MoViSign: A novel authentication mechanism using mobile virtual signatures

Nikhil Handigol, Te-Yuan Huang, Gene Chi Liu

1 Introduction

Signatures are a popular means of authentication. They, ideally, are unique to a person and, therefore, hard to copy. Two common types of signatures are traditional "pen-on-paper" signatures and electronic signatures. Electronic signatures are done on a custom-built electronic board using a stylus. The electronic board records information such as pressure, acceleration, and angle of the stylus in addition to the standard signature "image". This information is used by a signature recognition system to differentiate authentic signatures from fraudulent ones. Both types of signatures have their drawbacks. While traditional "pen-on-paper" signatures are prone to forgery by expert humans, electronic signatures require a special electronic board to record the signature.

In this project, we design and implement a novel mechanism of authentication, called mobile virtual signature, using mobile phones. Modern mobile phones such as the Apple iPhone and the Google G-phone are equipped with multiple sensors including accelerometers, orientation sensors, temperature sensors, proximity sensors, and multi-touch sensors. In our authentication mechanism, a person authenticates by signing "in the air" with the phone. The sensors on the phone, primarily the accelerometer and the orientation sensor, continuously record various parameters such as position, acceleration and angle. This information is then used by a Machine Learning program running on the mobile phone to recognize the signature.

The authentication mechanism has several advantages. First, just like the traditional signatures, mobile virtual signatures stay unique to each person and can be changed if necessary. Second, it makes forgery hard as the signature is made "in the air" and without producing any tangible output to copy from. Third, it is hard to map the information collected by the accelerometers back to the actual physical movement that produced the signature. Finally, the mechanism uses popular mobile phones, and hence can be deployed on a large scale without additional cost.

We have developed a prototype of the system on the Google G-phone. The G-phone runs the open source Android operating system which allows for easy development of new applications and reuse of components built by other developers.

2 System Design

We have implemented a Java application on Android to collect values from accelerometer and orientation sensors as our training set and run an SVM-based learning algorithm to classify authentic and non-authentic signatures.

The values from the accelerometer sensor are defined in the accelerometer reference [1]. We record acceleration in the X, Y and Z axes, whose positive directions are toward the right side, top, and front of the device respectively.

The values from the orientation sensor are defined in the orientation sensor reference [3]. We record three values from the orientation sensor [2]:

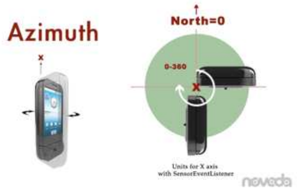


Figure 1: Azimuth

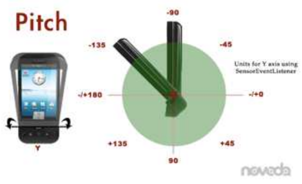
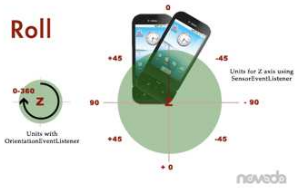


Figure 2: Pitch



Roll
0
+45
-45
Units for Z axis using SensorEventListener
0-360
Z
Units with OrientationEventListener
90
Z
- 90
+45
-45
+ 0
novoda

Figure 3: Roll

- Azimuth(Figure 1) - the angle in current reference to magnetic north, ranges between [0. 360].
- Pitch(Figure 2) - the degree to which the device is tilted forwards or backwards, ranges between [-180, 180].
- Roll(Figure 3) - the rotation of the device in relation to the bottom left hand corner of the screen, ranges between [-90, 90].

Using this application, the user can create input samples and label them as “authentic” or “non-authentic”. The application logs each input sample into a file, which is then used by the machine learning algorithm to train/test the authenticity of the input signature.

3 User Interface

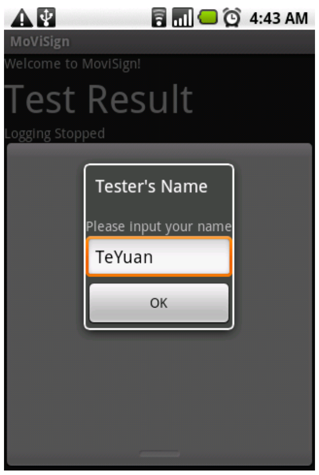


Figure 4: Entering user name.

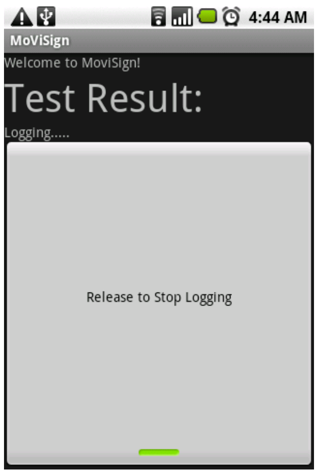


Figure 5: Data logging.

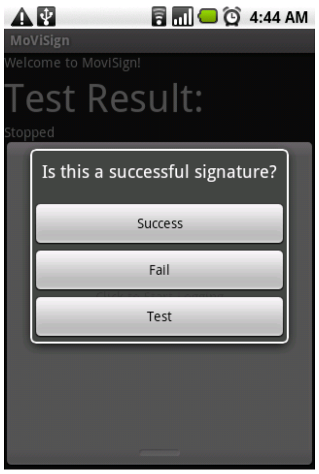


Figure 6: Training/testing the data

Figures 4, 5, and 6 show the graphical user interface (GUI) of the MoViSign system. The user begins by giving her name as the input after which she will be presented with a “log” button. The user then signs her signature while holding the “log” button down, and then releases the button. After the signature logging, the user is presented with 3 options:

- Success - Use the logged data as an authentic signature to train the system.
- Fail - Use the logged data as an non-authentic signature to train the system.
- Test - Test the authenticity of the signature based on the training data.

4 Training and Evaluation

4.1 Test Data

We generated about 400 signatures, including two of our group members' signatures. Each of the two signed for about 100 times as positive training data, and about 200 irrelevant patterns as negative training data.

4.2 Feature Extraction

For each signature, we log the output of the accelerometer and orientation sensor and store them in increasing time order based on timestamp. We then divide the log data evenly into s segments (for $s = 2, 4, 8, 16, 32.$). For each segment, compute the mean of accelerometer readings (in x, y, z directions, respectively) and orientation sensor (in aziroth, pitch, roll, respectively). We then use these six mean values as features for that particular data segment. Thus, we have $6 * s$ features for each signature.

4.3 Training

We use SVM with linear kernel and L1 soft margin. Our SVM is implemented using CVX, a convex optimization problem solver built on MATLAB (http://www.stanford.edu/ boyd/cvx/). We use C = 100 for the L1 soft margin. We use a very large C here so that the alpha values will not be affected by L1 soft margin constraint in most cases.

4.4 Evaluation

: Our first attempt was to use Leave-one-out cross validation. As it took too much time to train and evaluate on every single test data, we also tried K-fold cross validation for K = 20 on larger data sets.

4.5 Experimental Results

In Table 1 and 2, we show our experimental result for two different evaluations: Leave-one-out and 20-Fold Cross-Validation. As we increate the number of segmentation, the error rate also decreases as the number of features increases as well. However, if we have too many segmentations, the

Table 1: Experimental Result for Leave-One-Out

Segmentation	**Error Rate**	**false positive**	**false negative**
32	0.0489	0.0635	0.0303
16	0.0489	0.0476	0.0505
8	0.0578	0.0556	0.0606
4	0.0711	0.0794	0.0606
2	0.0889	0.1032	0.0707

Table 2: Experimental Result for 20-Fold Cross-Validation

Segmentation	**Error Rate**	**false positive**	**false negative**
50	0.0732	0.069	0.0427
32	0.0534	0.0621	0.0427
16	0.0534	0.0621	0.0427
8	0.0687	0.0897	0.0427
4	0.0725	0.0828	0.0598
2	0.0878	0.1172	0.0513

error rate would instead increase. For example the error rate for 50 segmentations in Table 2 is higher than that of 32 segmentations. This is because two little data points inside each segmentation, and since the reading of sensors are prone to the affect of noise. If we have too many segmentations, the features are affected by noise more.

5 Related Work

Our authentication mechanism falls in the category of online signature recognition. Previous work on online signature recognition [4] uses position(x, y), pressure, and orientation of the stylus on the electronic board as the input features. The system stores preprocessed genuine signatures captured from each user. It then uses dynamic time warping and HMM models to compute the similarity between the new input signature and the stored genuine signatures.

MoViSign differs from the previous work in terms of both the feature-set and the classification mechanism. MoViSign uses 3-D position, and orientation based on signatures generated by hand-gestures using mobile devices. It then uses supervised machine learning algorithm to classify signatures as aunthentic and fake ones.

6 Conclusion

Mobile virtual signature is a novel and potentially useful application of mobile phones. In this project, we designed and implemented the system using a simple SVM-based machine learning algorithm. Our results show that the algorithm does a good job of classifying signatures, with very low error rates.

References

[1] Android accelerometer reference. Website. `http://code.google.com/android/reference/android/hardware/SensorManager.html#SENSOR_ACCELEROMETER`.

[2] Android sensor orientation. Website. `http://www.novoda.com/blog/?p=77`.

[3] Android sensor orientation reference. Website. `http://code.google.com/android/reference/android/hardware/SensorManager.html#SENSOR_ORIENTATION`.

[4] Marcos Faundez-Zanuy. On-line signature recognition based on vq-dtw. *Pattern Recognition*, 40(3):981–992, 2006.