MyLock

Final Report

Octavio Escalante (Apper)

Wen Bo Li (Programmer)

Sanket Pandit (Programmer)

April 10, 2014

Total Report Word Count (excluding title page, sample projects, and reference): **1912**

Total Apper Context Word Count: **490**

Note: We would like our final presentation video and report to be posted on the course website. We do not wish to release our source code on the course website.

1.0 INTRODUCTION

A recent eMarketer study has shown that non-voice mobile activity accounts for 20% of media time per day by US adults[1]. Mobile devices now carry very sensitive information such as private pictures, messages, and documents. Various surveys have also shown that majority of the users do not protect their phone with password[2][3]. As such, access to someone's physical device allows the attacker to view and steal all the private information stored on the device.

The progress in the field of pattern recognition using machine learning algorithms is constantly enhancing the security options allowed on mobile devices. Fingerprint or face recognition once available only through very complex and expensive systems is now available for mobile devices. In spite of the progress of both hardware and software support for intelligent security applications, mobile devices still rely heavily on fixed password-based authentication which can be inconvenient to use at home in case of a complex password, and not secure in case of a simple password.

This application attempts to provide users with an option to store sensitive information in the phone which will be guarded by intelligent security mechanisms on mobile devices that enables the right level of complexity depending on the context of the user. That is, depending on the user's location, four different authentication mechanisms are proposed: no password, password, gesture recognition, and signature recognition. The application also aims to analyze the feasibility and ease of use of gesture and signature based authentication options.

2.0 OVERALL DESIGN

The following is the block diagram of MyLock:

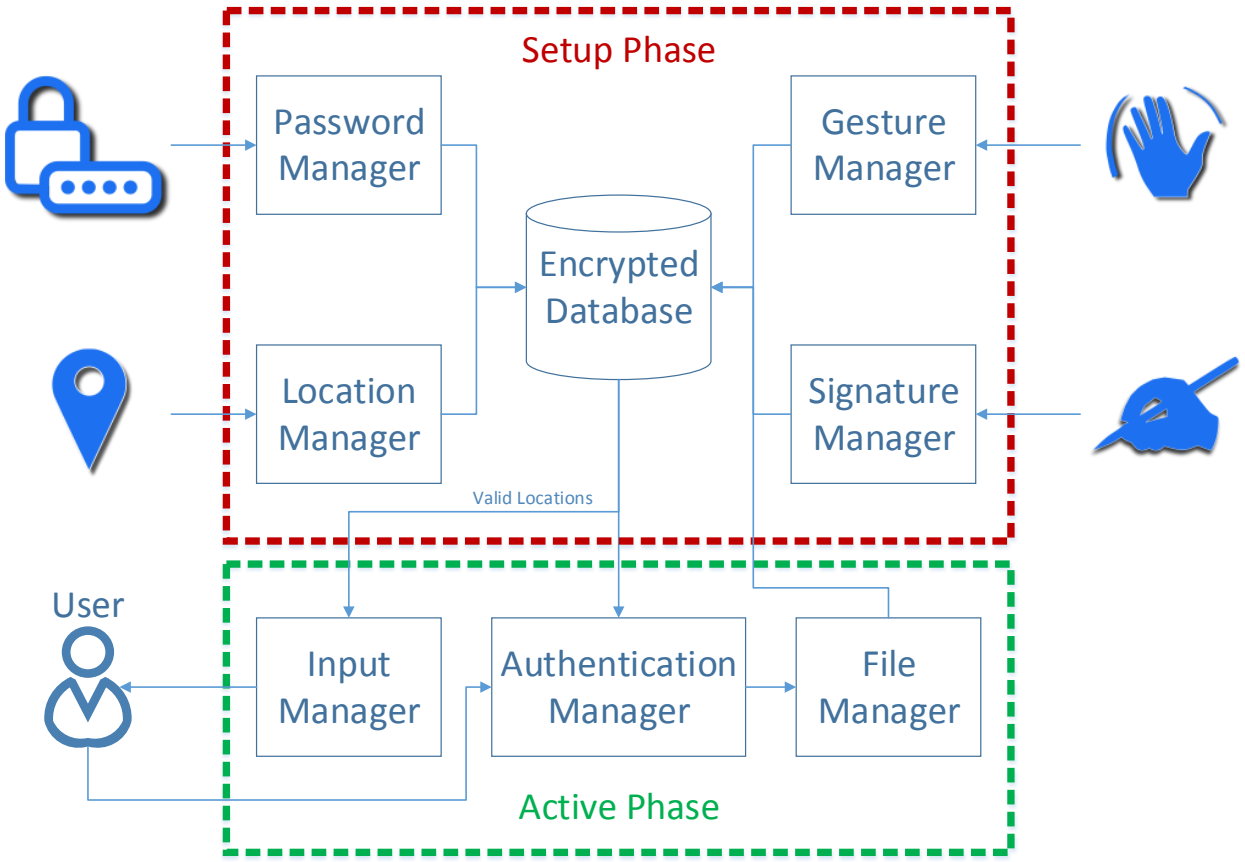


Figure 1: Block Diagram

The application usage can be divided into two phases

- **Setup Phase**: The user configures the application settings by entering his original gesture, signature, and defines safe zones.
- **Active Phase**: The locations based authentication option is activated. To view and save sensitive information securely, the user has to authenticate in the application through the authentication option at that location.

2.1 SETUP PHASE

The following blocks are used to configure the application in setup phase:

- **PASSWORD MANAGER**: The password manager stores the default password, which can be word(s) or numbers, into the database.
- **GESTURE MANAGER**: The gesture manager is responsible for requesting user input gesture (using the phone) 5 times. As the user performs the gesture, accelerometer data is recorded and passed on to a filtering algorithm for noise removal. Using the K-Means clustering algorithm described later, valid clusters are recorded and stored into the database.
- **SIGNATURE MANAGER**: Similar to gesture manager, signature manager is responsible for obtaining 7 samples of valid signatures from the user. They are then stored into the database as blobs.
- **LOCATION MANAGER**: Obtains the GPS and Wi-Fi information to determine the location and allow the user to determine the authentication type (No Password, Standard, Gesture, and Signature) for that zone. Once all the entries are verified, store the information into the database.
- **ENCRYPTED DATABASE**: Stores all password data during the setup phase and sensitive information entered in File Manager during the active phase. The database is encrypted with 256-bit AES encryption so in case of a forced intrusion to the database, the information cannot be decrypted without the key.

2.2 ACTIVE PHASE

The following blocks are used in the active phase:

- **INPUT MANAGER**: When the application starts, the input manager is activated and compares the GPS location and Wi-Fi information to the values stored in the database. It then asks the user for his/her authentication input.
- **AUTHENTICATION MANAGER**: The user input is compared with the values from the database. In the case of gesture authentication, the input value passes through filtering and is then divided into clusters. The newly created cluster is then compared with the values stored in the database to determine user authenticity.
- **FILE MANAGER**: Allows the user to create, update, and delete sensitive information. The files are stored in the encrypted database once the user saves the data.

3.0 AUTHENTICATION OVERVIEW

The following sections give a brief overview on the gesture and signature authentication algorithms used.

3.1 GESTURE AUTHENTICATION ALGORITHM

The motion gesture recognition manager is based on the process proposed by Schlömer et al. [x]. It consists of 4 main steps:

- **SIGNAL FILTERING** (Setup Phase): The noise recorded by the accelerometer is reduced by first removing data with values less than 1.2G and then omitting the vectors if none of their components is too different from the corresponding component of their predecessor.
- **CLUSTERING** (Setup Phase): The vector data is grouped in different clusters. This has the effect of finding significant points in the three-dimensional space through which the device passes during the motion gesture.
- **SEQUENCE PROBABILITY MODEL** (Setup Phase): In this step a probability model of the order in which the clusters found in the previous step are touched is produced using a Hidden Markov Model (HMM).
- **TESTING** (Active Phase)**:** In this step, the user provides his/her gesture for authentication. The same signal filters are applied on the input and new vectors are grouped to the closest clusters. Then probability of the order in which the clusters are found is computed using HMM. This probability is compared to a threshold. If the probability is greater than the threshold, then the input gesture passes and the user is granted access to the application.

3.2 SIGNATURE AUTHENTICATION

The signature recognition system is based on the research from Martinez et al. [x]. It consists of four steps.

- **IMAGE PRE-PROCESSING** (Setup Phase): All the images are cropped to the smallest fitting square. Then, all the images are scaled to the dimensions of the smallest cropped image.
- **FEATURE EXTRACTION** (Active Phase)**:** A black pixel count histogram is computed for the width and length dimensions.

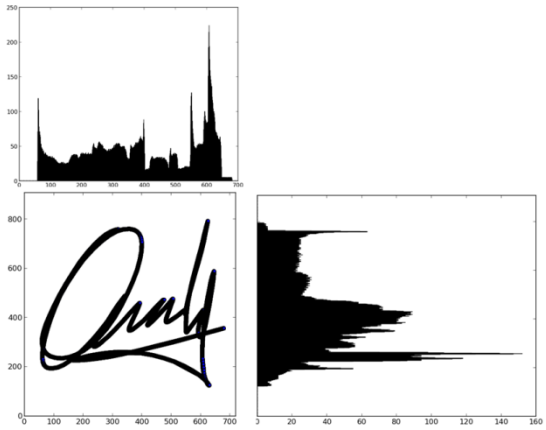


Figure 2: Feature extraction from signature

- **TRAINING** (Active Phase)**:** A Support Vector Machine algorithm is trained with a linear kernel. It attempts to maximize the margin *m* between the separated dotted lines that distinguishes the correct signature (circles) from forgeries (squares) using support vectors (points touching the dotted lines).

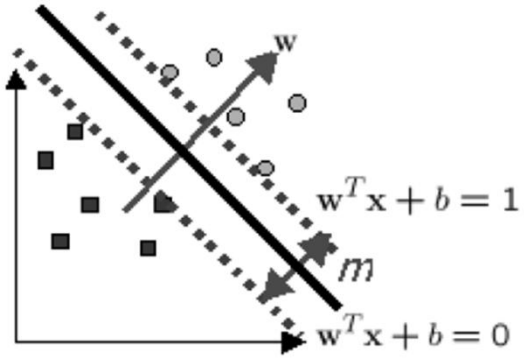


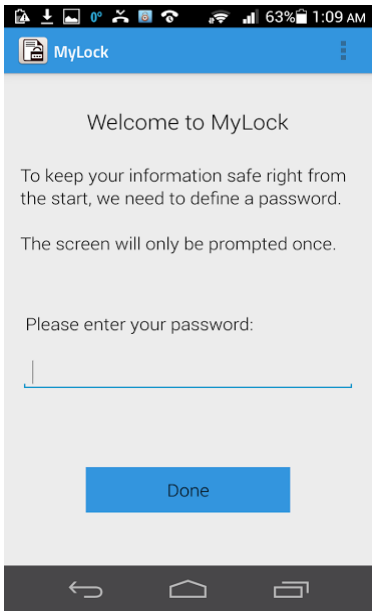
Figure 3: Support Vector Machine

- **TESTING** (Active Phase)**:** The distance between the new signature and the trained signature is computed. If the value falls within the acceptance region (between dotted lines), then the signature is accepted.

4.0 FUNCTIONALITY AND SCREENSHOTS

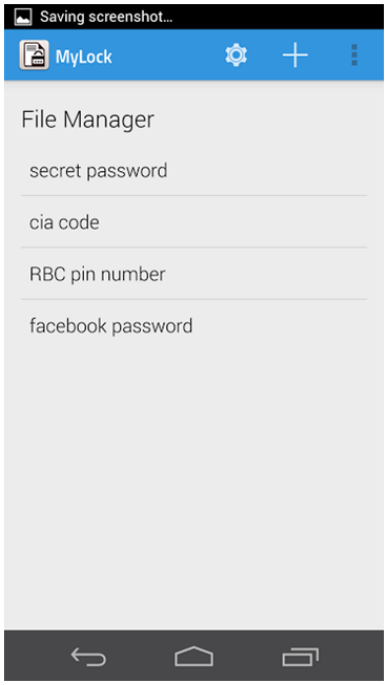
This section shows the functionality of the application such as file saving, gesture setup, signature setup and location setup.

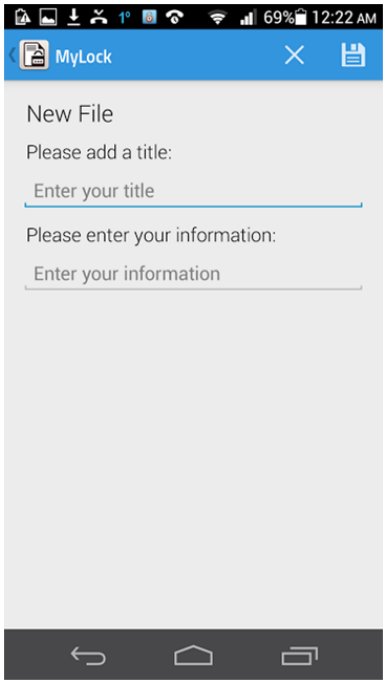
4.1 WELCOME SCREEN



The first screen requests the user to define a default password.

4.2 FILE MANAGER

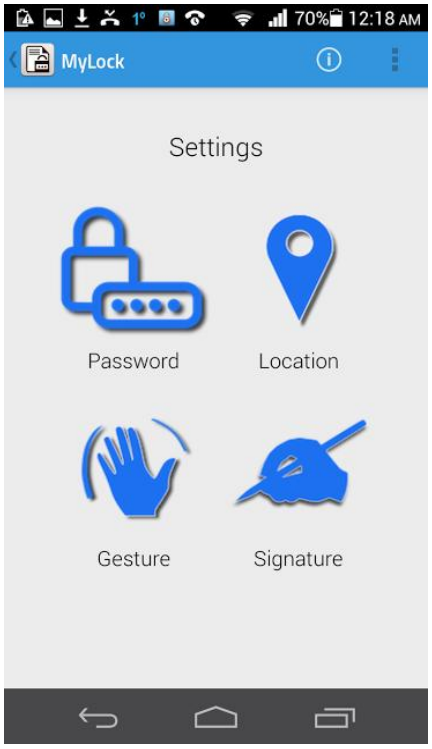




The screen on the left lists the titles of information the user has recorded. It is the first screen the application prompts after the authentication step.

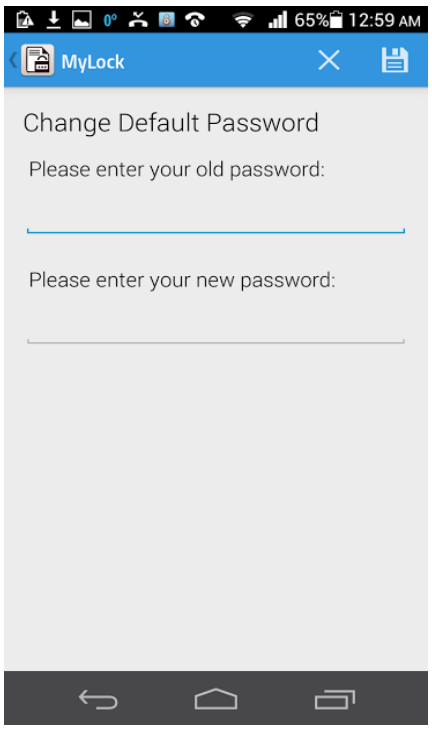
- Press the “+” button will show go to the image shown on the right where the user can save his/her personal information
- To delete a file, swipe the title to the left in the list
- To access the settings of the application, press the “mesh” button

4.3 SETTINGS



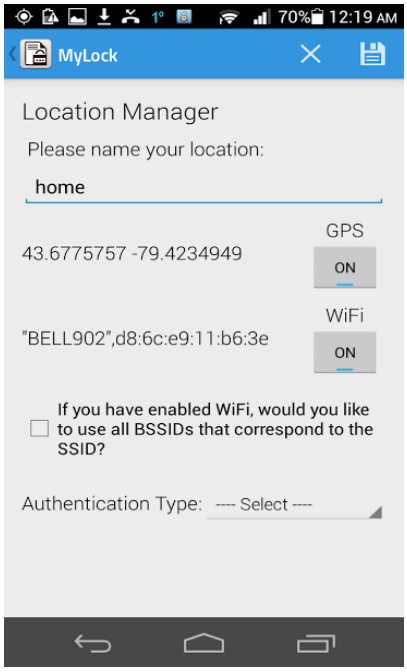
When the user presses the mesh button, the Settings page is prompted. In this screen, the user can select the security mechanisms that he/she wants to customize.

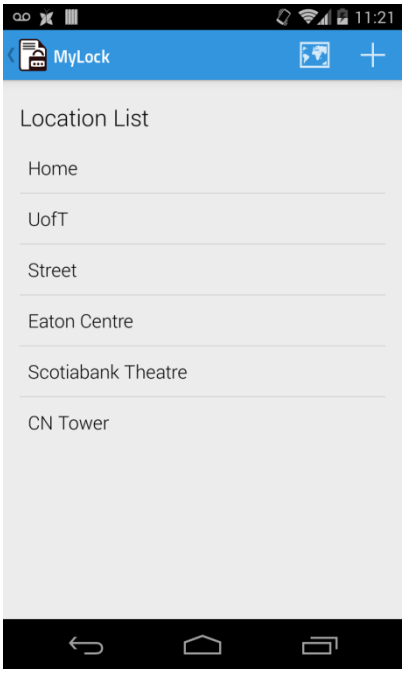
4.4 PASSWORD MANAGER

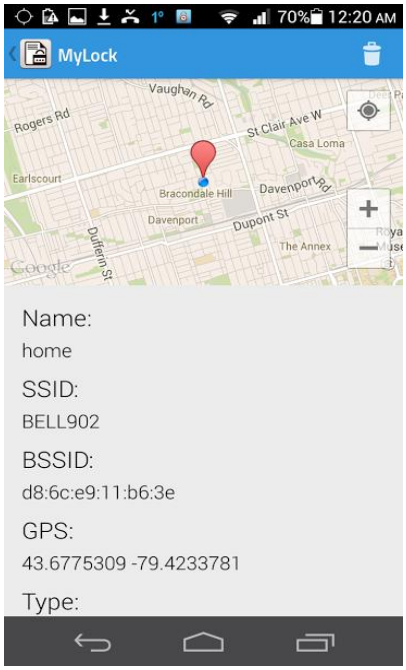


In this screen the user can reset the application's default password.

4.5 LOCATION MANAGER







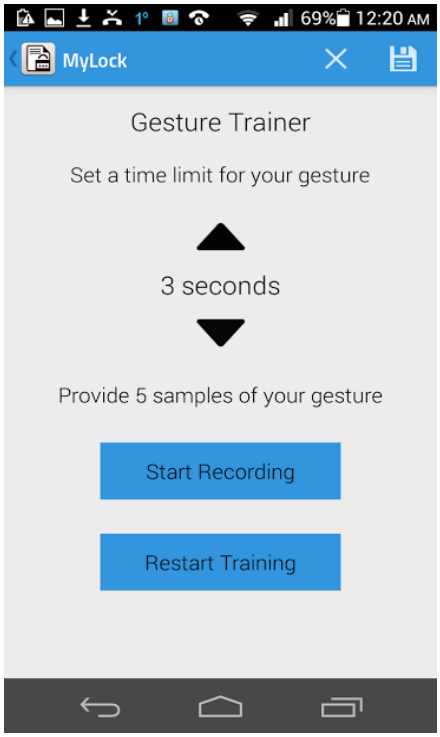
The screen on the left shows location manager page where:

- The user enters his/her location name, and sets GPS and/or Wi-Fi (SSID and MAC address)
- A checkbox option that allows the user to define the name of the network as part of this security zone instead of MAC address (useful for networks with multiple routers)
- Authentication type that the users wishes to use for that location

The screen in the middle shows the list of the locations set by the user.

- To delete a location, swipe left
- Clicking on a location gives a description page as shown by the screen on the right

4.6 GESTURE MANAGER

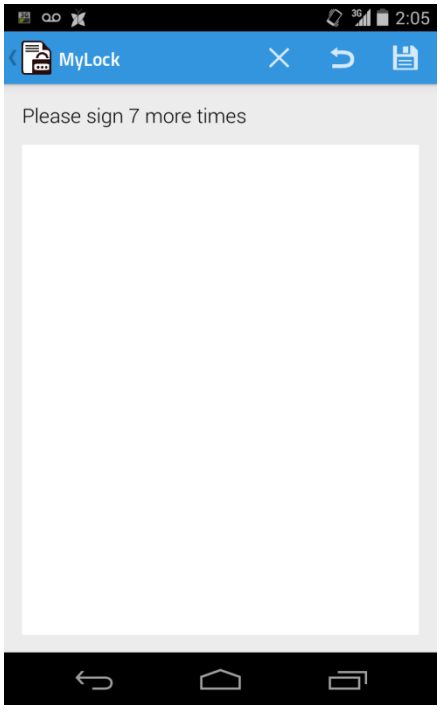


In this screen the user can set the motion gesture that will be used for gesture authentication.

1. The user must press the “Start Recording” button. The device will immediately start recording the motion gesture for the number of seconds specified.
2. After entering the 5 gestures, the user must click the “Save” button from the action bar.

At any time during gesture setup, the user can press the “Restart Training” button to restart the entire training process.

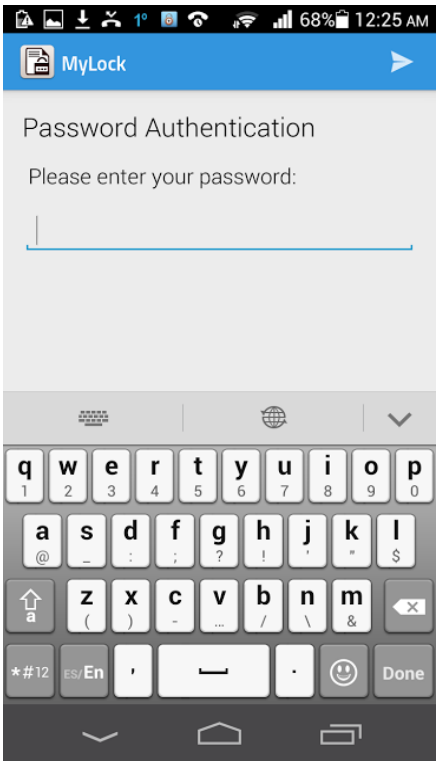
4.7 SIGNATURE MANAGER

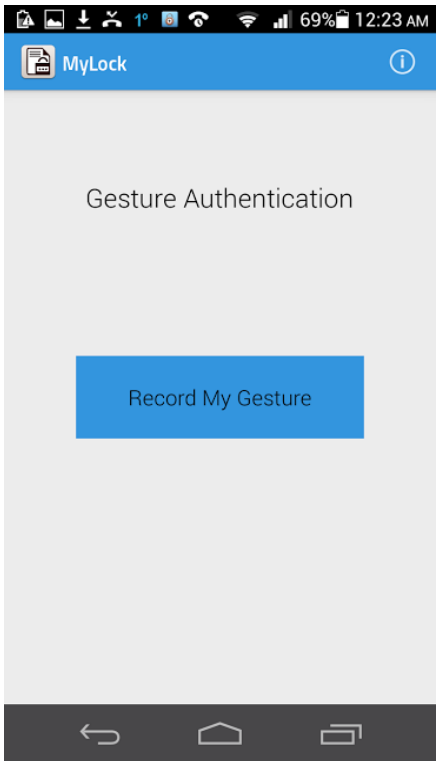


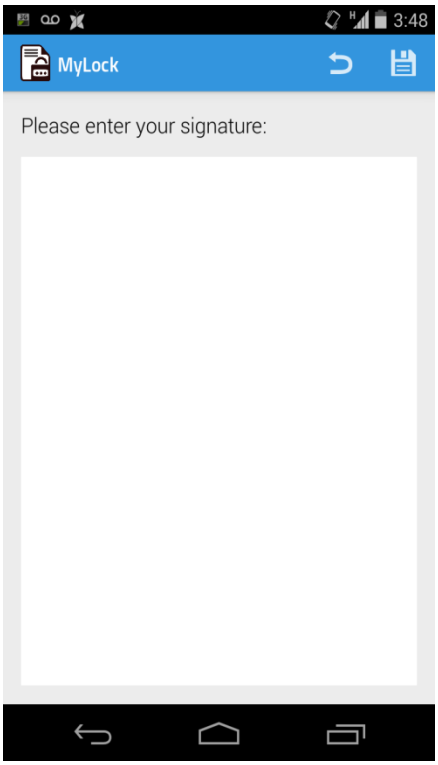
Similar to the Gesture Manager section (4.6), the user is requested to enter seven samples of his signature

At any time during signature setup, the user can erase his/her signature by pressing the undo icon (U-arrow) and cancel the training by pressing "X".

4.8 UNLOCKING SCREENS FOR DIFFERENT AUTHENTICATION OPTIONS







KEY LEARNINGS

We learnt that doing image comparison on contemporary smartphones can still be a memory intensive task. We constantly ran into out of memory errors while implementing signature recognition. Using a Google App Engine backend would have allowed us to implement various complex algorithms which would provide extra feature extractions for signature authentication.

The Apper originally proposed the signature recognition based on [5], which could not be implemented because of memory limitations. The paper did not recommend using histograms for signature comparison because of poor results. On the contrary, the team found that using histograms for comparison not only provides us with equally good authentication but also makes authentication more feasible on modern smartphones.

GROUP MEMBER CONTRIBUTIONS

Octavio Escalante (Apper)

- Initial location-based authentication and security levels idea
- Created initial mockups and final icons
- Wrote signal pre-processing code and adapted the gesture recognition code from [6,7]
- Wrote signature recognition processes in Python [5]
- Tested and proposed the parameters for the machine learning algorithms
- Worked with the programmers to translate the MATLAB and Python code to Java

Wen Bo Li (Programmer)

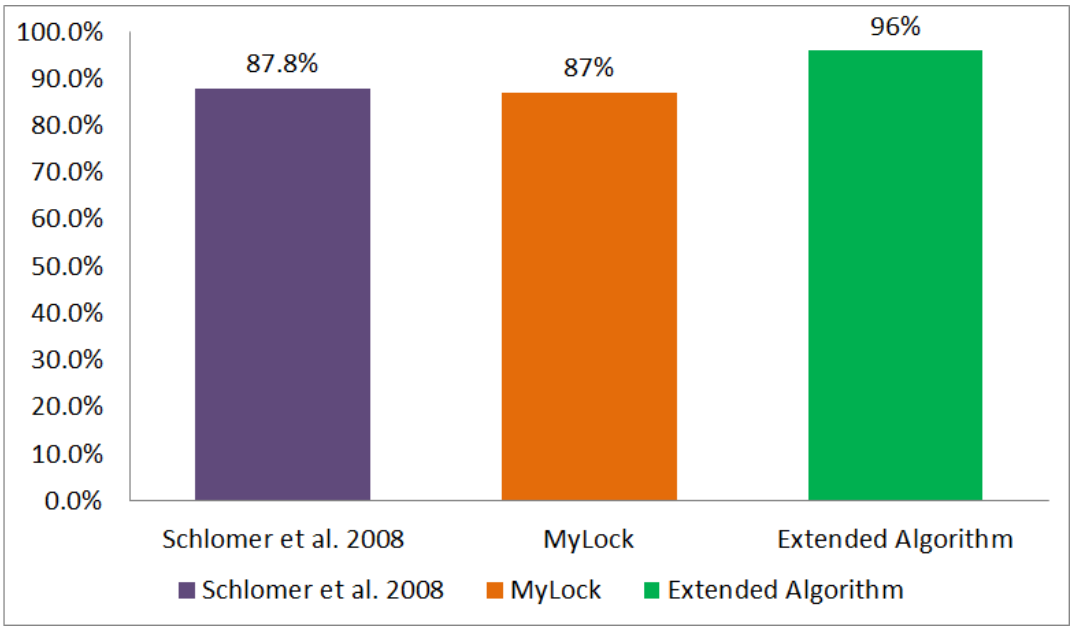
- Developed the location-based authentication through GPS and Wi-Fi
- Created a testing application for the Apper to obtain accelerometer data for MATLAB testing
- Worked with the Apper in converting gesture authentication algorithms to Java
- Tested and tuned various parameters in the pre-processing stage of gesture recognition
- Implemented the encrypted database using SQL Cipher and updated the application UI

Sanket Pandit (Programmer)

- Designed and implemented the UI for the application based on the mockups
- Integrated Google Maps API into the application
- Created an independent signature input application for the Apper
- Worked with the Apper in implementing signature authentication algorithms in Java
- Tested and implemented various image pre-processing algorithms for signature recognition

APPER CONTEXT

This application confirmed the capabilities of the machine learning algorithms used for motion gesture recognition. Similar accuracy results were achieved to those of previous research [6]. In addition, an extension of the algorithm from [6] consisting of running the training process 5 times and then testing new gestures using the 5 models generated previously was analyzed. The latter prevents the negative effects of random initialization of clusters during clustering and random perturbation of the transition matrix during the Hidden Markov Model training. The accuracy (1- False Rejection Ratio (FRR)) results for detection of a circle reported by [6], our application and the extended algorithm are shown in the figure below. This extension could be tested on the mobile device in the future.



Furthermore, this application also confirmed the limitations of this recognition method. Previous work [6] suggested that short or partial motion gestures recorded during the testing phase "can achieve relatively high probabilities". This makes false positives a recurring problem that can also be observed on our application. As the thesis suggests, a method that may avoid this problem would be to model parts of the motion gesture. As a result, our application could provide the means to continue the research of this general problem that applies for other uses of Hidden Markov Models.

In addition, the motion gesture recognition algorithm implemented in our application could be used for different purposes than those of authentication. For instance, the mobile device can be trained using the algorithm we implemented to recognize when a user wants to pick up the phone, hang up, or open the messaging application using only motion gestures.

Our application also provided valuable insights regarding the signature recognition problem. Previous work [8] suggests that psychological, mental, physical, and practical contexts of the signature recording as well as the time the signature is recorded; all affect the signature recognition. The latter

was confirmed as, although we had achieved nearly 17.5% of FRR from same day and time signature training and tests, the FRR that obtained raised to nearly 90% when the device was trained at one day and then tested at a different day. The effects of this problem can be reduced by taking into account more variability in the trained signatures. However, this has two negative effects: (1) on usability, as the user would need to record a large number of signature samples to train the device, and (2) on security, as the false acceptance ratio (FAR) increases as the variability that the model admits increases. Thus, our application may also provide a starting point for further research of signature recognition on mobile devices.

Finally, it is important to note that the signature recognition algorithm implemented on our application could be adapted for other outlier detection problems. For example, the mechanism could detect when an irregular usage of the device is taking place. Further research could find more applications for the outlier detection mechanism implemented in this application.

FUTURE WORK

The team aims to implement a Google App Engine backend that will allow us to run complex algorithms in the backend and grab the results once the tasks are done. It'll also allow us to store more features from signature and gesture as memory on a mobile device is small and limited.

Gesture Recognition works well for a long gesture; however, with small gestures the results are relatively inaccurate. We will divide the gesture obtained by the device into small sections and run the algorithm independently as recommended by [4].

Extensive testing on signature recognition is required to determine the optimum frequency at which the signature is recorded. Currently, the algorithm is strict in passing user input in phones with high frequency as we get more data and higher variability.

SAMPLE RESOURCES

Here is a list of sample resources we used in developing our application:

Location Manager

- Google Map API - https://developers.google.com/maps/documentation/android/

Gesture Recognition

- MATLAB code for gesture recognition [7] - www.creativedistraction.com/demos/gesture-recognition-kinect-with-hidden-markov-models-hmms/
- Jama Package for Matrix Operation - http://math.nist.gov/javanumerics/jama/

Signature Recognition

- LIBSVM (A Library for Support Vector Machines) - http://www.csie.ntu.edu.tw/~cjlin/libsvm/
- Apache Common Math - http://commons.apache.org/proper/commons-math/

REFERENCES

[1] US Time Spent on Mobile to Overtake Desktop, 2013, from http://www.emarketer.com/Article/US-Time-Spent-on-Mobile-Overtake-Desktop/1010095

[2] Survey Shows Smartphone Users Choose Convenience Over Security, 2011, from http://confidenttechnologies.com/news_events/survey-shows-smartphone-users-choose-convenience-over-security

[3] 67 Percent of Consumers Don't Have Password Protection on Their Mobile Phones, 2011, from https://www.sophos.com/en-us/press-office/press-releases/2011/08/67-percent-of-consumers-do-not-have-password-protection-on-their-mobile-phones.aspx

[4] Marco Klingmann, Nikolay Nikoleav, "Accelerometer-Based Gesture Recognition with the iPhones", Goldsmith University of London, September 2009

[5] Frias-Martinez, E., Sanchez, A., & Velez, J., "Support vector machines versus multi-layer perceptrons for efficient off-line signature recognition", Engineering Applications of Artificial Intelligence, 19(6), 693–704. 2006

[6] Schlömer, T., Poppinga, B., Henze, N., & Boll, S. (2008). Gesture recognition with a Wii controller. In Proceedings of the 2nd international conference on Tangible and embedded interaction (pp. 11–14). ACM. Retrieved from http://dl.acm.org/citation.cfm?id=1347395

[7] Hall, J. C. (2011). How to Do Gesture Recognition With Kinect Using Hidden Markov Models (HMMs). Retrieved from http://www.creativedistraction.com/demos/gesture-recognition-kinect-with-hidden-markov-models-hmms/

[8] Kekre, H. B., & Bharadi, V. A. (2010). Off-Line Signature Recognition Systems. International Journal of Computer Applications, 1(27). doi:http://dx.doi.org.myaccess.library.utoronto.ca/10.5120/499-815