
AMAYAR AMAYAR, 0.24184270886270917, 0.5213561425135821, 0.4908
AMAYAR AMAYAR, 0.28794230558199047, 0.5594630591931142, 0.4601
AMAYAR AMAYAR, 0.3007181690688951, 0.5604118970223344, 0.47255
```

Figure-21

5. Behavior Biometrics: Logger Program

Three sets of scenarios were developed to test the system: edit scenarios, browser scenarios, and game scenarios. One scenario of each type is presented below, as the others in each category are similar ones.

(a) Edit scenarios

Log-on to Logger application. Select Edit Paragraph option and then click on the Launch button.

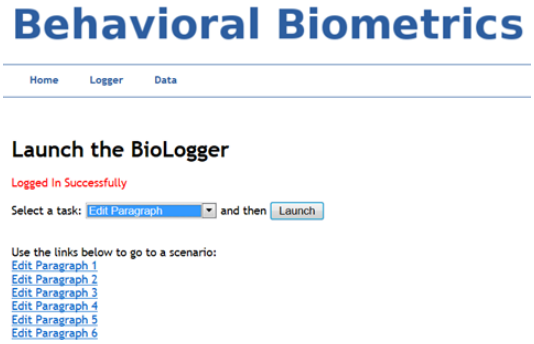

Figure-22

Edit Scenario #1:

Edit the paragraph below to match the one below it.

Hurricane Sandy hit the northeast coast of the united states in October. Sandy caused havoc across New York City and the boroughs. Storm surges causing flooding in streets, tunnels, and subway lines. This cut power in and around the city for extended periods of time. It took many weeks for power to resume on Long Island. The mayor estimates that the storm cost the city $19 billion. This storm caused devastation that will be remember forever by those affected.

Hurricane Sandy, also referred to as Superstorm Sandy, hit the Northeast coast of the United States on October 29, 2012. Sandy wreaked havoc across New York City and the boroughs. Storm surges caused flooding in streets, tunnels, and subway lines. Power was lost in and around the city for extended periods of time. It took many weeks for power to resume on Long Island. Michael Bloomberg, mayor of New York City, estimates that the storm cost the city $19 billion. This storm caused devastation that will be remembered forever by those affected.

(b) Browser scenarios

Log-on to the Logger application. Select the Browser Search option and then click on the Launch button.

Browser Scenario #1:

1. Go to Yahoo: http://www.yahoo.com/
2. Click on “Sports” (left menu)
3. Click on “MLB” (top menu)
4. Click on “Teams” (top sub-menu)
5. Click on “Boston Red Sox”
6. Click on “Team Report” for the Boston Red Sox
7. Go back two pages
8. Click on “New York Yankees”
9. Click on “Depth Chart” for the New York Yankees
10. Click on “Roster” for the New York Yankees (next to “Depth Chart”)
11. Click on “Sabathia, CC” (scroll if necessary)
12. In the search field above “CC Sabathia”, type “New York Yankees Captain” and click “Sport Search”

Exit tab or browser

(c) Game scenarios

Log-on to the Logger application. Select the Game option.

Game Scenario #1:

1. Log in at: http://vmonaco.com/biometrics/logger
2. Select “Game: Star Bubbles” from the task menu and then click on “Launch”
3. Once the application is running, minimize the application window so it will not interfere with you playing the game
4. Click on the Game: Star Bubbles link at the bottom of the page to access the game
5. Click on How to Play to review the rules of the game
6. When done, click on Back.
7. Click Arcade Game, first option on the list.
8. Click on Level 1 and play the game.
9. Exit the BioLogger when you finish playing; exit the BioLogger after each play.
   For each play, repeat the above as needed.

Data Capture (Test Scenarios – Logger program)

Test Samples	No. of Scenarios	Required Execution times of Scenarios	Completed by (Team-6 Members)
Edit Scenarios	6	8	Calvin, Enpaksh, Gowtham, Usha
Browser Scenarios	6	8	Calvin, Enpaksh, Gowtham, Usha
Game Scenario-1	1	8	Calvin, Enpaksh, Gowtham, Usha
Game Scenario-2	1	8	Calvin, Enpaksh, Gowtham, Usha

Figure-23

Events Captured in the Scenarios

(a) Keystroke events

press time: 'Key press timestamp'
release time: 'Key release timestamp'
key code: 'The key code'
key string: 'The symbolic key entered'
modifier code: 'Modifier code during the event'
modifier string: 'Modifier string during the event'
key location: 'The key location'

(b) Stylometry events

start time: 'Time the segment began'
end time: 'Time the segment ended'
text: 'Text entered'

(c) Motion events

time: 'Timestamp of the event'
x, y: 'X and Y location of the pointer device'
modifier code: 'Modifier code during the event'
modifier string: 'Modifier string during the event'
dragged: was the mouse dragged or just moved?'

(d) Click events

press time: 'Pointer click press timestamp'
release time: 'Pointer release timestamp'
button code: 'Pointer button'
press x, y: 'Start X & Y location of the pointer device'
release x, y: 'End X, Y location of the pointer device'
modifier code: 'Modifier code during the event'
modifier string: 'Modifier string during the event'
image: 'Region in which the event took place'

(e) Scroll events

time: 'Timestamp of the event'
amount: 'Amount of scroll'
rotation: 'Scroll direction (either +1 or -1)'
type: 'Scroll type (0 for unit 1 for block)'
x, y: 'X and Y location of the pointer device'
modifier code: 'Modifier code during the event'
modifier string: 'Modifier string during the event'

Behavioral biometric hierarchy diagram

- Motor control level – keystroke + mouse movement
- Linguistic level – stylometry (char, word, syntax)
- Semantic level – target likely intruder commands

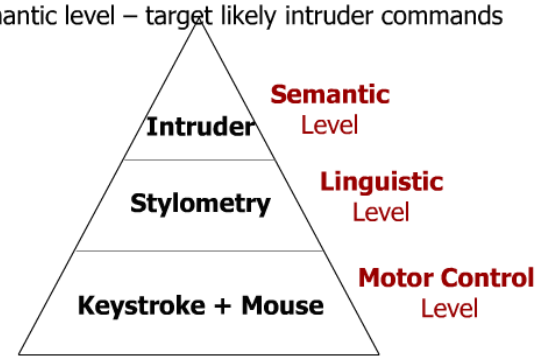

Figure-24

6. Conclusion

Post the study of the multi-biometric system, the assembling of the Biometric files for keystroke, mouse movements and stylometry features into a single output .CSV file stands completed (i.e. Assembler program).

Also, verification of all the test scenarios on the behavior biometrics stands completed using the Logger program.

7. References

[1] [Online] Available: http://www.csis.pace.edu/~ctappert/it691-12fall/projects/multi-biometric.htm

[2] C.C.Tappert, S.Cha, M.Villani, and R.S. Zack, "Keystroke Biometric Identification and Authentication on Long-Text Input," Int. Journal Information Security and Privacy (IJISP), 2010.

[3] R.S. Zack, C.C. Tappert, and S. Cha, "Performance of a Long-Text-Input Keystroke Biometric Authentication System Using an Improved k-Nearest-Neighbor Classification Method," Proc. IEEE 4th Int. Conf. Biometrics, Washington D.C., September 2010.

[4] S.Yoon, S-S Choi, S-H Cha, Y. Lee, and C.C. Tappert on 'the individuality of the iris biometric'. Int. J. Graphics, Vision & Image Processing, 5(5): 63-70, 2005.

[5] J.V. Monaco, N. Bakelman, S. Cha, and C.C. Tappert, Developing a Keystroke Biometric System for Continual Authentication of Computer Users, Proc. 2012 European Intelligence and Security Informatics Conf., Odense, Denmark, 2012, pp. 210-216.

[6] C.Tappert,S. Cha, M. Villani, and R. Zack, "A keystroke biometric system for long-text input," Int.J. Info. Security and Privacy, 2010, pp. 32-60.