Leveraging Smartphone Hardware Capabilities for Alternative Authentication

Isaac Long

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2014-95
http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-95.html

May 16, 2014

Acknowledgement

I would like to thank Ty Lee, the Capstone project's industry advisor, for providing guidance and feedback throughout the course of the project. I would also like to thank his colleagues, Francis Hsu and Saurabh Tewari, for also providing guidance and feedback, especially during the initial stages of the project.
I would like to thank Don Wroblewski, the Capstone project's faculty advisor, for providing guidance throughout the course of the project, and for reading and providing feedback on this thesis.
I would like to thank Laurent El Ghaoui, my concentration advisor, for reading and providing feedback on this thesis.
I would like to thank Te-Yuan Huang, one of the creators of MoviSign, for

allowing us to use the MoviSign code as a foundation for our project.

University of California, Berkeley College of Engineering

MASTER OF ENGINEERING - SPRING 2014

EECS

SPCOM

Leveraging Smartphone Hardware Capabilities for Alternative Authentication

Isaac Long

This **Masters Project Paper** fulfills the Master of Engineering degree requirement.

Approved by:

1. Capstone Project Advisor:

Signature: _____ Date _____

Print Name/Department: Don Wroblewski/Fung Institute

2. Faculty Committee Member #2:

Signature: _____ Date _____

Print Name/Department: Laurent El Ghaoui/EECS and IEOR

Abstract

In this project, we explore two forms of user authentication, alternative to the alpha-numeric password, through utilization of various hardware components in a modern smartphone. These components act as input channels for biometric and geolocative data, which can be used to validate the identity of a user through gesture-based authentication (via the accelerometer and orientation sensor) and location-based authentication (via the WIFI sensor). We develop an Android application that incorporates these two authentication methods. We implement two different gesture-based authentication algorithms adapted from existing work done in this field of security research. We find that one algorithm is more convenient to train and implement than the other algorithm, but also performs more poorly in terms of accuracy compared to the other algorithm. We also implement a location-based authentication algorithm based on scanning the WIFI access points around the user's current location. We introduce thresholds into the algorithm to provide tolerance and robustness against inconsistent WIFI access points, and these thresholds show improvement in the algorithm's performance.

Introduction

User authentication is an important component of any Internet service today. With the wealth of sensitive and personal information stored online, an authentication system that is robust and easy to use is thus crucial to ensure security and confidence among consumers. Currently, the alphanumeric password is the most ubiquitous form of authentication; however, it is hardly perfect, and still poses many challenges in terms of usability and robustness [1]. Hence, the problem of improving or finding a better form of user authentication is a pressing one for many Internet companies, and is also the aim of many products in the current market.

There has been much research exploring alternative measures to validate user identity. One approach is to leverage on biometric data. Biometrics functions as good identifiers because they are (ideally) unique to an individual, and thus are more reliable than knowledge-based methods of authentication such as the password [2]. An example of a strong biometric is the fingerprint, a physical biometric which is currently used by many governments and immigration agencies to identify a person on his passport or Personal Identification card. Usage of biometrics requires some form of input device to extract the required data; in the case of a fingerprint, many airports and immigration offices have a fingerprint scanner for such a task. Thus, to leverage on biometric identifiers in user authentication of any Internet service, users must have convenient access to such input devices.

The modern smartphone comes with a whole host of sensors and devices[1] that allows for the input of not only biometric data, but also other forms of behavioral identifiers. Given the ubiquity of smartphones – 60% of all mobile phone owners currently own a smartphone [4] – the convenience of utilizing smartphone hardware capabilities as authentication input devices is there.

Indeed, several smartphone companies have already incorporated built-in apps that leverage on these sensors to provide additional security. The iPhone 5s comes with Touch ID, a security app that allows the user to train the iPhone to recognize his fingerprint [5]. The Android 4.0 operating system also comes with Face Unlock, another security app that allows the user to train his Android phone to recognize his face. Both these apps leverage on existing hardware capabilities (the fingerprint scanner and the camera, respectively) to provide an alternative security feature that replaces the traditional approach of entering

[1] According to [3], a modern smartphone has a gyroscope, magnetometer, accelerometer, camera, GPS, humidity sensor, microphone, pressure sensor, proximity sensor, temperature sensor and WIFI sensor.

a passcode to unlock the smartphone. These two apps are good examples of how to take advantage of the hardware of a smartphone to devise alternative authentication means.

For our Capstone project, our goal was to continue to explore this space, with the aim of building the foundation for developing a novel smartphone app that leverages on smartphone capabilities for authentication. Ideally, the authentication would be for a web service such as Yahoo, where users login to the service with a username and password. Our approach was to firstly look at research done on more unique and less popular forms of authentication, and to develop and synthesize the key ideas from the research. We would implement these ideas in our app, and evaluate the strengths and drawbacks of the implementation accordingly.

Our Capstone project borrows heavily from MoviSign, an app developed by Stanford researchers [7] that relies on the accelerometer and orientation sensor on Android phones. As the name of the app suggests, MoviSign allows users to train their phone to recognize their signature. This is achieved by holding on to the phone as a pen and signing in the air, as though the air was the paper; the accelerometer and orientation sensors keep track of the movement of the phone, thereby translating the signature movements into a time sequence of data, or a discrete time signal. A person's signature acts as a behavioral biometric, and can thus be used to identify him.

We also looked at other research done on the accelerometer, and found UWave [8], a similar accelerometer-based gesture recognition app. UWave differs from MoviSign greatly in terms of the algorithm used to train the app, and the implementation of the app. A fundamental issue with such authentication apps is the accuracy and reliability of the app; how often does the app fail to recognize user, or incorrectly recognize a different user? Another issue is the problem of convenience; how much effort does it take to train the app to recognize the user? Both UWave and MoviSign had different approaches to these two issues, resulting in two different algorithms that had different strengths and limitations. We implemented these two algorithms separately in our app development process, and assessed the performance and weaknesses of either algorithm.

Another approach we took was to examine the use of geolocation as a behavioral biometric. That is, if a user who frequently logs in from a certain location, such as his home or his workplace, that user will likely continue to login from that location in future. As such, gathering information about his location and using that as an additional factor in authentication seemed like a reasonable approach. Indeed, we

based our idea off of Google's development of a Wi-Fi positioning system[2] [9]. We leveraged the use of the WI-FI network sensor – an essential hardware component of any smartphone with access to Internet – to obtain a "WI-FI signature" of a user's location. This signature is essentially a list of WIFI access points that the user's phone can detect, which should ideally be unique to that user's location. As such, we could use this additional information as an extra layer of authentication.

[2] Google collected information about the IDs and strengths of WIFI access points at various locations, similar to how they collect the pictures for Google Maps. As such, the composition of WIFI access points at any given point on a map can be used to identify one's location.

Methods

As mentioned in the Introduction, our approach to this project was to examine existing research done in the field of alternative authentication methods, and to attempt to improve on and synthesize the various ideas into a novel app. Given the many operating systems available for smartphones, we decided to narrow the scope of our project to developing an app for the Android platform, primarily due the team's familiarity with the programming language (Java-based). There was also a wider variety of Android-based smartphones available (Samsung, LG, Huawei), and thus procuring the smartphones for testing would be easier. For the project, testing was done on a Samsung Galaxy Nexus running on the Android 4.3 (Jelly Bean) operating system.

Accelerator and Orientation Sensor, MoviSign

After a round of initial browsing of research papers and ideas, we decided to focus on developing gesture-based authentication. This was achieved through two sensors on the smartphone: the accelerometer, and the orientation sensor. The accelerometer sensor recorded the acceleration of the phone at unit time intervals in the X, Y, and Z axes in 3D Cartesian space (see [10] for more details). The orientation sensor recorded the current azimuth, pitch and roll of the phone in 3D Cartesian space; or in other words, the angle that the phone made with respect to the X, Y and Z axes (see [11] for more details). Thus, by using the phone as pen, a user could sign his password in the air; the accelerometer and orientation sensor essentially converted the movements the phone made during the signature into 6 channels of time series of data points (X, Y, Z, azimuth, pitch, and roll).

In particular, we found open-source code for *MoviSign*, an Android app developed by Stanford researchers that utilized such an algorithm [7]. *MoviSign* used the Support Vector Machine (SVM) technique for Machine Learning to train the phone to classify authentic and non-authentic signatures. Data collection was done on an Android phone using the accelerometer and orientation sensor as described above. The Stanford researchers divided up each of the 6 channels of time series data into 32 equal segments and took the average of each segment, resulting in 32 values for each channel, and a total of 192 values for all 6 channels. These 192 values were then used as a 192-dimension feature vector in SVM training – this was done in *CVX*, a convex optimization program in MATLAB.

Accelerator and Orientation Sensor, SVM Algorithm: Test 1

Our team used this code as a foundation, and adapted it in several ways. Firstly, instead of CVX, we wrote the SVM code in *liblinear*, a Machine Learning program for MATLAB. Secondly, we preprocessed each channel of raw data with a simple moving average filter of size 3 to reduce the impact of sensor noise, which the accelerometer is prone to have [12]. We also applied minor feature scaling to data for more accurate SVM training, and also set an arbitrarily-chosen minimum requirement of 3 seconds of input time, to ensure that signatures had sufficient entropy. For testing, we collected a total of 223 sets of input data from 7 different users, asking them to sign their first name with the smartphone; each user's signature was different from the others. For the SVM training, signatures belonging to a certain user were used as positive training examples for that user, while signatures belonging to the 6 other people were used as negative training examples. There were also 57 random and irrelevant signatures used as negative training examples, for the purposes of training a more robust SVM.

During the data collection phase, we discovered that the process of repetitive input of signatures was a tedious and tiring one. This was a weakness of the SVM method; the SVM required a relatively large amount of training data to produce an accurate classification algorithm. A casual user of such an authentication method, especially in the context of a Yahoo user trying to create an account, would be highly frustrated with the need for so many inputs just to set up the authentication infrastructure. It was noted that the users with lower number of training examples had a lower accuracy.

Another problem with the SVM method was the fixed feature vector size. Since each time series was divided into 32 segments, signal information would definitely be lost (especially for higher frequency components of the signal); longer signals would lose even more information, which would reduce the reliability and accuracy of this authentication scheme. Given that the raw input signals gathered in our test data ranged anywhere from 2000 time steps to 4000 time steps, this potential and imbalanced loss of information presented a problem.

Accelerator and Orientation Sensor, UWave and Dynamic Time Warping

Because of these two limitations, we sought to incorporate the algorithm used by UWave, a similar gesture-authentication program implemented in C on multiple platforms (Lenovo T60, MDA Pocket PC, Rice Orbit Sensor) [8]. UWave only utilized the accelerometer on the input devices, but was based on different design principles, such as quantization of data, and template adaptation. The key algorithm behind the design of UWave was the use of Dynamic Time Warping (DTW), a dynamic programming

algorithm that measured the similarity between two time series signals. DTW took two input signals that possibly varied in length, and output a positive "distance value" (or "similarity score") signifying how similar two signals were; the lower the distance value, the more similar the signals (see Image 1 below). DTW was especially useful for comparing time series with discrepancy in their alignments (for example, if one of the signals was stretched in time, while the other was compressed in time) [13]. Since the signatures were essentially 6 channels of time series signals, DTW was applicable to processing the signatures as input signals.

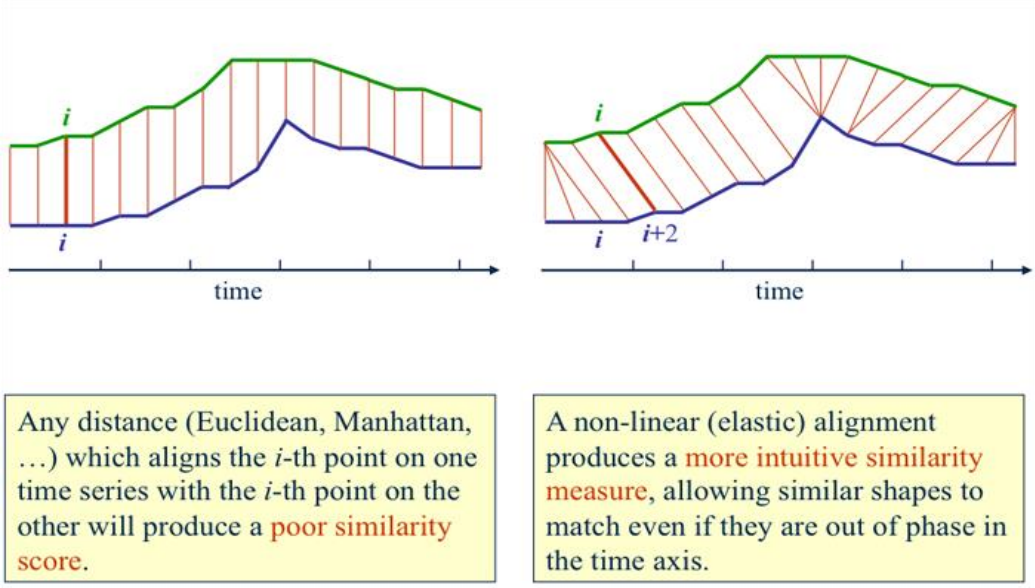

Figure 1 courtesy of [18]: DTW uses a non-linear alignment (right) for a more intuitive similarity measure

Thus, DTW seemed like a good solution to the two problems that MoviSign was experiencing. Firstly, unlike the large amount of training input required by SVM, users would only need one input to set up a model signature; subsequent authentication attempts would only require the DTW to compare the new input with the model signature to return a distance value. Secondly, DTW would be able to handle varying lengths of input signals, without the need to truncate the signal into a fixed number of segments. At the same time, UWave's implementation was only done on the 3 channels of data from the accelerometer (X, Y, Z), thus the inclusion of the orientation sensor (3 additional channels of data: pitch, azimuth, roll) could possibly provide additional information about an input signature that increased the accuracy of the classifier.

Accelerator and Orientation Sensor, DTW Algorithm: Test 2

We thus implemented a DTW algorithm in MATLAB, based on open source code found in [14]. Our team needed to determine a distance value threshold to be used to classify authentic and non-authentic signatures. Ideally, signatures belonging to the same person should have relatively low distance values, while signatures from two different people should have high distance values. We hoped the data collected would reflect that hypothesis by presenting a bimodal distribution with a clear threshold to separate either mode. To find this threshold, we collected between 10 to 12 signatures per person from a new batch of 7 people, using the adapted MoviSign app. We obtained a total of 76 sets of input data, each with 6 channels of input signals (X, Y, Z, azimuth, pitch, and roll, as defined in the previous section). All combinations of pairs in the data (76 choose 2) were passed as input into the DTW algorithm in MATLAB to obtain distance values for each pair; each channel in each data set was only compared with the same channel in another data set. Based on the distance values, we examined the means and standard deviations of each user's data, and experimented with using multiples of half a standard deviation from the mean as various thresholds for a classifier.

WIFI Sensor

We decided to add an additional dimension of authentication in the form of a WIFI signature. Mobile users who logged into Internet websites tended to do so regularly from the same locations – for example, a user's home or workplace. This could be considered a behavioral biometric; indeed, many Internet services already monitored the geolocation of users based on the IP address of their recent logins, and flagged suspicious logins from locations that deviated drastically from this norm (for example, a sudden login from an IP known to be in another country or continent). Based on the assumption that the WIFI access points around a user's home or workplace were relatively unchanging, the list of WIFI access points could be used as a signature to leverage a similar mechanic; since users would login frequently from their home or workplace, the WIFI access points associated with the logins would theoretically be mostly the same each time. This mechanic would thus be used to authenticate the location of the user, which would in turn support the authentication of the user's identity.

Using the built-in Android WifiManager class [15], we implemented a WIFI scanner (incorporated into the existing MoviSign app) that would register and save the list of WIFI access points that the smartphone could detect. Each access point's SSID, BSSID, capabilities, frequency and level (signal strength) was recorded; the combination of these elements would theoretically make an access point

uniquely determined[3], much like a human's fingerprint. The user was required to link each list with his name and the name of the current location. Thus, when a user later wanted to authenticate himself at the same location, the WIFI scanner would search for the stored list with his name and location, and compare the list with the current list of WIFI access points that the smartphone scanned. As long as both lists contained a certain percentage of identical access points[4], the location would be considered authenticated as a match.

WIFI Sensor, Signal Strength Threshold: Test 3

A problem with this approach was that certain weaker WIFI access points tended to occasionally disappear and reappear during WIFI scans, thus negatively affecting the integrity and consistency of the lists. For example, if a saved WIFI list consisted of mostly weak WIFI access points, a user may have trouble authenticating the current location due to the inconsistent appearance of these WIFI points, resulting in a discrepancy of matched access points. As such, a threshold of signal strength needed to be determined, so that weaker, inconsistent access points could be ignored by the scanner. To determine this threshold, we performed WIFI scans at 10 different locations (5 apartments, 3 campus locations, and 2 off-campus cafes) at various times in the day. Each location was scanned 10 times. A set of reliable and "consistently seen" access points was determined based on the access points' appearance in all 10 scans, and the remaining access points were considered unreliable. Since signal strength tended to vary for access points in both sets, the average signal strength was determined for either set, to identify an appropriate threshold of signal strength to distinguish reliable and unreliable access points. After determining an appropriate threshold of signal strength, we altered the WIFI scanner to ignore access points that had signal strengths below the threshold.

WIFI Sensor, Match Classification Threshold: Test 4

With this new algorithm, we redid the test at 8 new locations (3 apartments, 4 cafes and 1 campus location) to verify the signal strength threshold, as well as to select an appropriate match classification threshold of identical access points to classify a location as a match. That is, what percentage of the access points that appears on both lists must there be before the algorithm considers both locations to be the same? Each location was scanned 10 times (with the new signal strength threshold) for a total of

[3] A BSSID is ideally unique to each router. [16]

[4] For the purposes of our app, two access points were considered identical if they had the same SSID and BSSID. Determining if the capabilities and frequency of the access points matter in deciding if two access points are identical is an area for future work.

80 data points; each data point was tested against all other data points (including itself) and classified as a match or not a match, and the accuracy was calculated for varying match thresholds.

Results

Test 1: Accelerator and Orientation Sensor, SVM algorithm

The first test was done on the signature recognition function of the app, using only the team's implemented SVM algorithm for Machine Learning. The algorithm tried to classify each sample as a true sample or a false sample for each user. Table 1 presents the number of True samples of each user and the corresponding accuracy that the app has for that user. Accuracy is defined as the percentage of correctly identified samples out of the total number of samples (223). Note that the number of true samples does not add up to 223, as 57 random and irrelevant signatures were specifically used as false training samples.

User Name	No. of True Samples	Accuracy
Chirag	34	98.6%
Anna	62	91.03%
Isaac	39	99.1%
Jess	5	99.1%
Mandy	9	99.1%
Wong	7	99.1%
Tom	10	97.8%

Table 1: Number of true samples and accuracy for each user out of 223 total test samples.

From the data, we see that most users have between 97-99% error rates. Anna's test set was the anomaly, having a large number of true samples, but also having a 91% error rate. To get a better idea of the performance of the SVM algorithm, the team calculated the recall and precision. Recall is defined as the percentage of correctly identified true positives out of the total number of true samples for the

user. Precision is defined as the percentage of correctly identified true positives out of the total number of samples that were identified by the algorithm as a true sample.[5] Table 2 presents these numbers.

User Name	No. of True Samples	Recall	Precision
Chirag	34	94%	97%
Anna	62	92%	80%
Isaac	39	100%	95%
Jess	5	60%	100%
Mandy	9	100%	82%
Wong	7	71%	100%
Tom	10	80%	72%

Table 2: Recall and Precision for each user

Test 2: Accelerator and Orientation Sensor, DTW Algorithm

The next test was done on the signature recognition component implementing the DTW algorithm. The goal of the test was to determine suitable thresholds that could distinguish and classify users, based on the distance values output by DTW and the corresponding averages and standard deviations of the data sets. Each signature was compared pairwise against the 75 other signatures, and the distance value of each channel was recorded. For each user, the data was divided into two categories: “User vs. User”, where the two signatures were from the same user, and “User vs. Others”, where the two signatures were from different users.

Figures 2a, 3a, 4a and 5a show the average distance values for the two categories for Anna, Isaac, Chirag and Diego respectively. Figures 2b, 3b, 4b, and 5b show the standard deviation for the two categories for Anna, Isaac, Chirag, and Diego respectively.

[5] In Machine Language terminology, Recall: True Positives / (True Positives + False Negatives); Precision: True Positives / (True Positives + False Positives)

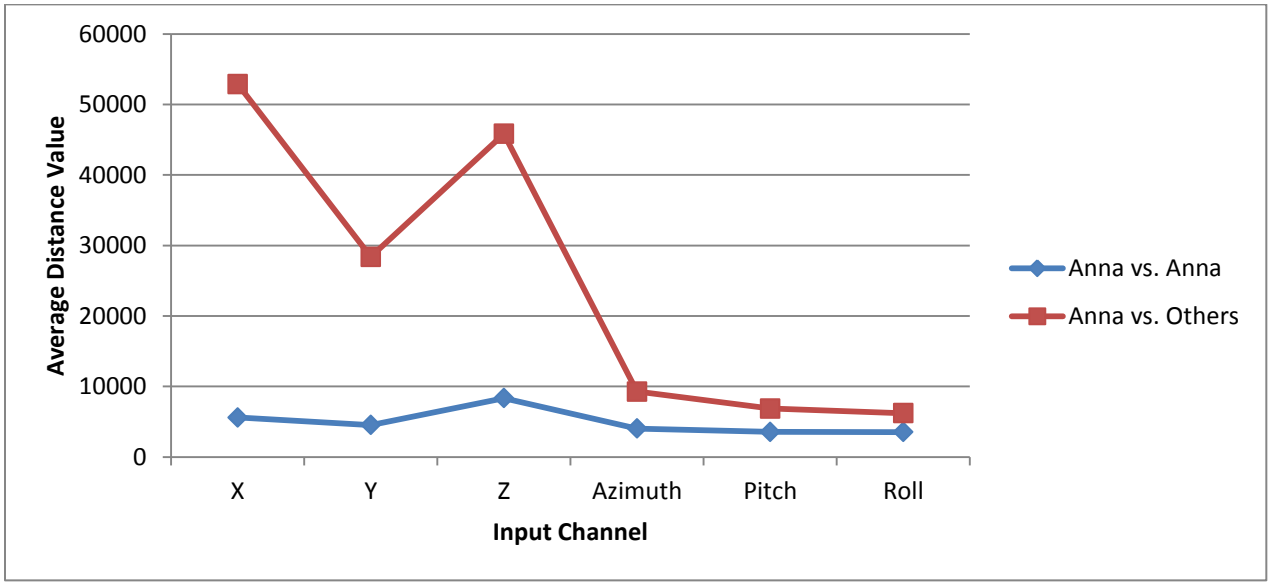

Figure 2a: Average distance values for Anna's data for each of the 6 channels

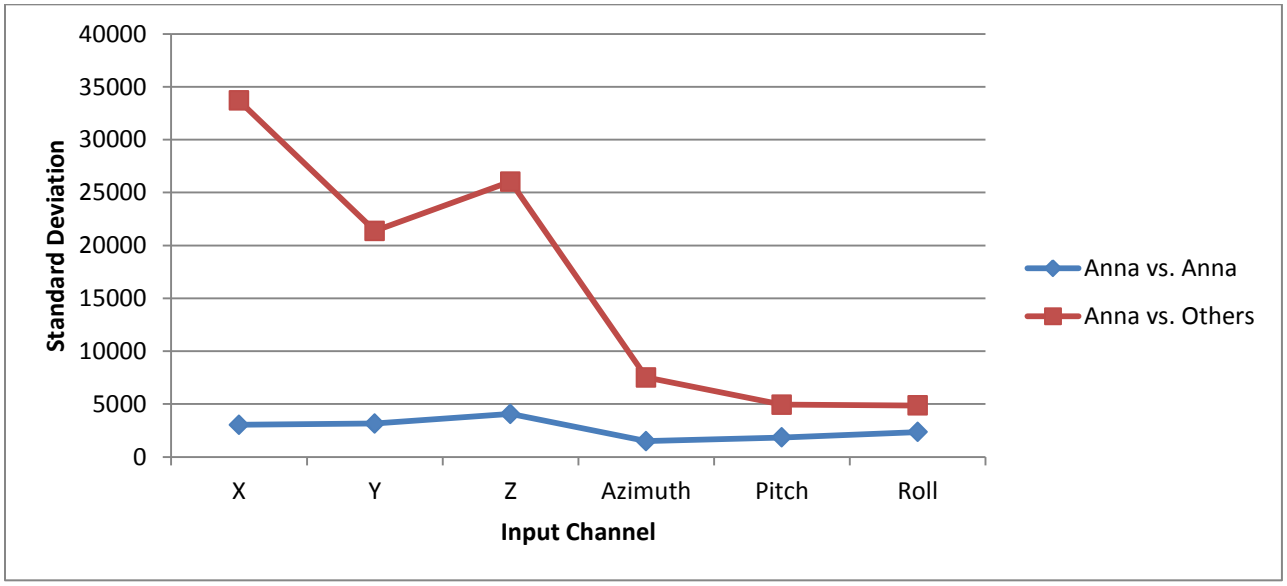

Figure 2b: Standard Deviation for Anna's data for each of the 6 channels

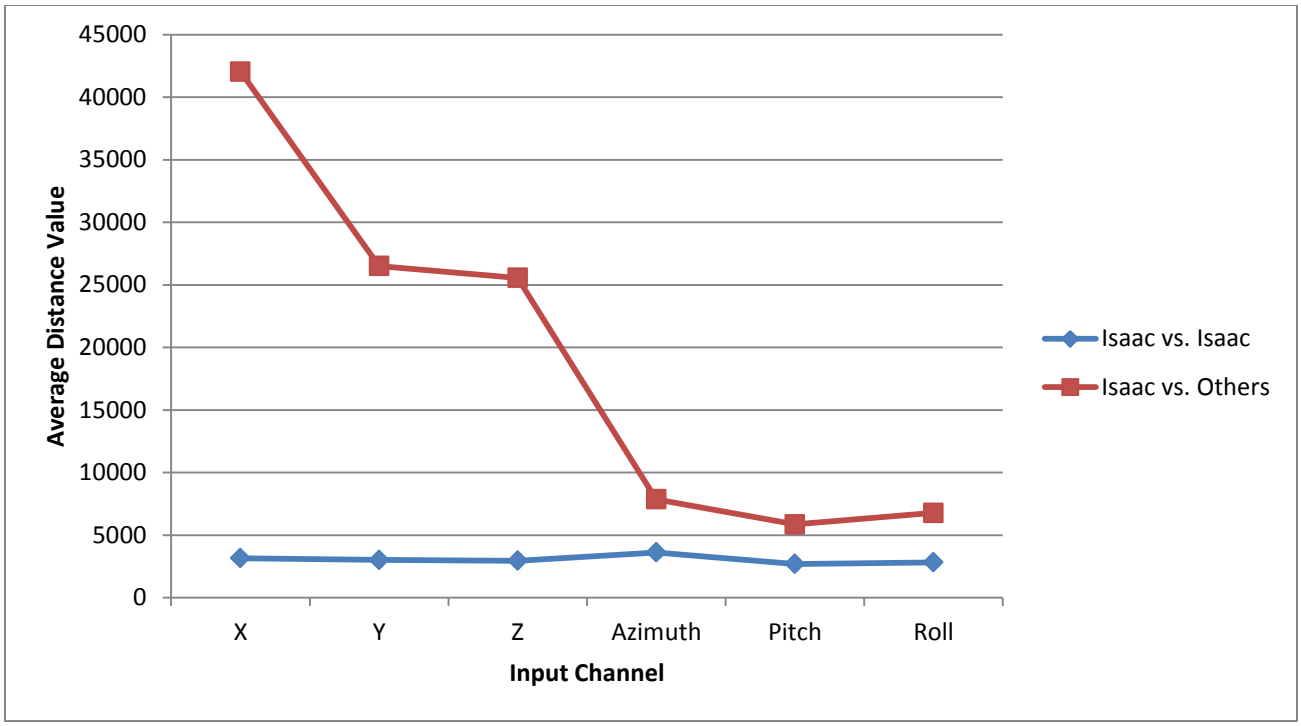

Figure 3a: Average distance values for Isaac's data for each of the 6 channels

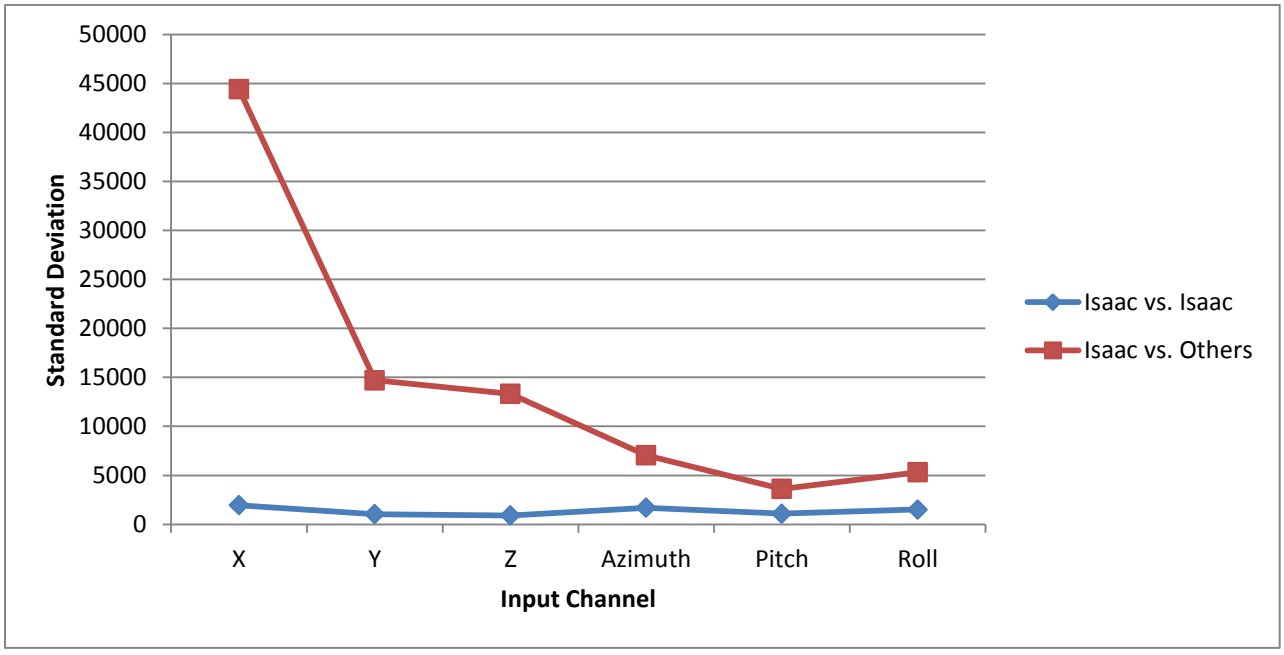

Figure 3b: Standard deviation for Isaac's data for each of the 6 channels

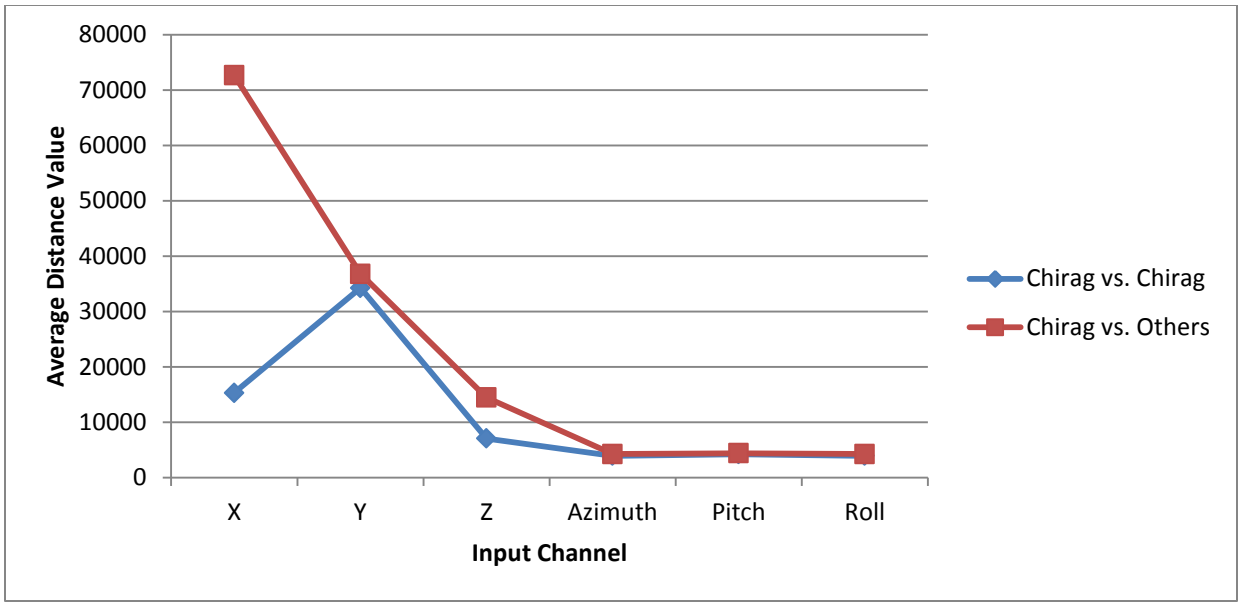

Figure 4a: Average distance values for Chirag's data for each of the 6 channels

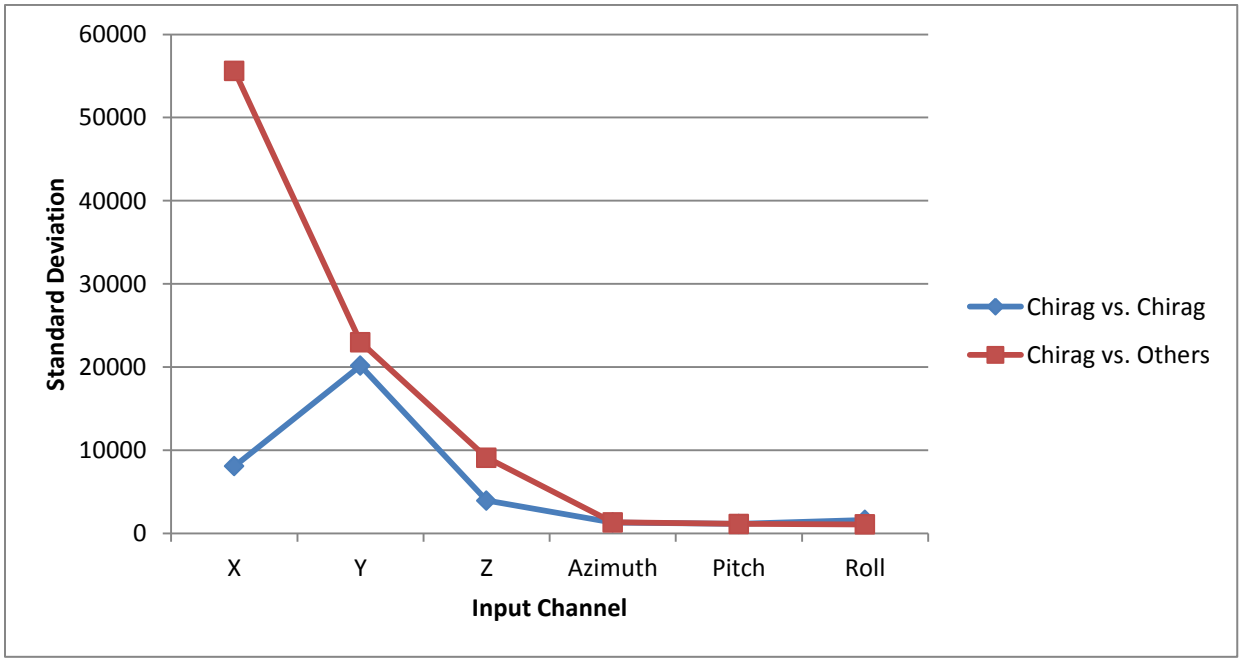

Figure 4b: Standard Deviation for Chirag's data for each of the 6 channels

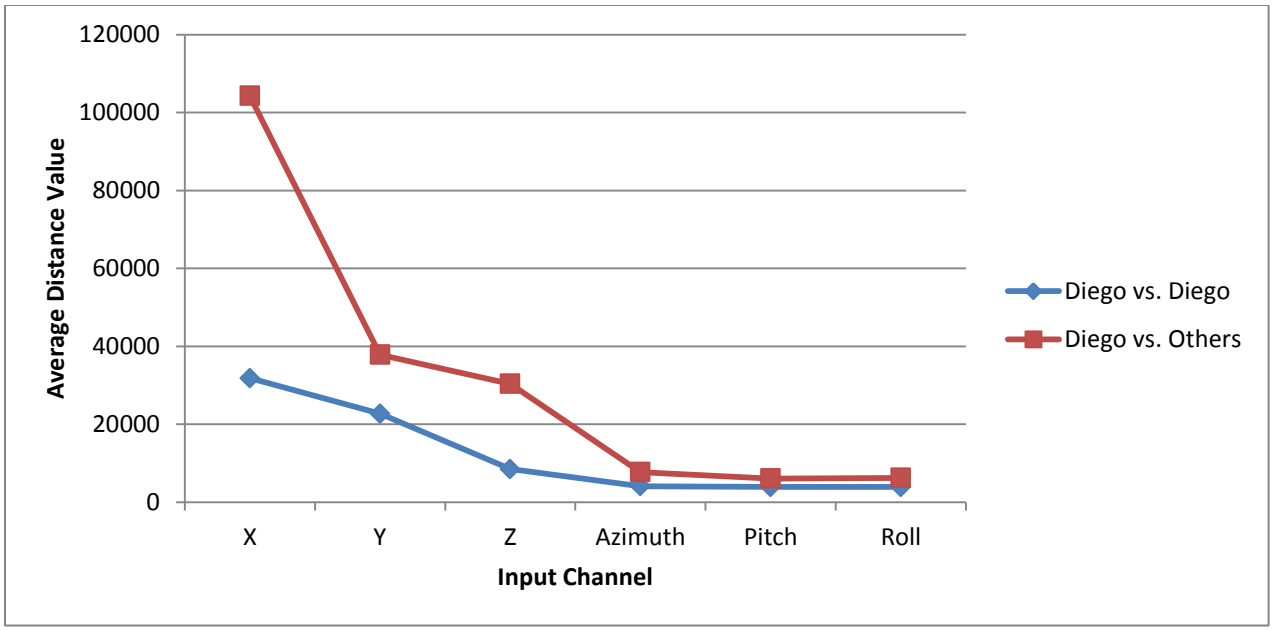

Figure 5a: Average distance values for Diego's data for each of the 6 channels

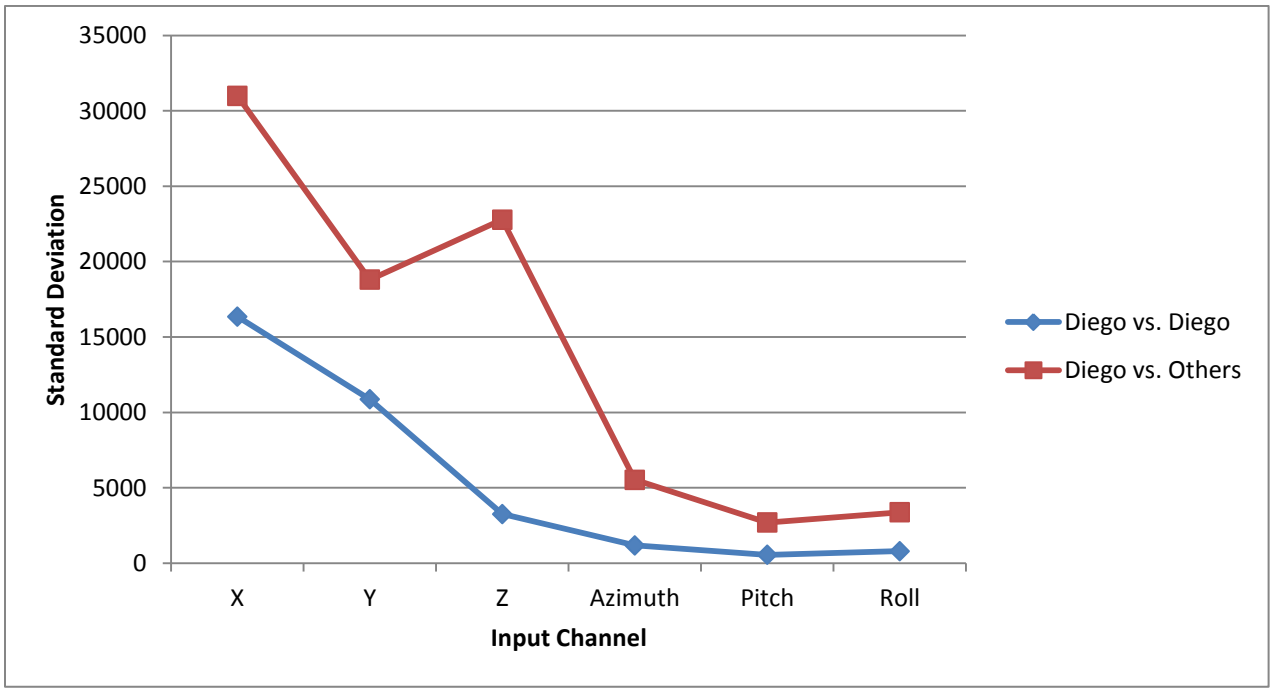

Figure 5b: Standard deviation values for Diego's data for each of the 6 channels

Using the average distance for each channel in the "User vs. User" category, we determined a threshold based on half a standard deviation from the mean of each channel for each user. That is, if a distance value fell within half a standard deviation above or below the mean distance value, we would consider the two signatures to be a match. We reclassified the data according to this threshold for each channel, and determined true and false positives. We then experimented with increasing the threshold, using multiples of half a standard deviation for each threshold, and calculated the true and false positive rates again for each new threshold. With this data, we generated Receiver Operating Characteristic (ROC) curves, which are plots of the true positive rates against false positive rates at the various thresholds[6]. Figures 6a, 6b, 6c, 6d, 6e, 6f, and 6g show the ROC curves.

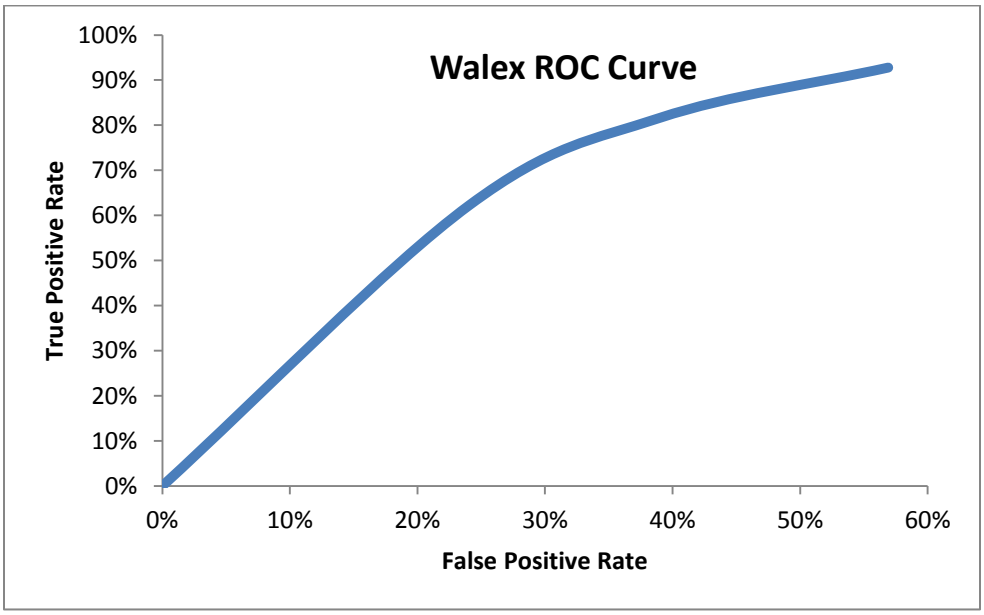

Figure 6a: Walex's ROC Curve

[6] Note that Recall is the equivalent of True Positive Rate, while Precision is the equivalent of True Positive Rate / (True Positive Rate + False Positive Rate)

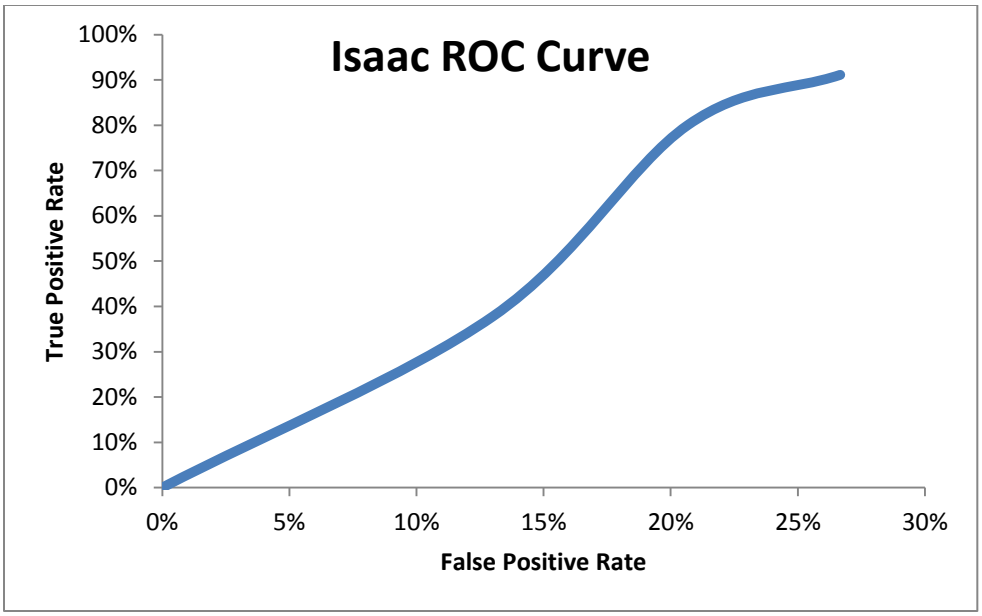

Figure 6b: Isaac's ROC Curve

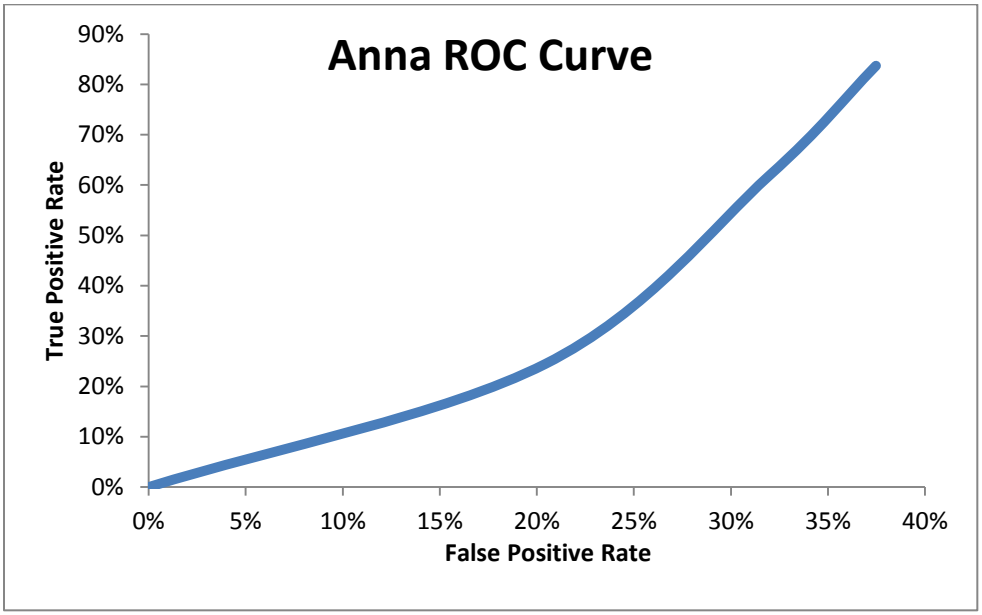

Figure 6c: Anna's ROC Curve

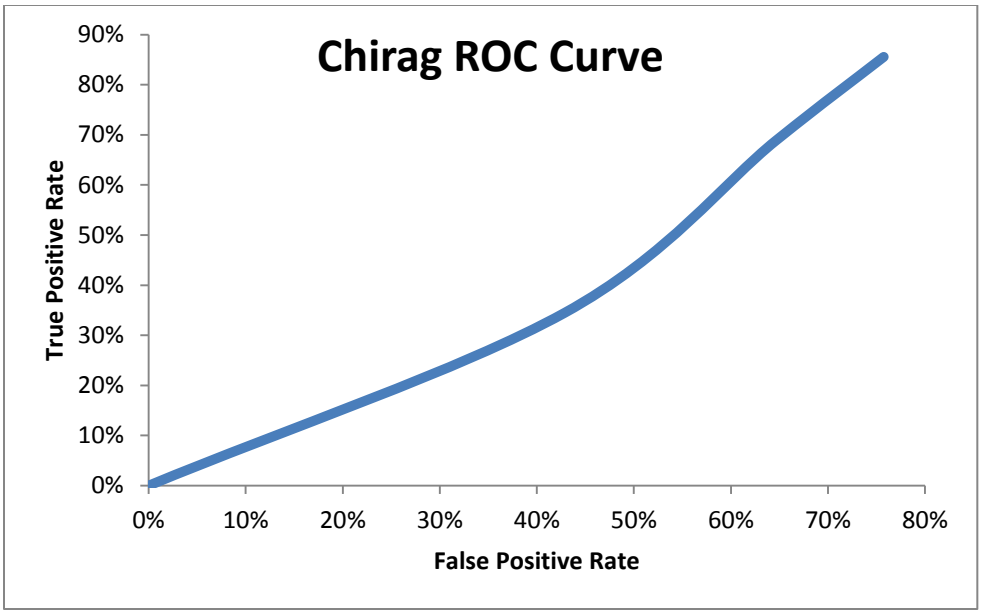

Figure 6d: Chirag's ROC Curve

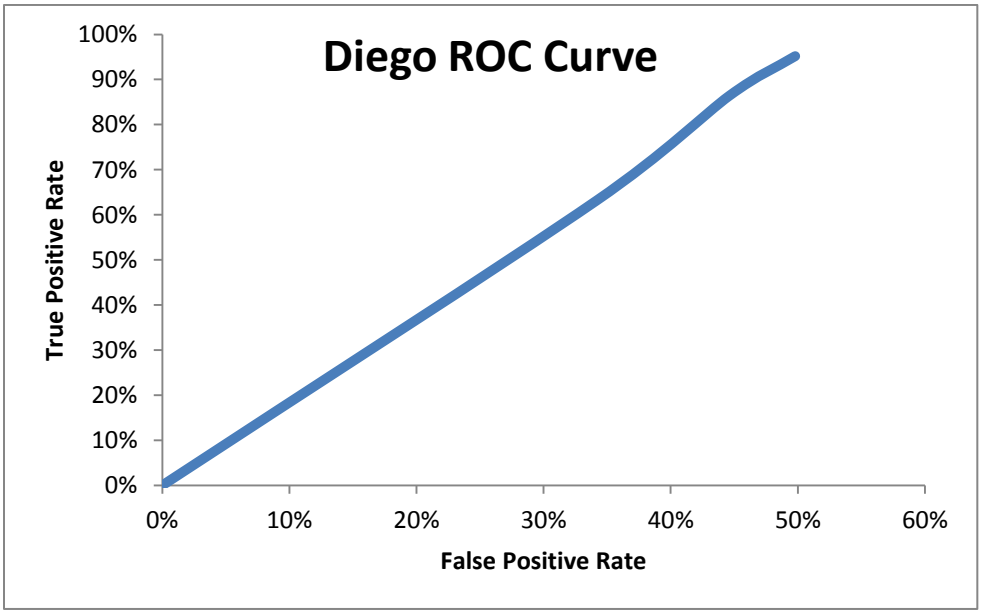

Figure 6e: Diego's ROC curve

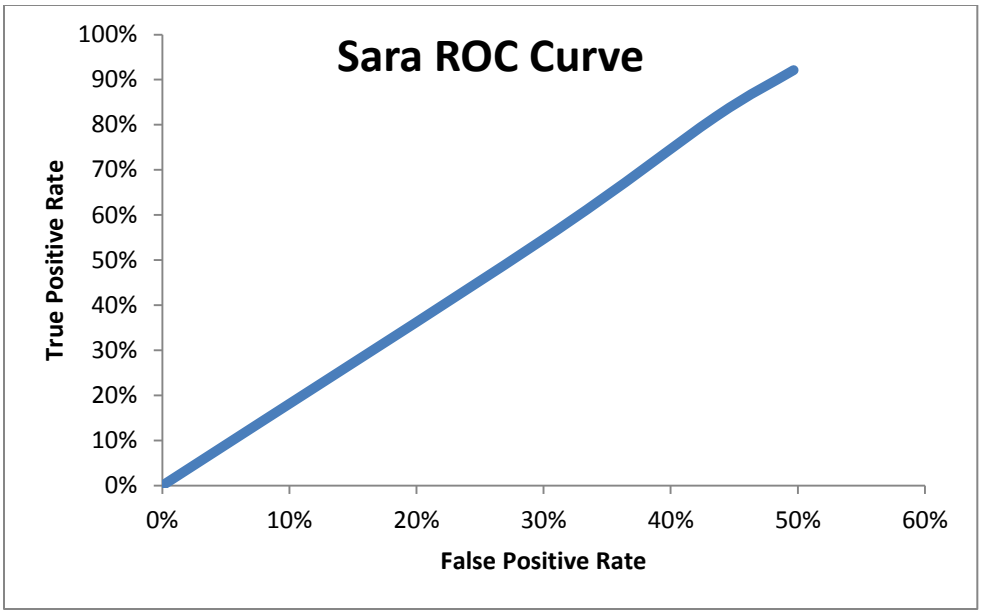

Figure 6f: Sara's ROC Curve

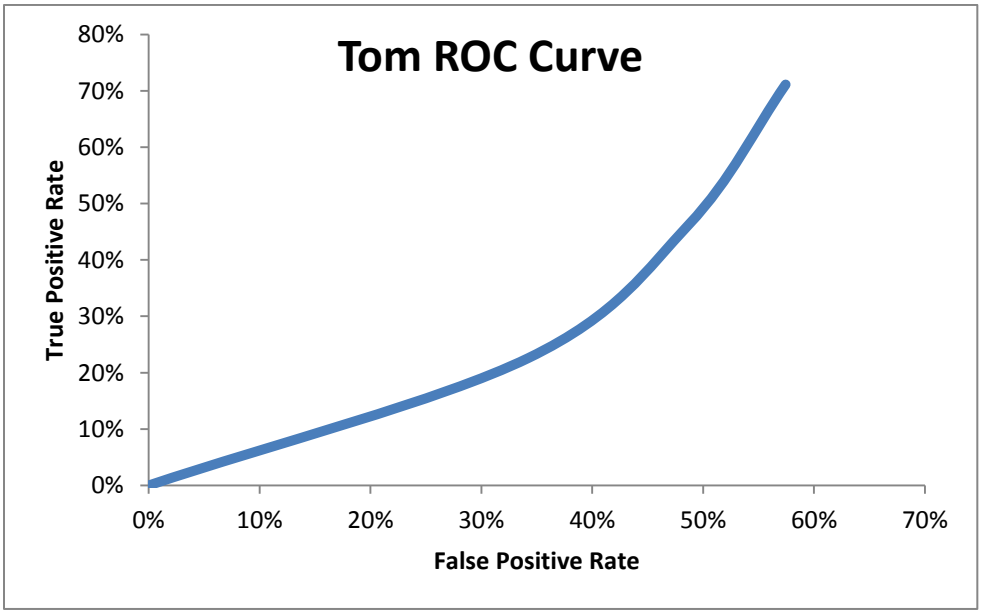

Figure 6g: Tom's ROC Curve

Test 3: WIFI Sensor, Signal Strength Threshold

The following tables show the statistics of the WIFI access point test done at 10 locations. At each location, 10 WIFI scans were done at various times in the day. As long as an access point (AP) was seen at least 8 times out of the 10 scans, it was considered to be “consistently seen”. Table 3 shows the maximum number of access points seen for a single scan at that location, as well as the number of access points that were “consistently seen” across all 10 scans at that location.

The accuracy for a single scan was calculated as the ratio of “consistently seen” AP to the total number of AP seen in that scan. Table 3 also shows the average number of APs seen per scan, as well as the average accuracy over 10 scans for each location.

Location Name	Max. AP in one scan	No. of “consistently seen” AP	Average no. of APs seen per scan	Average accuracy per scan
Apartment 1	30	15	24.2	63%
Apartment 2	16	10	14.0	73%
Apartment 3	23	29	19.4	83%
Apartment 4	29	19	23.9	81%
Apartment 5	15	10	12.9	78%
Campus 1	24	8	20	71%
Campus 2	20	13	18.1	72%
Campus 3	36	27	32.9	83%
Café 1	16	12	14.1	86%
Café 2	16	7	11.7	64%

Table 3: WIFI Scan statistics for test without threshold

The team then examined the average signal strength of the consistently seen and non-consistently seen APs at each location, and the results are show in Table 4.

Location Name	Average strength of AP of non-“consistently seen” AP (dB)	Average strength of “consistently seen” AP (dB)
Apartment 1	-82.1	-73.1
Apartment 2	-87.0	-74.0
Apartment 3	-85.1	-74.4
Apartment 4	-82.6	-66.3
Apartment 5	-84.2	-69
Campus 1	-86.1	-79
Campus 2	-84.4	-76
Campus 3	-83.3	-75.5
Café 1	-82.7	-76.5
Café 2	-82.4	-57.6

Table 4: WIFI Scan average signal strength,

As can be seen from Table 4, the average signal strengths are less than -82 dB for non-“consistently seen” APs, and are distinctly separated from the average signal strengths of “consistently seen” APs. As such, a threshold of -82 dB was chosen, and access points with signal strengths lower than -82 dB were removed from the data. The statistics from Table 3 were recalculated with the new threshold, and are shown in Table 5.

Location Name	Max. AP in one scan	No. of "consistently seen" AP	Average no. of APs seen per scan	Average accuracy per scan
Apartment 1	23	11	17.1	66%
Apartment 2	9	7	8.3	86%
Apartment 3	13	10	11.8	85%
Apartment 4	23	17	19.2	90%
Apartment 5	12	8	10.2	80%
Campus 1	13	8	11.5	70%
Campus 2	16	12	13.7	90%
Campus 3	26	21	24.5	86%
Café 1	12	10	10.2	99%
Café 2	11	6	7.9	82%

Table 5: WIFI Scan statistics for test with threshold of -82 dB

Figure 7 shows the average accuracy per scan with and without the threshold of -82 dB for each location. There was an increase in average accuracy for most locations when the threshold is used; the average value of the average accuracy was 75.3% for the test without the threshold, and was 83.3% for the test with the threshold.

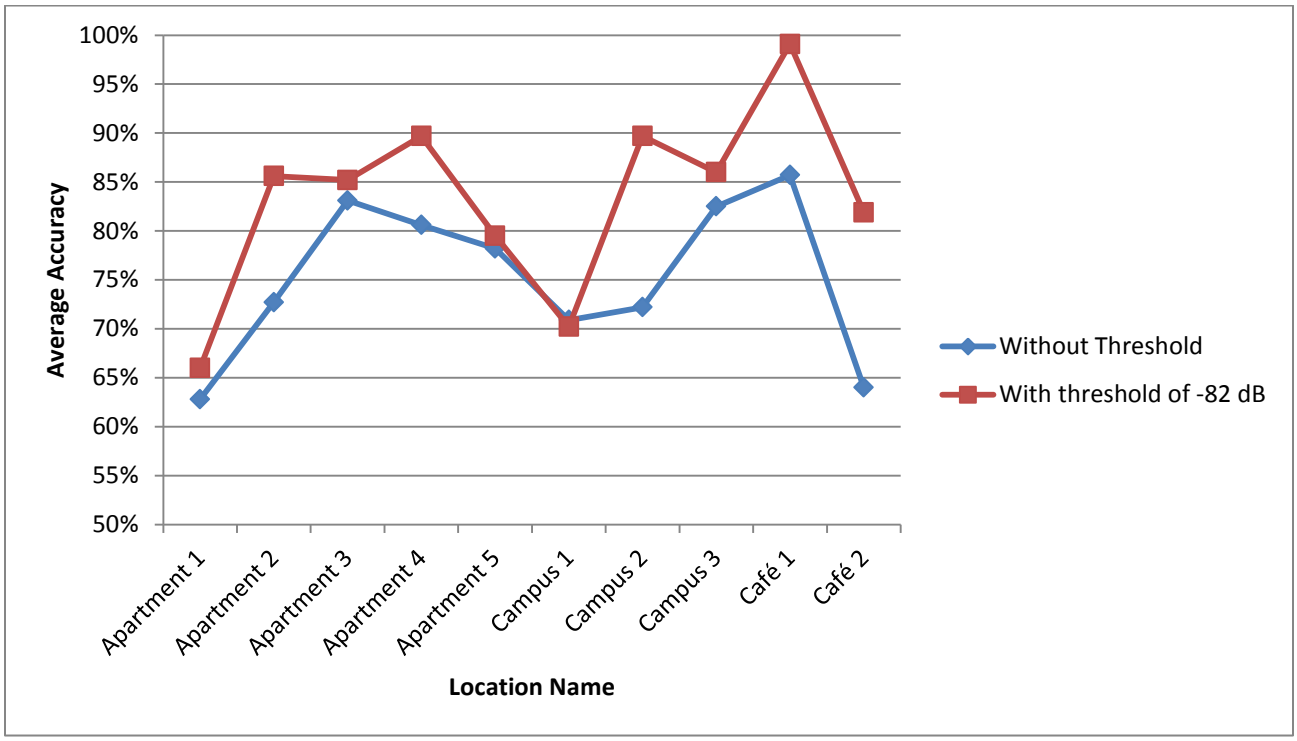

Figure 7: Average accuracy of WIFI Scan test with and without the threshold

Test 4: WIFI Sensor, Match Classification Threshold

With the signal strength threshold selected, the test was redone with the new algorithm at 8 new locations, and each location was scanned 10 times. Each data point was tested against all other data points and classified (as either the same location or different location) based on four match classification thresholds (60%, 70%, 80% and 90%), and the accuracy of classification for each location was calculated. Figure 8 shows the average recall for each location for each match classification threshold.

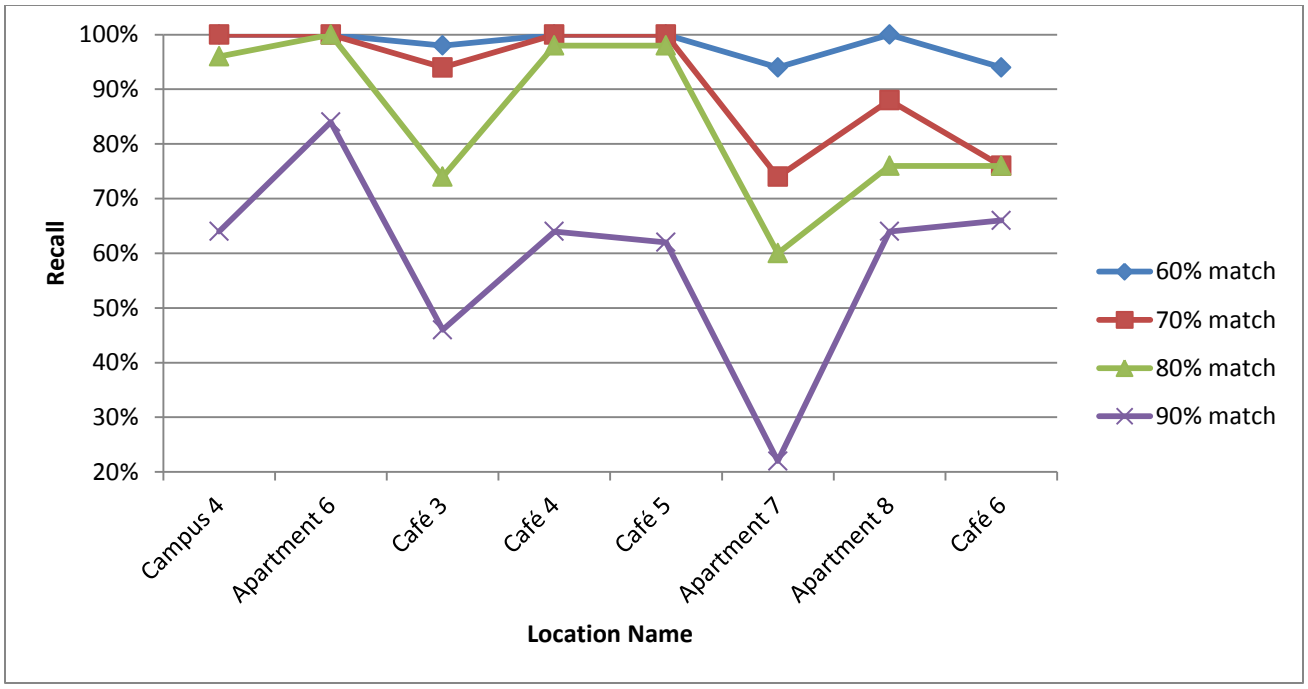

Figure 8: Average Recall of WIFI for varying match classification thresholds

It is noted that for all tests, there were no false positives generated by the algorithm, thus precision was consistently 100% across all locations. Hence, only recall is shown. The average recall across all locations was 98%, 92%, 85% and 59% for the 60%, 70%, 80% and 90% match classification thresholds respectively.

Discussion

Accelerometer and Orientation Sensor: Introduction

The tests done on the accelerometer and orientation sensor aim to provide a foundation for developing a practical implementation of gesture-based authentication as an authentication mechanism. Ideally, gesture-based authentication would be implemented compulsory step that would allow a user to login to his account, similar to the current function of the alphanumeric password.

Accelerator and Orientation Sensor, SVM Algorithm: Test 1

As mentioned in the Methods section, the SVM algorithm theoretically requires a large number of samples to be able to perform optimally. This characteristic is reflected Table 2, where the users with comparatively fewer true samples (Jess, Mandy, Wong, Tom) exhibit relatively lower precision and/or recall rates. Isaac and Chirag's test sets do have comparatively more true samples, and the algorithm's performance on their test sets is better.

An anomaly is Anna's test set. Despite having the most training examples, the accuracy on her set, as seen from Table 1, is surprisingly low compared to the other users. From Table 2, we see that the main contribution to this lowered accuracy is the 80% precision, meaning that there were many false positives. One possible reason for this is the simplicity of Anna's signature, which consisted of mostly vertical movement and minimal horizontal movement (due to very narrow 'a' and 'n' letters). This means that most of the activity happens in the Y channel, while the other channels are not as well used. As discussed in the Methods section, the truncation of data to 32 segments could have resulted in further information loss, making Anna's training signatures too simple and similar to other user's signatures, resulting in high false positives. Though our implementation attempted to minimize this lack of entropy by requiring a minimum of 3 seconds of input, signatures that do not fully utilize all 6 channels will unfortunately lack entropy as well. A possible area of future work would be to mitigate this problem better; perhaps by pre-emptively identifying overly simplistic signatures input by users and requiring a different signature (similar to how certain web services require the user to have a password with a minimum number of lowercase and uppercase letters).

Accelerator and Orientation Sensor, DTW Algorithm: Test 2

Examining Figures 2a, 3a, 4a, and 5a, we see that in general, all of the means of the "User vs. Others" category are higher than the "User vs. User" category, which suggests that on average, two signatures from the same user do have a lower distance value than two signatures from different users. Previously, we had hoped that each of these two categories would represent a distinct mode of a bimodal distribution that would be easily separated by a threshold. Unfortunately, when we consider the standard deviations from Figures 2b, 3b, 4b, and 5b, we see that the range of values around those means are relatively large, and the overlap between the distributions of the two categories is quite significant. Also, some channels, such as Chirag's Y channel from Figure 4a, have very high "User vs. User" means that are very close to the "User vs. Others" mean, thus choosing any threshold to classify a signature is already guaranteed to have many false positives and false negatives. Note that having a larger threshold (more standard deviations) allows more signatures be classified correctly (less false negatives), but also introduces more false positives; having a smaller threshold results in the opposite (more false negatives and false positives), so there is a tradeoff. Starting with many false positives and false negatives makes the classifier doomed to perform poorly, regardless of how the threshold is tweaked.

Indeed, we see from Chirag's ROC curve in Figure 6d that the false positive rate almost matches the true positive rate, resulting in a poor classifier regardless of threshold. A possible explanation for this is that Chirag's signatures are unfortunately just not consistent enough for the algorithm to work, especially in the vertical direction (Y channel). Looking at other ROC curves for the other users in the Figure 6 series, we see that performance is slightly better, but to achieve close to 100% true positive rate requires about 30-50% false positive rate; that means that at 100% recall, precision drops to about 60-75%.

This raises a potential issue: user's signatures may be inconsistent, despite them being from the same user, and even slight variations in the signature may result in a classification error. This is the downside of the convenience provided by the DTW algorithm, versus that of the SVM algorithm. Since SVM requires many signatures to train on, it is more robust to variations in user input. Thus, while SVM is less convenient to train than DTW, the potential for DTW to be more inaccurate is a significant problem that must be mitigated when developing a practical implementation of such a gesture-based authentication app.

Accelerometer and Orientation Sensor: Other Issues

The most pertinent observation from both gesture-based authentication tests is that none of the user's test sets had perfect recall and precision. That means that at least once for every user, either the algorithm wrongly rejected a true signature (imperfect recall), or the algorithm wrongly accepted someone else's signature as the test user (imperfect precision). The former error presents an annoyance to the user, as their legitimate authentication attempt was rejected, and they would need to redo the authentication again to login. The latter error is arguably more problematic; an unauthorized attacker could successfully authenticate as the user, which could result in a lot more damage and loss for the user. Compared to password authentication, which has perfect recall and precision, the performance of gesture-based authentication is still not as reliable. Thus, having gesture-based authentication as the lone, compulsory authentication method (similar to the password's current role) may not be practical for web services, which do have a lot to lose with imperfect precision.

Still, that is not to say that the two algorithms for gesture-based authentication are completely useless, since both algorithms did perform relatively well for a few users in our tests; more work needs to be done on improving the algorithms to ensure consistently good performance across the board for every user. A suggestion made by our mentors at Yahoo was to combine all 6 channels of data into 1 by calculating the aggregate energy of each of the 6 channels of continuous time data. We leave this as a possible signal processing modification to the algorithm for future work.

WIFI Sensor: Introduction

The tests done on the WIFI sensor aim to provide a foundation towards developing a practical implementation of WIFI access point signatures as an additional authentication factor. Theoretically, the idea seems sound. A user's location is a strong biometric that is linked heavily to a user's behavior, and can provide vital information to be used for authentication. The idea of authentication based on an access point signature provides additional complexity and entropy as compared to the traditional alphanumeric password, since the signature comprises of the SSID and BSSIDs of all the access points on the list; this additional complexity makes it harder for an attacker to guess the "password" (i.e. the signature) through brute force. Similarly, as strong, complex passwords are often hard to remember, and usually result in users having to write them down [17], this WIFI sensor-based requires the user to only remember where the location is, thereby minimizing the risk of having an attacker see the "password".

Still, there are multiple issues in developing a practical implementation and Tests 3 and 4 both attempt to address some of these issues, while continuing to raise others.

WIFI Sensor, Signal Strength Threshold: Test 3

Table 3 illustrates the unreliability and inconsistency of WIFI sensor signatures. The "maximum access points seen for a single scan" statistic is shown to demonstrate how many non-"consistently seen" access points can potentially be picked up in one unlucky instance. This may present a problem for users who attempt to 'register' their WIFI access point signature for the first time; that is, if the model signature is stored with so many unreliable access points in that one unlucky instance, future attempts to authenticate could result in scans that do not include these unreliable access points, resulting in poorly matched lists. Table 4 attempts to address this by showing a bimodal distribution of signal strengths between "consistently seen" and non-"consistently seen" access points, hinting at a correlation between signal strength and reliability of access point. This was the basis of introducing a signal strength threshold to ignore the (hopefully) unreliable access points, and implementation of the threshold resulting in an improved performance in Table 5.

It is important to note from Table 5 that while the threshold decreased the maximum and average number of access points seen (likely due to the omission of non-"consistently seen" access points), it also resulted in a decrease in the number of "consistently seen" access points. This has two implications. Firstly, this means that some "consistently seen" access points exhibited signal strengths that were below the threshold – this is not surprising, since Table 4 only shows the average signal strength (which also implies that some non-"consistently seen" access points are greater than the threshold). However, this has a huge effect on the second implication, which is that some previously considered "consistently seen" access points are now unfortunately deemed to be non-"consistently seen", thereby contributing to the error rate. Thus, while the introduction of a threshold helps to lower the error rate by discarding unreliable access points, the threshold also simultaneously increases the error rate by redefining access points as unreliable.

This simultaneous increase and decrease of the error helps to explain the results in Figure 7. In the graph, it is clear that the improvement in accuracy seen by introducing a signal strength threshold was not consistently applied across all locations. While the accuracy in Café 1 increased to almost 100%, some locations barely saw an increase in accuracy; in the case of Campus 1, accuracy actually decreased slightly. This variation can be attributed to the variation in the distribution of signal strengths of

"consistently seen" access points. For Campus 1, there were a few "consistently-seen" access points that did indeed fall below the threshold during one or two of the scans, resulting in all instances of that access point contributing to the non-"consistently seen" error rate. This demonstrates that there is indeed a wide variation in the distribution of signal strengths, and obtaining a solid threshold that improves the performance of all locations is a difficult problem that could be an area for future work. Despite this, introducing a threshold generally did show an increase in average performance from 75% to 83%, so we conclude that including a signal strength threshold in a WIFI sensor implementation should be a step in the right direction.

WIFI Sensor, Match Classification Threshold: Test 4

Figure 8 deals with the other threshold in consideration: the match classification threshold. Since the signal strength threshold does not cleanly eliminate unreliable access points, there are thus still non-"consistently seen" access points that need to be dealt with in the signatures. This is an important parameter that will greatly determine the success of a practical implementation of the WIFI sensor as an authentication factor. If the threshold is too high, access point signatures are likely to be classified as a mismatch (false negative), resulting in legitimate users being unable to authenticate correctly. If the threshold is too low, legitimate users will be able to login easily, but this also means that an attacker can more easily break into user accounts as well, as the attacker only needs to identify a small number of access points on a user's signature to be able to login successfully.

As we can see from Figure 8, decreasing the threshold from 90% to 60% helps to increase the recall for all locations, so a lower threshold will result in more successful authentications by legitimate users; an average 98% recall for a 60% threshold means that out of a 100 authentications by a legitimate user, an average of 2 of those attempts will be wrongly rejected. Interestingly, lowering the threshold does not increase the number of false positives – unlike the gesture-based authentication algorithm discussed in previous sections, where two different input signatures from two different people could be confused for the same one, the WIFI sensor algorithm will not confuse two different access point lists from two different locations for the same one. However, this does not mean we should definitely use a 60% threshold; an attacker will still benefit from a lower threshold, and we discuss this issue of possible attacks in the next section.

What then is a good threshold to use? Unfortunately, we do not have a solid answer. Our results can only show that a 90% threshold is probably not a good threshold to use due to relative low recall rates,

but the other threshold values may all be appealing to a practical implementation, depending on the implementation's tolerance for potentially low recall rates. We leave this as a problem for future work, although we have provided the method to generate these values to approach this problem.

From Figure 8, we also see that while a 90% threshold performs relatively poorly across the board, Apartment 7 performs exceptionally poorly. A look at the raw data for Apartment 7 provides clues for the explanation; Apartment 7 had five "consistently seen" access points, but more than 16 non-"consistently seen" access points that occurred quite frequently (but unfortunately not frequently enough to be deemed "consistently seen"). However, this raises a potential issue for a practical implementation of the WIFI sensor: there may be unfortunate locations like Apartment 7 with many unstable signals. Given the small number of test locations, it's not possible to infer from the data just how often these locations exists, and whether the existence of such locations would have a significant impact on the authentication method if implemented. Future work could include identifying the frequency of such unfortunate locations and their corresponding impact on the authentication method.

WIFI Sensor: Other Issues

The biggest issue with this WIFI sensor authentication method is its reliance on fixed WIFI networks that must be established as a model signature beforehand. This means that if a user is logging on to the Internet from a location with no or minimal WIFI access points (e.g. through 3G), this method will not work at all. Also, users cannot login from a completely new location as well, which may present a huge convenience problem for users who move around to new places often. These two factors seem to indicate that this WIFI authentication cannot be a compulsory authentication step, but rather an optional (or perhaps 'opt-in') authentication step.

This does not mean WIFI sensor authentication is completely useless. A possible opt-in implementation is an option for a user to declare a "secure location" for his account. If logging in from his "secure location", the user need only authenticate his location with the WIFI sensor, and skip the regular authentication steps. Other possible options for this implementation can include giving additional privileges to a login from a "secure location", such as permission to perform more sensitive user account operations like password resets. We leave the exploration of the feasibility of such options as future work.

As mentioned in the previous section, the attacks on this WIFI sensor authentication method are rather peculiar. In some ways, this method is less secure than the password if an attacker knows the victim's

location, and has physical access to the same location; the attacker can thus easily beat this authentication method. However, an attacker who knows and has access to the location is likely someone who knows the victim, which potentially decreases the number of possible attackers. Also, without actual physical access to the location, an attacker must employ more advanced and technical methods to replicate the access point signature (such as actual network-based attacks like packet sniffing or Man-In-The-Middle [19], or actually constructing a replica of WIFI access points to mimic the user's location). Compared to simply entering a stolen password (perhaps obtained through phishing) on his computer, this authentication method increases the difficulty of executing an attack.

Conclusion

For our project, we implemented a gesture-based authentication method and a location-based authentication in an Android app. For the gesture-based authentication, both the SVM algorithm and DTW algorithm have their own strengths and weaknesses. The SVM algorithm is more robust to minor variations in a user's signature compared to the DTW algorithm, and generally performs better when properly trained. However, it is less convenient to train as compared to the DTW algorithm, and also performs poorly on simplistic signatures. For the location-based authentication, the inconsistency and instability of WIFI access points poses a significant issue for the performance of the authentication method. We implemented two thresholds to discard weaker access points and provide tolerance for inconsistent WIFI access points, and these two thresholds result in varying degrees of improvement depending on the level of the threshold chosen.

Acknowledgements

I would like to thank Ty Lee, the Capstone project's industry advisor, for providing guidance and feedback throughout the course of the project, and helping to develop the vision of the Capstone project. I would also like to thank his colleagues, Francis Hsu and Saurabh Tewari, for also providing guidance and feedback, especially during the initial stages of the project.

I would like to thank Don Wroblewski, the Capstone project's faculty advisor, for providing guidance throughout the course of the project, and for reading and providing feedback on this thesis.

I would like to thank Laurent El Ghaoui, my concentration advisor, for reading and providing feedback on this thesis.

I would like to thank Te-Yuan Huang, one of the creators of MoviSign, for allowing us to use the MoviSign code as a foundation for our project.

References

[1] R. Condon. (2011, Nov 1). *Alternatives to passwords: Replacing the ubiquitous authenticator* [Online]. Available: http://searchsecurity.techtarget.com/magazineContent/Alternatives-to-passwords-Replacing-the-ubiquitous-authenticator, last access Mar. 14, 2014

[2] Jain, A., Hong, L., & Pankanti, S. (2000). "Biometric Identification". Communications of the ACM, 43(2), p. 91-98. DOI 10.1145/328236.328110

[3] *Building a Wireless Sensor Network using Smartphones* [Online]. http://www.mistralsolutions.com/hs-downloads/tech-briefs/dec11-article1.html , last access Mar. 14, 2014.

[4] B. Gaille. (2014, Jan 31) *Current Smartphone Manufacturer Market Share Statistics. BrandonGaille.com* [Online]. Available: http://brandongaille.com/current-smartphone-manufacturer-market-share-statistics/, last access Mar. 14, 2014

[5] *iPhone 5s: Using Touch ID* [Online]. http://support.apple.com/kb/HT5883, last access Nov. 10, 2013.

[6] K. Korcz. (2011, Oct 19). *6 Cool Features Coming With Android 4.0 Ice Cream Sandwich* [Online]. Available: http://www.geeksugar.com/Ice-Cream-Sandwich-Features-20073482

[7] N. Handigol, T. Huang, and G. C. Liu. (2009, May). *MoViSign: A novel authentication mechanism using mobile virtual signatures* [Online]. Available: http://cs229.stanford.edu/proj2009/HandigolHuangLiu.pdf

[8] Liu, Jiayang, et al. (2009). "uWave: Accelerometer-based personalized gesture recognition and its applications." *Pervasive and Mobile Computing* 5.6 (2009): 657-675.

[9] R. Leiteritz, (2010, April 27). *Copy of Google's Submission Today To Several National Data Protection Authorities On Vehicle-based Collection of Wifi Data For Use In Google Location Based Services* [Online]. Available: http://static.googleusercontent.com/media/www.google.com/en/us/googleblogs/pdfs/google_submission_dpas_wifi_collection.pdf , last access Mar. 14, 2014.

[10] *SensorEvent|Android Developers* [Online]. http://developer.android.com/reference/android/hardware/SensorEvent.html#values, last access Mar. 14, 2014.

[11] *SensorManager|Android Developers* [Online]. http://developer.android.com/reference/android/hardware/SensorManager.html#getOrientation(float[], float[]), last access Mar. 14, 2014.

[12] *Accelerometer Sensor Data Processing* [Online]. Available: http://seattlesensor.wordpress.com/2013/01/01/accelerometer-sensor-data-processing/, last access Mar. 14, 2014.

[13] *Dynamic Time Warping* [Online]. http://luscinia.sourceforge.net/page26/page16/page16.html , last access Mar. 14, 2014.

[14] *Dynamic Time Warping (DTW) – File Exchange – MATLAB Central* [Online] http://www.mathworks.com/matlabcentral/fileexchange/43156-dynamic-time-warping-dtw, last access Mar. 14, 2014.

[15] *WifiManager|Android Developers* [Online] http://developer.android.com/reference/android/net/wifi/WifiManager.html last access Mar. 14, 2014.

[16] *Understanding the Network Terms SSID, BSSID and ESSID* [Online] http://www.juniper.net/techpubs/en_US/junos-space-apps12.3/network-director/topics/concept/wireless-ssid-bssid-essid.html last access Mar. 14, 2014.

[17] Mullins, Michael. (Jan, 2006). *Help users create complex passwords that are easy to remember* [Online] Available: http://www.techrepublic.com/article/help-users-create-complex-passwords-that-are-easy-to-remember/, last access Apr. 10, 2014.

[18] Danias, Vasilios. (2009). Approach. [Online] Available: http://homepages.inf.ed.ac.uk/group/sli_archive/slip0809_c/s0562005/theory.html , last access May. 1, 2014.

[19] Common Types of Network Attacks [Online]. Available: http://technet.microsoft.com/en-us/library/cc959354.aspx , last access May. 1, 2014.