Keystroke Biometric: Data Capture Resolution Accuracy

Michael Lam, Upen Patel, Mark Schepp, Teresa Taylor, and Robert Zack
Seidenberg School of CSIS, Pace University, New York, NY 10038, USA
mlam018@gmail.com, upenppatel@gmail.com, ms86530n@pace.edu, echevarriat@gmail.com, robert.zack@stpart.com

Abstract

This study extends work on the Keystroke Biometric Authentication System developed in Pace University's Seidenberg School of CSIS. This system can identify with a high degree of accuracy the typing characteristics that are unique to an individual. The system consists of three components: a Java applet which collects raw keystroke data over the internet, a feature extractor, and a pattern classifier. The aim of this study is to examine the available methods of capturing keystroke data, determine the accuracy of the data captured by the existing Java applet, and compare the accuracy of our system with the reported accuracy of other keystroke systems. In more than half of the 36 machines tested we found that the millisecond digit of the clock was frozen or inaccurate, resulting in a recording accuracy of centiseconds rather than milliseconds in those machines. Another finding was that the dwell times (key press durations) were roughly normally distributed and the flight times (from the press one one key to the press of the next) were roughly log normal distributed.

Introduction

Over the last several years, Biometrics have become increasingly popular in authenticating user identity [1,4]. Biometrics fall into two main classes: physiological and behavioral. Physiological characteristics such as fingerprint, voice, or iris traits can be used to identify specific aspects unique to each person. Behavioral characteristics associate with the learned behaviors of an individual [4]. This study, the previous work and developed system that is based on is concerned with behavioral patterns. Our area of work is focused on measuring the accuracy of a system developed at Pace University' Seidenberg School of Science and Information Systems to measure keystroke biometrics of a user for the purposes of authenticating identity.

Keystroke Biometrics is one of the least studied biometric applications employed for user authentication [1,4,7]. The majority of studies focused their attention on shorter text inputs such as passwords and user names [4]. One of the advantages of Pace's Keystroke Biometric System, and keystroke systems in general, is that they do not require the use of expensive and specialized computer equipment [4]. The system requires only that a user has a functioning keyboard, internet access and the appropriate java applet installed on their computer. The test-taker website has been configured to determine if the appropriate Java applets are installed on the computer connecting to it. If the correct Java applet is not installed, the site directs users to a hyperlink where that software can be downloaded. Another advantage is that the system is purely software based and can easily be incorporated with other existing authentication processes. This technology aims for the ability to authenticate test takers for online testing and harden passwords [4].

While the Pace University System uses this method of capturing keystroke dynamics through a Java applet and data capture over the internet, that there are alternative keystroke systems that employ other capture methods. These alternatives can involve systems and applications such as low-level assembly or Visual Basic [5]. One of the purposes of our work is to determine the accuracy of the Pace University system when compared with these alternatives.

Keystroke Dynamics are based on the fact that patterns of behavioral or learned actions can be classified and embedded into biometric keystroke systems in an attempt to monitor these actions for a continuous or short term period [1]. Understanding those patterns and shifts in the typing percentages are representative of factors that affect the results such as the dwell time and flight time [4]. Events are recorded twice per keystroke, upon the press (keypress) and release (keyrelease) of each key [5]. The "flight" time is measured as the time between the striking of consecutive keys. Flight time is calculated by subtracting the recorded press time for one key from the press time of the following key. The "dwell" time is defined as time as the time between the press and release of the key[4]. Dwell is calculated by subtracting the press time from the release time of a single keystroke [4]. Both are important in the verification of user identity [9]. All the keystroke systems we investigated reported that they measured the keystroke events in milliseconds.

The Pace University Keystroke Biometric Test Taker authentication system is composed of a set of interconnecting PHP scripts and a Java applet which performs the actual keystroke sample collection. The first is a web-based keystroke entry system. The keystroke program collects and measures typing characteristics of individual test-takers. In order to participate in the test, an individual must enter their personal data, which is stored in the demographic file. The second portion of the system is the Bio feature extractor, which is also referred to as the (BioFeature++). The third part of the system is a pattern classifier application, also known as the Biometric Authentication System (BAS) [4]. This study is focused on measuring the accuracy of the first of these components.

Environment and Data Sample

Our experiments and the conclusions drawn from this study are based primarily from two sets of experiments performed using the Pace University Keystroke Test Taker system. The first set of tests was performed in 2006, the second in Fall 2009. These results were also categorized by the type of keyboard being used to perform the experiment – specifically, whether the user was on a laptop or a desktop, because the different layout and size of the keyboards results in different dynamics in the keystrokes [4]. Where we were able, we noted the manufacturer of the machine used to take the test, since temporal resolution of keystroke capture programs have been known to vary by manufacturer and operating system installed on the machine taking the test [9].

The data from the tests taken in the Spring 2009 sample is larger than the Fall 2009 sample. There were 180 tests in the spring, and 46 taken in the fall. We had concerns about the much smaller sample size of the fall tests, and might have dismissed them in favor of focusing solely on the spring, but for the fact that we were only able to note the machine manufacturer for the smaller fall samples. The scope of our study was limited to tests taken on machines running a Microsoft Windows operating system because the web site hosting the test taker system during the course of our work was compatible only with Microsoft Internet Explorer (See Appendix III).

Accuracy Study for Millisecond Digits

Our first step in determining the accuracy of the Pace University Keystroke Biometric system was to examine some of the raw data files from the aforementioned tests taken on the system. There was concern voiced by previous developers that the program was not capturing the keystrokes in milliseconds, as intended, but rather in centiseconds. The reason for this suspicion is that a cursory examination conducted during development work on the system appeared to contain a disproportionate amount of appearances of the numeral 2 in the millisecond position.

In determining whether or not this was an error in the system, we examined the raw data files of the test results, and noted the number of appearances for each numeral logged in the millisecond position for the keypress and keyrelease times.

For the graphs below, the first bar in green represents the recorded keypress, the second in red denotes the keyrelease. Blue represents the dwell time. Along the horizontal axis, the numbers listed indicate the numeral appearing in the millisecond position for each record.

This data was derived from 180 tests taken by 36 different users on the Pace University Keystroke Biometric System in 2006. Five of these tests, all taken by the same subject, were dismissed from the sample. The reason we dismissed them was that, upon examination of the keypress and keyrelease times for this particular user, all keystroke times recorded in this subject's tests ended in the numeral 0. Upon examination of the other test logs, this occurrence was determined to be an anomaly limited to this one test taker. Our total sample for this data numbers 246,369 keystrokes.

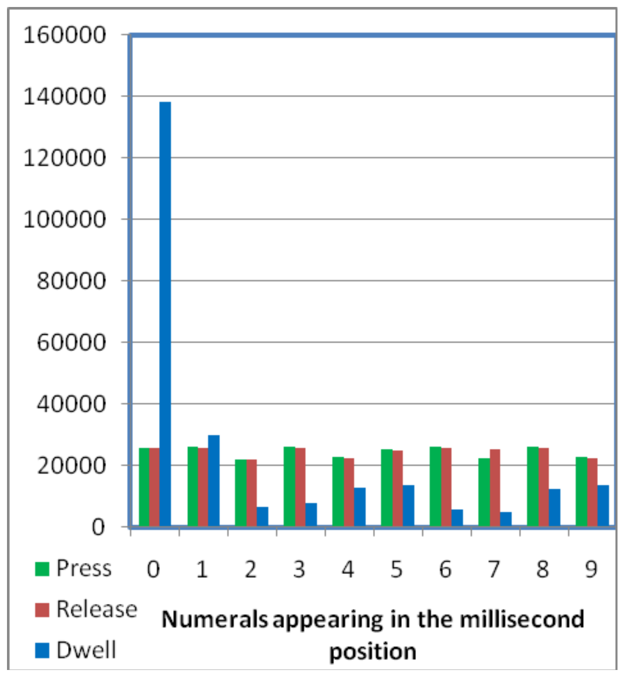

Figure 1. Chart of the millisecond digit appearances for all recorded press, release, and dwell times

Fig. 1 illustrates our findings. While the milliseconds recorded for the press and release times appear evenly distributed, there is a strong recurrence of the numeral 0 in the millisecond position for the dwell times. At first glance, this is curious, because of the fact that dwell times are derived by subtracting the keypress from the keyrelease. Logic would tell us that if the press and release millisecond digits were evenly distributed, the dwell time milliseconds would be, as well.

In order to take a closer look at these last digits recorded in the press and release column, we omitted dwell time from our next graph because the high recurrence of 0 in the first graph distorts the scale in Fig.1.

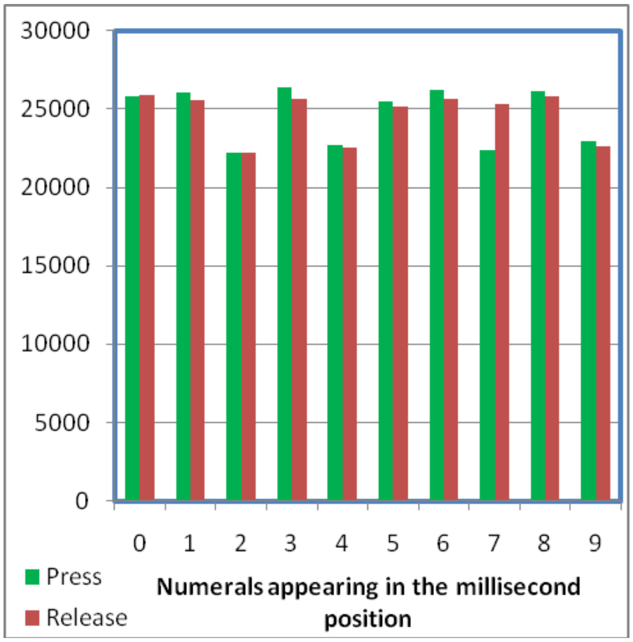

Figure 2. Chart of all millisecond digits for press and release only

Fig. 2, containing only the millisecond digit numbers for press and release records, illustrates a more evenly distributed recurrence of each digit. In the keypress records, no numeral is more than one standard deviation (sample has standard deviation of 1728) higher than the mean (24,636). The same can be said of the recorded release times – no numeral tally is higher than one standard deviation of the mean. Despite the recurrence of the 0 digit we saw in the dwell times (Fig.1), there does not appear to be a strong recurrence of any numeral in the records of the actual keystrokes themselves.

We then proceeded to divide our keystroke data by that which was entered on a laptop and that which was entered on a desktop.

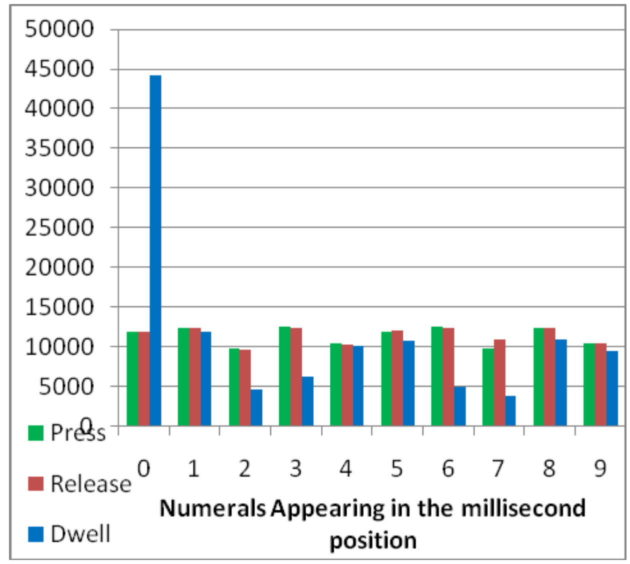

Figure 3. Chart of appearances of numerals in millisecond position – desktops only

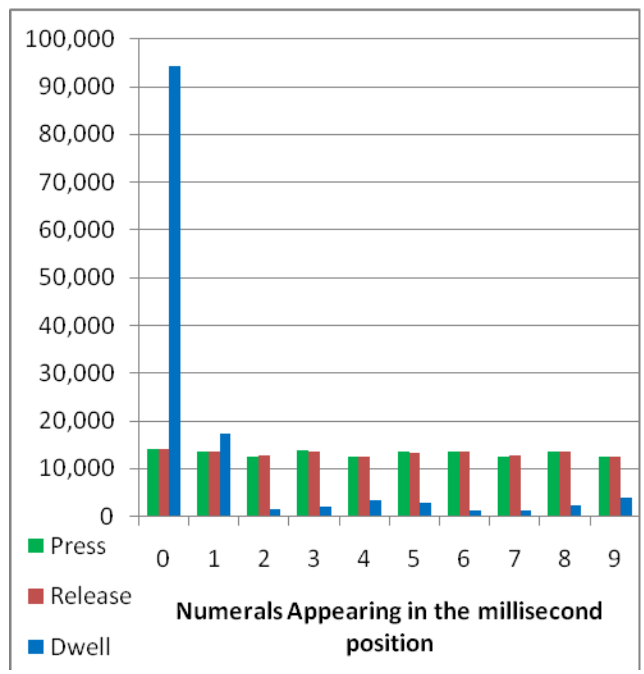

Figure 4. Chart of appearances of numerals in the millisecond position on laptop machines

Figs. 3 and 4 illustrate the breakdown by desktop and laptop respectively, with dwell and flight times recorded. All recorded press and release times for each of the 10 digits fall under one standard deviation above the mean.

We believe that the millisecond numerals recorded for flight and dwell times are evenly distributed, across both laptops and desktops.

Despite the even distribution of millisecond digit recorded in the press and release times in Figs. 3 and 4, we still see the strong recurrence of 0 in the millisecond position for both, just as we did in Fig. 1.

Before determining the cause of the millisecond digit recurrence in the dwell times, the next part of our study involved organizing the results by machine type. It is important to note that for this portion of the study, we did not use the same data sample as for the studies illustrated in Fig.1-4, for the reasons stated in the "Environment and Data Sample" portion of this document. The sample for the following was taken from the 46 tests taken in 2009 (as opposed to the 180 taken in the for the previous set of data). This is because the subjects had to be surveyed to determine the manufacturer of the computer on which they performed the test. The subjects from the earlier test sample were no longer available to survey.

Our sample sizes for each machine type were shrunken further by the varying types of machine each subject used to perform the tests. What follows are tables indicating the last digit count for press and release times on the three most commonly used machines.

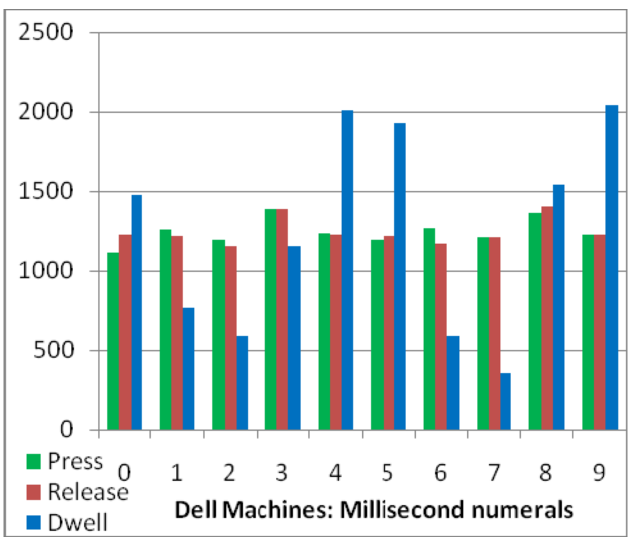

Fig 5. Appearances of numerals in the millisecond position on Dell machines

Although we are dealing in smaller sample sizes, we do not see much difference in the breakdown of the press and release millisecond digits when separating by machine manufacturer. The Dell sample illustrated in Fig.5 is the most even, and also composed of the largest sample size. We have not yet seen an instance of a millisecond digit, in either press or release times, exceed one sandard deviation from the mean. It is also perhaps interesting to note that the dwell times listed in this sample do not feature the strong recurrence of the millisecond digit 0.

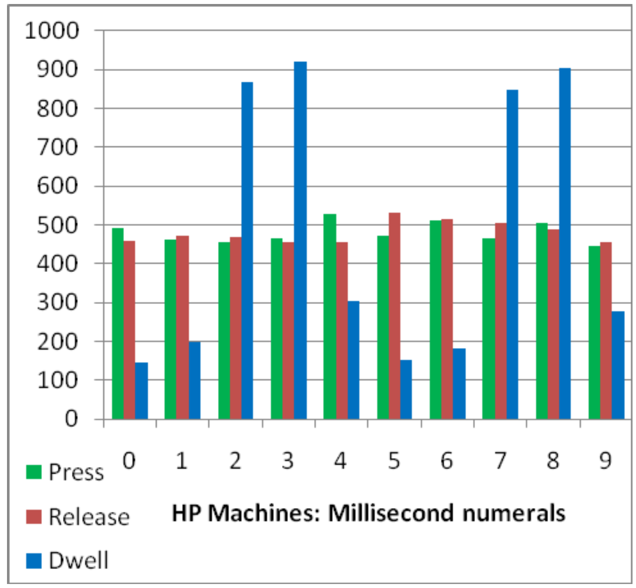

Figure 6. Appearances of numerals appearing in the millisecond position on Hewlett-Packard machines.

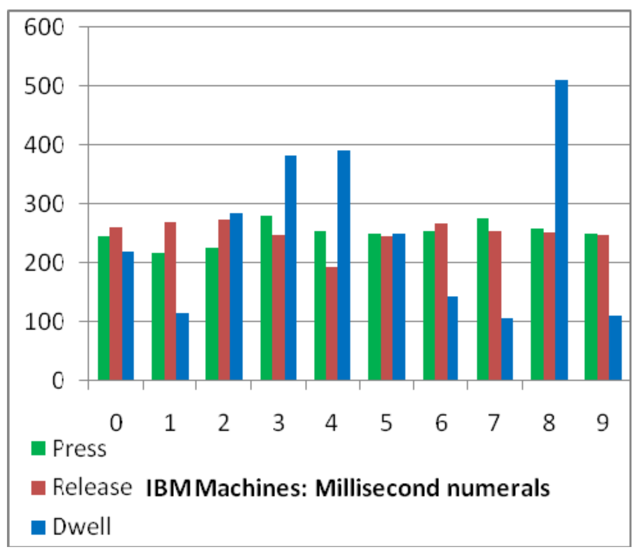

Figure 7. Appearances of numerals appearing in the millisecond position on IBM machines

In the Hewlett-Packard and IBM samples, illustrated in Figs. 6 and 7, we see no recurrence in any study exceeding what we would expect from a standard distribution of numerals 0-9 in the press and release times.

As in Fig. 5, the dwell times are again not as evenly distributed, although the 0 digit is not the most strongly recurring there, either.

Although we did not determine the same strong recurrence of the numeral 0 (as seen in the 2006 data) in the millisecond position for dwell times in the Fall 2009 data, there were other strong recurrences of note in the dwell times that will be mentioned in the context of the following section.

Distribution of Dwell and Flight Times

Our second group of experiments was designed to look at the recorded press and dwell times between 1 and 300 milliseconds (ms) recorded in our data samples.

Recorded flight and dwell times of less than 1 or greater than 300 were considered outliers and excluded from the study. The data is taken from the 180 keystroke tests on the Pace University System in 2006. The total sample for this portion of the study was 246,369 keystrokes. Of this total, 7,054 of those the dwell times recorded fell within our outlier categories of either less than 1 ms or greater than 300 ms, and do not appear on the graph.

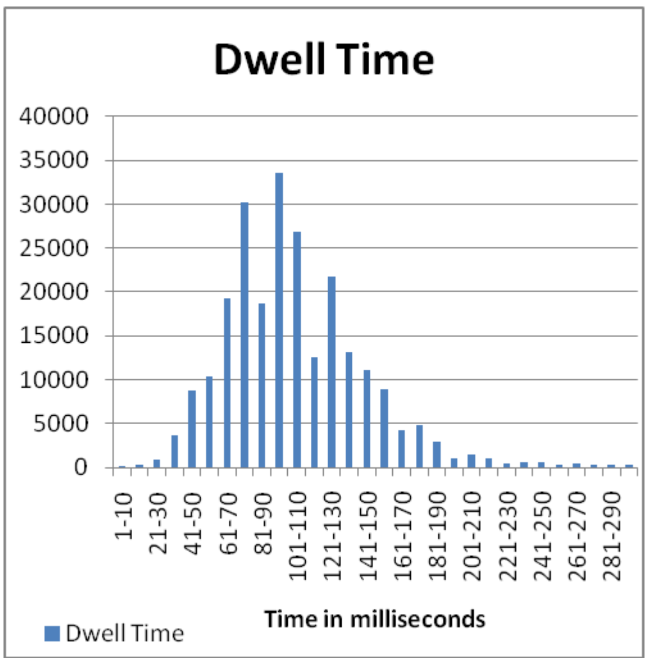

Figure 8. Histogram of all recorded dwell times

Bins for the preceding graph (Fig. 8) are sorted by ten. For example, the first bin measures all times recorded 1-10 ms, the second 11-20ms, and so on until 291-300ms.

Fig. 8 illustrates something close to a standard bell-curve with an apex in the 91-100 ms range for the dwell time.

The two bins representing the 81-90 and 111-120 ms ranges disrupt the pattern, falling shorter than they should given their position in the greater curve.

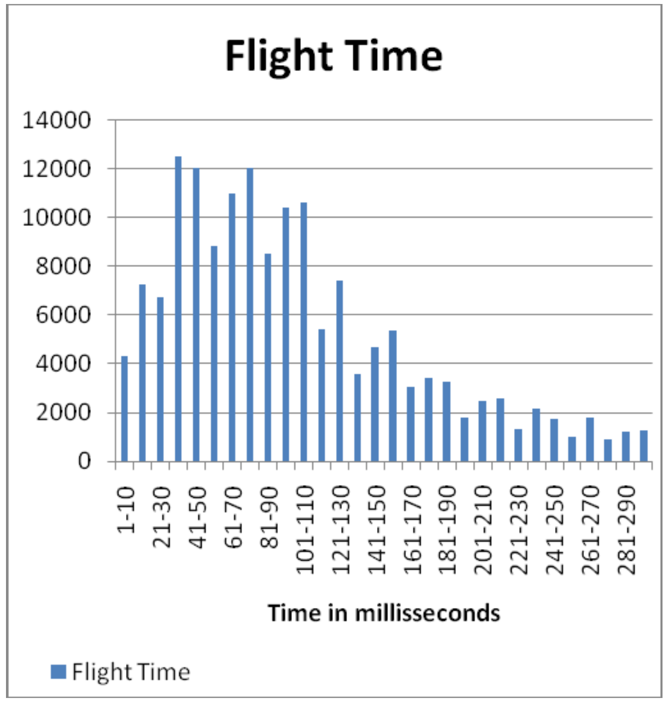

Figure 9. Histogram of all recorded flight times

Fig. 9, representing the flight times, is harder to classify. With an apex in the 31-10 ms range, the only pattern it appears to follow is a tail-off after the 120ms. As mentioned previously, 158,810 keystrokes out of 246,369 fell into the outlier category for this specific study. The reason for that is because when recording flight times, as a function of "nextkey" (the press time of the following key) minus the keyrelease, there is a tendency for negative numbers to result. This is because there are several common typing patterns that result in the one key being held down for the duration of the press and release of the following key – such as when one holds down the SHIFT key to change letter-case, or the ALT or CTRL keys to perform a hotkey function.

However, when examining the data further, we discovered what we thought to be an unlikely occurrence, and one that could possibly explain the high recurrence of the numeral 0 in the millisecond position for flight time illustrated in Fig. 1.

Specifically, while tallying the recorded dwell and flight times for each of the bins ranging 10ms in Figs. 8 and 9, we noticed that an inordinate amount of the recorded times ended in zero. For example, 80 percent of the recorded flight times in the range of 61-70ms was exactly

70ms, the majority of dwell times in the 91-100 bin was 100ms, and so on. This strong recurrence seemed highly unlikely in light of our data indicating that that the digit recorded in the millisecond position was much more evenly distributed (see Fig. 2).

Accuracy study by machine

A closer look at the recorded flight and dwell times indicated that in most cases, the press and release times for each recorded keystroke ended in the same digit. Since dwell time is simply function of keyrelease minus keypress, this resulted in a very high recurrence of dwell times ending in the numeral 0.

Approximately 51 percent of all dwell times recorded in this sample ended in the digit 0 – meaning that 51 percent of the time, the millisecond digit for press and release times for a single keystroke was the same numeral.

Again we need to stress that this high recurrence of 0 in the millisecond position for dwell times was found only in the 2006 sample. In the 2009 sample, which we used to examine the distribution of millisecond digits across different machine manufacturers (Figs. 5, 6, & 7), we saw recurrence of other digits besides 0. For example, in that sample we made the observation that in the range of 61-70ms for dwell time, 91 percent of the recorded times were 63ms. In the 121-130ms range, 85 percent of the times were recorded as 125ms. In the 41-50ms range, 82 percent of the times were 47ms. There were similar recurrences in several other ranges of 10ms.

While this smaller sample from the 2009 tests did not display the same recurrence of numbers ending in 0 for recorded dwell times that we see in Fig. 1, it did have other strong recurrences of specific times ending in various digits. This indicates that while the tendency to record the same millisecond digit in press and release figures from Spring 2009 was not repeated in the Fall 2009 tests, the system still contained strong recurrences in dwell times that are less easily observed and harder to categorize.

An examination of the individual logs from each test revealed that in some cases the same digit was repeating in the millisecond position. For example, in some cases the same millisecond digit would appear throughout the entire test, or would appear consecutively for several strokes before moving on to the next digit, which was then also repeated for several more strokes. In the 2006 data sample, we aso found that often the same numeral would be repeated in the millisecond position for both press and release records, which lead to the high repetition of the 0 digit in dwell times.

However, this did not occur across all machines. Some of the machines from the test sample featured an evenly distributed number of digits in the millisecond position. What follows is a breakdown of the number of machines that recorded high recurrences of certain millisecond digits, whether that occurred in the press, release, or dwell records. For the purposes of the study, we counted a high repetition as more than 1.5 standard deviations above the mean. The green column indicates the number of machines "working properly" or evenly distributing all millisecond digits. The red bar indicates the number of machines that were "stuck," as in containing a high recurrence of a single digit in the keystroke records. The yellow bar indicates a machine that did not have a high recurrence of a single digit, but a widely uneven distribution of several digits.

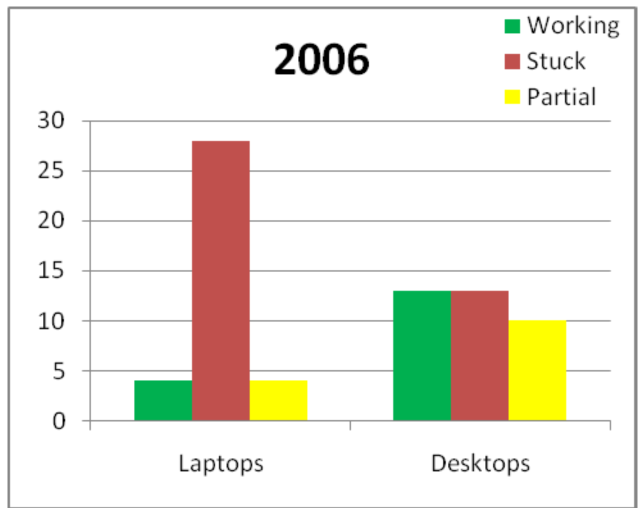

Figure 10. Number of machines accurately and inaccurately measuring milliseconds - 2006 sample

There were 72 machines from the 2006 sample – 36 laptops and 36 desktops. Fig.10 illustrates that a much larger number of laptops were stuck on a single millisecond digit. 28 out f the 36 laptops featured a high recurrence of a millisecond digit (roughly 78 percent). Almost all of these were the numeral 0 recorded for dwell time – indicating the same numeral was recorded in the millisecond position for press and release time in these cases. Four of the laptops contained an even distribution of all millisecond digits, and four were "partially stuck" – featuring an uneven distribution of more than one digit. Among desktops, we recorded 13 accurate, 13 stuck, and 10 partially stuck machines.

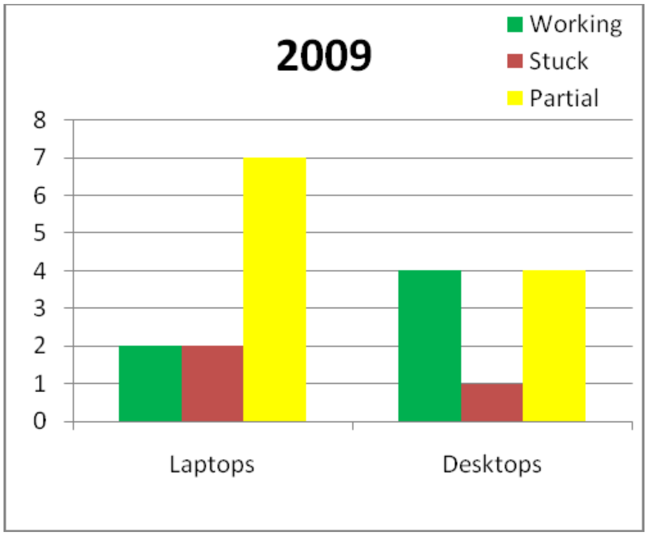

Figure 11. Number of machines accurately and inaccurately measuring milliseconds - 2009 sample

As noted earlier, the repetition of the 0 millisecond digit in dwell times from the 2006 sample was not repeated in the 2009 sample. More commonly found in the 2009 tests were machines that would repeat a specific millisecond digit for several keypresses before moving on to the another digit. While we were limited to a smaller sample of 18 machines (11 laptops, 9 desktops), we recorded two working, two stuck, and seven partially stuck laptops and four working, one stuck and four partially stuck desktops. Again, the desktops had a higher percentage of accurate clocks.

Alternative systems

When examining accuracy and capture methods of our own keystroke biometric system, it is important and informative to take a look at some of the other available programs. We will also be noting what method of keystroke capture they employ, and comparing the reported accuracy results with our own system here at Pace University.

The following are some relevant examples. We are comparing the existing FAR (False Acceptance Rate) and FRR (False Rejection Rate), along with reported temporal resolution and capture method. Given what we know about keystroke dynamics, the accuracy of the system's ability to measure keystrokes might influence the system's performance rates.

Comparison Table:

System	Capture Method	Temporal Resolutio n	FRR	FAR
KeystrokeID [3]	Java Applet	NR	3.0%	0.01%
bioChec [11]	Multiple methods	MS	3.0%	0.001%
Multiconference Experiment [5]	ActiveX Control	MS	15%	2%
BAKER [8]	Low Level Assembly	MS	0.1%	1.5%
Pace System [4]	Java Applet	MS	5.5%	2.6%

Fig. 12 - Comparison table of various keystroke capture programs

In Fig. 10, red text denotes a system designed for identification, and blue denotes one designed for authentication. Temporal resolution is listed were reported. "MS" stands for milliseconds and "NR" indicates and unreported figure.

According to our research, there are several methods of capturing this keystroke dynamic information. It is often done at the operating system level. In Windows, for example, it can be accessed and recorded in simple events involving the hook mechanism or keyboard driver.

The keyboard input on most PC hardware platforms is recognized by operating system as hardware interrupts where two events are recorded: a key is pressed (on Windows it is internal operating system message WM_KEYDOWN), and the key is released (MS Windows internal operating system message: WM_KEYUP) [9]. The timing of these events are believed to be accurate within 15 ms [9]. There are software programs written for Java, Visual Basic, and ActiveX that can specifically be installed on a machine to monitor and record these events.

This method of recording keystroke events as they appear in the operating system was the most widely employed by the systems we examined. Because most operating systems have a temporal resolution of milliseconds, this is in effect the most accurate resolution available for the majority of keystroke biometrics [6].

It should be noted that as we investigated these various coding methods, it was reported that researchers and testers believed that inaccuracies caused by delay at the software level were marginal when compared with lag time in the transfer of data between the keyboard and computer [5]. This is one of the main reasons why keystroke testing is organized by type of keyboard, and laptop/desktop. It is worth bearing this in mind as we discuss the accuracy of theses coding methods.

The Pace System does have the highest False Acceptance Rate of the above listed.

Unlike the others listed in the above table (excepting one configuration option for bioChec), the Pace system does not capture the keystroke events at level of the operating system. As we have noted, it captures the times recorded by a Java application over the internet.

Like the Pace system, "KeystrokeID" uses a Java Applet to capture keystrokes. While we were unable to determine where the capture actually occurs and is recorded in the applet, the system does require an installation of software, making it unlikely the capture is performed over the internet. KeystrokeID also has the highest reported accuracy listed. Although it should be noted that is a commercial product that has probably been tested and honed most rigorously and that the reported accuracy comes from their promotional materials and not an academic study.

"bioChec," a keystroke biometric authentication system developed by Checco Services, Inc., reports similar system performance rates. In a correspondence with our team, one of the officers of the company reported that bioChec could be tailored to employ several methods of keystroke capture. A Win32 application was cited foremost among methods used to capture the KEYUP and KEYDOWN events at the OS level, and the officer claimed they preferred to use a Java application when capturing the keystrokes over the internet. As this information comes from a Checco employee and promotional materials, the same reservations regarding reported accuracy and performance rates apply.

The unnamed system introduced by the multiconference experiment (2006 International Multiconference on Computer Science and Internet Technology) used an ActiveX control that interfaces with the DirectX function in Windows. Where DirectX integrates data from the machines program and systems, the ActiveX controls installed there by this system monitor keypress and keyrelease times as they are [5].

BAKER, a system designed by academics at the University of Southampton, reportedly uses a hidden function installed into the hooking system of Windows. There, these hidden functions record the keyup and keydown events as they pass from the keyboard driver to the system, along with the timestamp for each [8].

Conclusion and recommendations for future study

In the distribution of numerals appearing in the millisecond position, we have not seen a strong recurrence of any numeral in press or release times. We believe there is an even distribution of millisecond numerals across all tests for the press and release records.

We did observe a recurrence in millisecond numerals between press and release for the same keystroke in the larger of our two data samples. Fig. 1 illustrates the strength of the recurrence of 0 in the last position of dwell times. This occurrence is 2.8 standard deviations above the mean. This indicates a high recurrence of the same digit appearing in the millisecond position for press and release times of a single keystroke record.

In the smaller of our two data samples, we again saw no recurrence of millisecond digits in press or release, across either the entirety of the sample or in the breakdowns by manufacturer. And while we did see strong recurrences of specific dwell times recorded as we did in the larger sample, these were not limited to numbers ending in 0. A larger sample size may be desired for greater confidence in the results, and any differences in the configuration between the 2006 and 2009 test taker web applications and web site should be examined to determine possible cause. In either case, the Java applet may require refactoring to establish the certainty of a temporal resolution in milliseconds.

In addition, we observed a high percentage of machine clocks with inaccurate temporal resolutions. A cursory glance might indicate the temporal resolution on computer clocks are getting better, as the 2009 sample contained a much lower percentage of "stuck" clocks (roughly 28 percent against 57 perceent in 2006), but further study and a larger sample is required.

Although our data samples for the manufacturer breakdown was limited to systems running MS Windows, it might be informative to examine results in alternative environments. For this it would be necessary to configure the keystroke test taker system to work on browsers other than MS Internet Explorer.

References

[1] Notares, George. "Cheap Biometrics – use Keystroke Dynamics to Identify and Verify Users"

http://www.g-loaded.eu/2008/05/08/cheap-biometrics-use-keystroke-dynamics-to-identify-and-verify-users/ Last accessed on 12/9/09

[2] Monrose, Fabian and Rubin, Aviel. "Keystroke Dynamics as Biometric for authentication." http://www1.cs.columbia.edu/4180/hw/keystroke.pdf Last accessed on 12/9/09

[3] Probayes Corporation. "Bayesian Keystroke Dynamics Solution." http://docs.google.com/gview?a=v&q=cache:WIjsknBMo28J:www.probayes.com/IMG/pdf/Probayes_bkds_EN_0907.pdf+probayes+Keystroke+Dynamics+false+acceptance+rate&hl=en&gl=us&sig=AFQjCNFMcGMrDDzsckTQwurZeTpZZhhfnQ Last accessed on 12/9/09

[4] Proceedings of Student-Faculty Research Day, CSIS, Pace university, May 8, 2009. "Keystroke biometric Authentication System Experimentation."

[5] Adrian Kapczyński, Paweł Kasprowski, Piotr Kuźniacki. "Modern access control based on eye movement analysis and keystroke dynamics." http://www.proceedings2006.imcsit.org/pliks/126.pdf Last accessed on 12/9/09

[6] Gláucya C. Boechat, Jeneffer C. Ferreira, and Edson C. B. Carvalho, Filho. "Using the Keystrokes Dynamic for Systems of Personal Security" *World Academy of Science, Engineering and Technology 24 2006*

[7] Shanmugapriya, Mrs. D., Padmavathi, Dr. G. "A Survey of Biometric Keystroke Dynamics: Approaches, Security and Challenges" *International Journal of Computer Science and Information Security,Vol. 5, No. 1, 2009*

[8] Marsters, John-David. "Keystroke Dynamics as Biometric" http://eprints.soton.ac.uk/66795/01/thesis.pdf Last accessed on12/9/09

[9] Rybnik, Mariusz; Tabedzki, Marek; Saeed, Khalid. "A Keystroke Dynamics Based System for User Identification." 7th Computer Information Systems and Industrial Management Applications.

[10] Villani, Mary, et al. "Keystroke Biometric Authentication and Verification on Long-Test Input" Behavioral Biometric for Human Identification

[11] bioChec FAQ http://www.biochec.com/FAQ.htm Last accessed on 12/9/09

Appendix

Source code for the Pace University Keystroke Biometric Java Applet:

```
import javax.swing.*;
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
//import java.security.*;
//import java.rmi.*;

public class KeySpeed16_Applet extends Applet
 implements ActionListener
{
  private int APPLET_WIDTH = 700;
 private int APPLET_HEIGHT = 420;

private KeySpeed16 keySpeed;

public void init()
  {
    java.net.URL url = this.getDocumentBase();
    String query = url.getQuery();

    //query = "Hug&Hort&Fable&1&kb&pc";    //test
purposes
    try {
      String[] user = query.split("&");
      keySpeed = new
KeySpeed16(user[0].toUpperCase(),
user[1].toUpperCase(),
                user[2].toUpperCase(),
user[3].toUpperCase(),
                user[4].toUpperCase());
      add (keySpeed);
      setSize (APPLET_WIDTH, APPLET_HEIGHT);
    }
    catch (Exception e) { }  }
  public void actionPerformed (ActionEvent event)
  {  //System.setSecurityManager(new
RMISecurityManager()); }}
```

Raw data from keystroke Test Taker Results:

Spring 2009: http://utopia.csis.pace.edu/cs691/2008-2009/team4/ Last accessed on 12/9/09
NOTE: Logged under side tab "Raw Data"

Fall 2009:
http://utopia.csis.pace.edu/cs691/2009-2010/team4/team4/index.html Last accessed on 12/10/09

Pace University Keystroke Test Taker System:
http://utopia.csis.pace.edu/cs691/2009-2010/team4/Team4/2005-2006/team4/ Last accessed on 12/12/09