Mobile Based Online Signature Verification for Multi-modal Authentication

Navid Forhad, Bruce Poon, M. Ashraful Amin, Hong Yan

***Abstract*—Providing authenticity of information has always been a challenging task in todays age of technology. Emergence of smart phones has only broadened the field. However, it also provides new means which can be employed to implement better authentication models. We have implemented a multi factor biometric authentication system that utilizes mobile platform. This model can easily be implemented with existing single or multi factor authentication model which enable a more sophisticated and dependable authentication for day to day use.**

***Index Terms*— Authentication, Smart Phone, Signature, Biometric.**

I. INTRODUCTION

AUTHENTICATION is the process of proving or verifying one's identity. We authenticate ourselves everyday countless times. Whether it's opening a door with a key or riding a bus using a ticket, we have to authenticate ourselves. For the first example, we authenticated using the key and for the second example, we authenticated using the bus ticket[1].

However, most common form of authentication used by us is facial recognition. Most of human transactions happen face-to-face because of it's reliability. When face-to-face situation is not applicable, we use other methods like hand-writing recognition or stylistic recognition (e.g. a person's writing style or painting style) for authentication.

The authentication methods used by us can be categorized in three types: something we know, like passwords; something we have, like bus tickets or tokens; and something we are, like our face, voice, signatures etc. The third type is known as biometric. There is also a fourth type that is gaining foothold nowadays which is someplace we are. This is based on our location and typically uses GPS (Global Positioning System).

With the emergence of computers, the authentication issue has become more important. A lot of research is done to ensure authenticity of the users of computers. Though password or token based authentications are easy to implement in electronic authentication systems, they are prone to hacking. On the other hand, biometric authentication is more complicated to implement and maintain. However, it provides a much more reliable and secure mode of authentication.

Multi-modal input based authentication model is another way to increase the reliability and security of the authentication mechanism. In this model, instead of relying on a single mode of input, we combine multiple modes of input for authentication process.

Recent advance in mobile devices (i.e. smart phones) have also shifted the tides of authentication models. They have gained a high level of popularity in the context of convergence and ubiquitous access to information and services. This makes mobile devices a prime candidate for implementing authentication models based on them. The possibilities are endless if we combine mobile based solution with computer based ones to implement authentication models.

Forhad et al. [2] proposed an authentication approach which combines these different types of authentication to achieve a robust system. It leverages smart phone to capture users signature along with other credentials like username and password to authenticate the user.

The solution is a simple client-server based model. The client (mobile) application captures the users data and the server application verifies the data.

We follow the data collection protocol to ensure the collected data is both consistence and variant. To get a better understanding of the proposed authentication approach, we provide information on the system architecture of the software for both client & server and information on the development platform. In addition to the algorithm performance, we also include the system performance base on the hardware and network provided for the experiments.

II. RELATED WORKS

A simple form of biometric authentication that is done using Mobile devices is secret path authentication [3]. This type of authentication is now very common in mobile devices and is used to authenticate mobile device users. Signature based authentication can be considered as a more advance form of this type of authentication where user uses their own signature as the secret path.

Though the boom in smart phone market is more recent, Other hand-held devices like PDAs became prolific long ago. Many works have already been done on authentication systems that employ PDAs [4]. Most of these works use feature-based signature verification to authenticate identity [5] [6] [7]. *Feature-based* systems model the signature as a holistic multidimensional vector composed of global features. These multidimensional vector samples are then processed through a *Neural Network* to train the authentication system. Another system of verification is *function-based* system. This system

This work is supported by Independent University, Bangladesh.

Navid Forhad is with the Computer Vision & Cybernetics Research Group, SECS, Independent University, Bangladesh, Bashundhara, Dhaka 1229, Bangladesh (e-mail: nforhad@gmail.com).

Bruce Poon is with the School of Electrical & Information Engineering, University of Sydney, NSW 2006, Australia (e-mail: bruce.poon@ieee.org).

M. Ashraful Amin is with the Computer Vision & Cybernetics Research Group, SECS, Independent University, Bangladesh, Bashundhara, Dhaka 1229, Bangladesh. (e-mail: aminmdashraful@iub.edu.bd).

Hong Yan is with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong, China (e-mail:h.yan@cityu.edu.hk)

extracts time function from the signature (pen/stylus coordinates, pressure, etc.) and performs signature matching via elastic or statistical techniques like Dynamic Time Warping (DTW) [8] or Hidden Markov Models (HMM) [9].

Considering this, our approach to the mobile based online-Signature authentication can be considered unique. It is a *function-based* system that extracts time function from the signature and uses them to create a string representation of the biometric signature. By using well known string comparison algorithm, the system verifies the signature.

III. METHODOLOGY &IMPLEMENTATION

To test our approach, we have collected maximum of 20 sample signatures from different subjects using a Smartphone with pen. However, we were only able to collect maximum of 20 samples for 8 subjects. Therefore, all calculation detailed in this paper are based on the 160 samples.

To verify the generated strings, we used the **Approximate String Matching algorithms** - Lavenshtein Distance, Damerau-Lavenshtein Distance, and Sift3 [10].

A. Data Collection Protocol

To ensure that the collected data is both consistent and variant, we followed a protocol. The rules of the protocol were:

- People have do some practice runs with the system before actually giving the signs .
- To ensure that the signs they are giving are accurate people have to give 5 signs consecutively.
- To ensure variance, the group will follow the above step for four days.

First rule ensures that the subjects attain a minimum level of familiarity with the system. Second rule is there for assurance that the sign taken form the subjects are identical to one another. However, this raises another problem. If a task is done repetitively, the subject will get good at it and the samples will become too much similar. We also need some variance to ensure accuracy. To balance out the consistency and variance, we apply the third rule. This rule is a safeguard for maintaining the balance.

A sample of the raw data is given below:

```
M  110.0023   365.5539   0.4700   0.0000   0
L  110.0023   365.5539   0.4700   0.0000   12
L  110.0023   367.1162   0.4800   0.0000   12
L  110.0023   367.1162   0.4800   4.8000   12
L  113.2310   366.2830   0.4800   0.0000   36
L  115.6264   367.9495   0.4800   0.0000   36
L  122.3963   367.0121   0.4900   0.0000   37
.. .
...
...
```

The raw data contains space delimited values. Each line represents the states of one point that is registered by the device. If we take the first line of the sample raw data given above, then 'M' represents the type of point. In this case it is move. The second value 110.0023 represents the points value on the x-axis. The third point represents the points value on the y-axis. The fourth value 0.0000 point represent the tilt of the pen. And the last value 0 represents the elapsed time after the first point is registered. For our verification system we did not use the last two values because tilt value is less reliable and as the system was a little new to the subjects, their time was varying drastically.

B. Data Processing

The signature data which we collected were text files with X and Y coordinate information. Since user can orient the device in any way they want, the raw samples inherit differences that needed to be taken care of before generating the strings. Figure 1 shows the JPEG version of one original signature and figure 2 shows the plot of the X and Y coordinates.

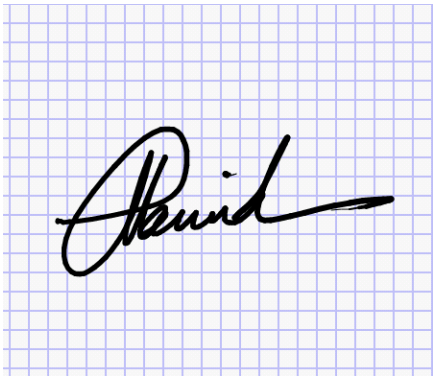

Fig. 1. JPEG version of one original signature

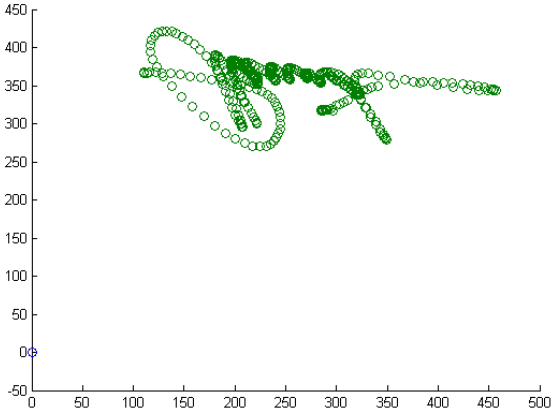

Fig. 2. Plot of the X and Y co-ordinate

The first step is to scale the data to a fixed range of X and Y value. We scale the data to X range of 0 to 1 and Y range of 0 to 1. Figure 3 shows the same sample plotted after scaling.

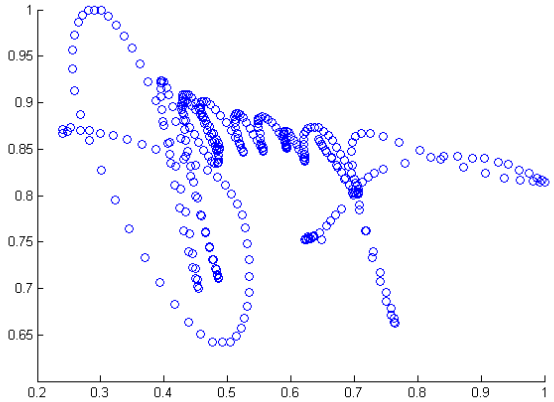

Fig. 3. Plotted after scaling

From the plotted figures, the sign is upside down. This is because the X-Y plane for most display units (including mobile devices) are flipped over the X-axis.

Next, we then reflect the points to make the signature straight. Figure 4 shows the signature after reflection.

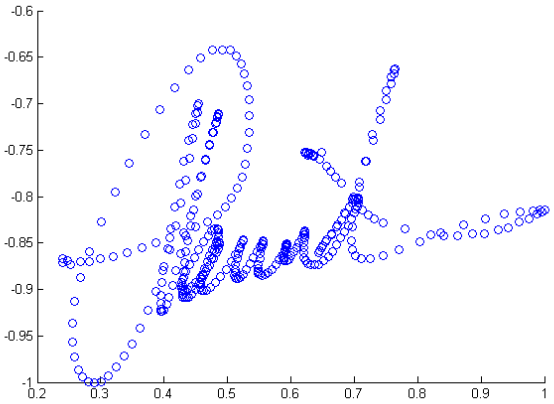

Fig. 4. Signature after reflection

We then translated the sign so the first point lied on (0,0).This helps generating the string because the origin of all samples become the same. Figure 5 shows the plotted signature after translation.

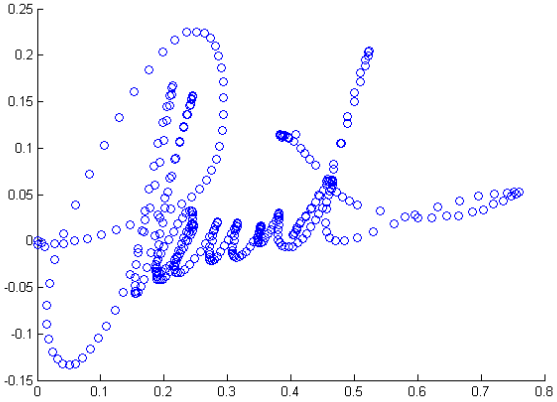

Fig. 5. Plotted signature after translation

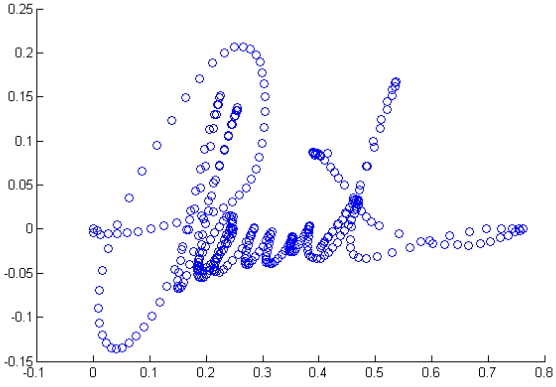

Fig. 6. Signature after rotation

Finally, we rotated the sign to make every sample rotational invariant. Figure 6 shows the sample after we rotated it using the line that passes through (0,0) and the *center of mass* point. However, rotating the signs didn't help out as much as expected. Therefore, we removed rotational variance at the end by changing out string generating algorithm.

C. Methods of String Generation

To generate string from signatures, we needed to fashion algorithms of our own which could take the raw signature data and produced a string. We then used *approximate string matching* algorithms to calculate the similarity of the string.

We put together multiple algorithms which could produce a string from the raw signature data. For each new algorithm, we eliminated some of the short falls of the old ones. We named the algorithms to the way they work.

1) Frequency String Method: For this method, we divided the XY plane into 10 x 10 grid and started counting the points from (0,0). For each grid cell, the algorithm counts the number of X and the number of Y that falls into the grid. Therefore, the algorithm basically counts the *frequency* of X and Yin the signature for each grid. To illustrate how the algorithm works, consider figure 7(a) and table I.

TABLE I :FREQUENCY STRING GENERATION STEPS

Point No	x-counter	y-counter	string
4	4	4	X4
7	3	7	X4Y7
10	6	3	X4Y7X6
13	3	6	X4Y7X6Y6
14	4	1	X4Y7X6Y6X4Y1

Each time the counter is reset, the previous value is appended to the generated string.

2) Angle String Method: The **Angle String** method uses the angles that are made by the line of two consecutive points of a signature path and the x-axis. Figure 7(b) shows two such angles. This information is appended to the string along with

the type of the point. Currently mobile devices[1] can capture three types of points. The points that are registered by the device when the pen is hovering over the screen, the first point registered when the pen touches the screen after hovering and the consecutive point that is registered when the pen is touching the screen. The types are hover, move and line.

An example of the string that this method generate is "H45M45L3L20". Here the letters 'H', 'M', and 'L' represents the type of the point. i.e. hover, move, and line respectively. "H45" means that the current point is of type hover and the line through this point and the point after it creates a 45 degree angle with the x-axis. Just like the previous method this method iterates through all the signature path points and generates the string accordingly.

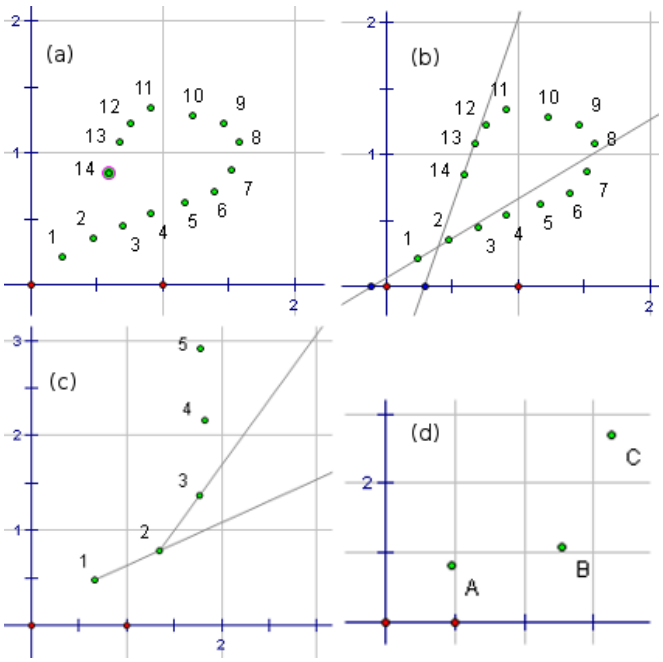

Fig. 7. Calculating Frequency String

3) Side Angle String Method: Just like **Angle String** method, **Side Angle String** method works with angles and point type. However, the angle this method works with is the external angle made by the triangle of three consecutive points on the signature path. If the current point is *n,* then the angle that would be associated with it is the supplementary angle to *Ln(n + 1)(n* + 2). Such an angle is shown in figure 7(c).

The point 1 to 5 are points that has been registered by the mobile device. The method iterates through these points and calculates the side angle string. According to the figure mentioned just now, when the method is at point 2, it calculates the point that is supplementary to the angle *L*123 or the external angle of the triangle Δ123 created at point 2 which lies on the line that goes through point 1 and point 2. An example of the string that this method generate is"H45M45L3L20".

If signature is considered as a path, then **Side Angle** method calculates how much the next point deviates from the current path. If the sign was a straight line, then the deviation will always be zero. One drawback of this method is that it only calculates the value of deviation. It does not state which way from the original path did the deviation occurred.

To capture or calculate this, a modified version of **Side Angle** method is constructed. This method is named **Rotation Invariant Side Angle String** Method. The method is called *rotation invariant* because rotating the points will not affect the outcome of this method. It considers the sign as a path which starts at the first registered point and ends at the last registered point. It takes three points *A,B,* and *C,* where *B* is the current point, *A* is the previous point, and *C* is the next point in the path and then calculates whether the path turns left or right at point B. Consider the figure 7(d). If we draw a line from point *A* to point *B*, then this method calculates which side of the line *AB* point *C* is on.

To calculate the side of the point *C*, this method uses the equation 1.

$$R = (B_x - A_x)(C_y - A - y)(B_y - A - y)(C_x - A_x) \quad (1)$$

If *R* is zero, then C lies on the same line. If *R* is negative, C lies on the right side, and for positive *R,* C lies on the left side.

Calculation of the deviation is done using equation 2. This gives the value of the angle *LABC* (figure 2(d)). The deviation angle is 180 -*LABC.*

$$\angle ABC = \cos^{-1}\left(\frac{ab^2 + bc^2 + ac^2}{2(ab)(bc)}\right) \quad (2)$$

Here, ab is Euclidean distance of A and B, bc is Euclidean distance of B and C, and ac is Euclidean distance of A and C.

When generating sign string, the modified version also considers the direction of the path. The method first appends the current points type to the string, then the direction of the next point, and finally the deviation angle. An example of the generated string is "HL4OMR35LLI".

Since modern smart phones have high resolution screen, they can register lot of points during the capturing of sign. Therefore, the string generated for signs using any of the methods presented above becomes very large. Comparing the *edit distance* of such large strings is very costly. Hence we further modified the **Rotation Invariant Side Angle String.** In this version, we reduced the number of points in the signature path by method of quantization and then generated the string using **Rotation Invariant Side Angle String.** We named this version **Reduced Rotation Invariant Side Angle String.** Compared to the other strings, the strings generated from this are much smaller in length which reduces the cost of running the string comparison.

D. Authentication

We generated string of each sample for every subject. After then, we calculated the edit distance for each subject. That is, we compared every sign of on subject with each other and found out the average edit distance of the samples. For a signature of this person to be authentic, it has to have a score which is close to the average score of the sample signatures. It

[1]This information is verified only for Android OS based mobile devices

means that we need a minimum and maximum score for each subject. To calculate the minimum and maximum value, we first calculated the mean value λ. We then calculated the standard deviation δ. So the minimum value is $\lambda - \delta$ and the maximum value is $\lambda + \delta$. Figure 8 shows a bar graph of the minimum and maximum score for each subjects reference signature.

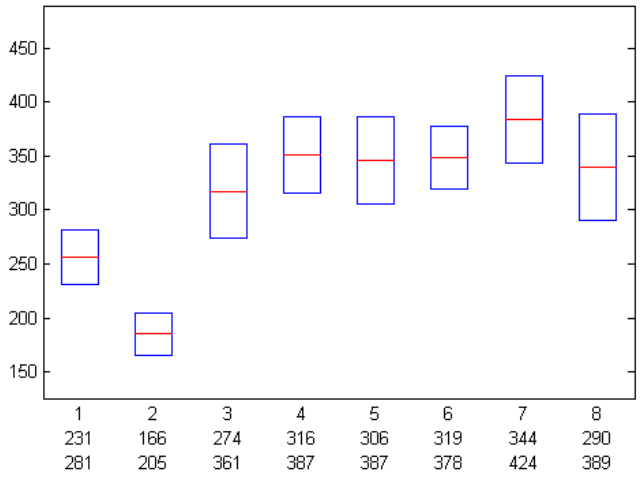

Fig. 8. Minimum and Maximum Scores of Sample

IV. SYSTEM ARCHITECTURE

Since the system will be more effectively used in a N-factor authentication system, the basic system architecture needs to be simple. This will simplify integration with other systems. The basic system level architecture is shown in Figure 9.

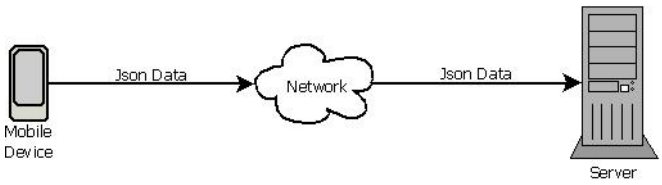

Fig. 9. Basic System Level Architecture

In Figure 9, the signature information is sent to the server using Java Script Object Notation (JSON) object over the network with REST API. The server then processes the information and also sends a reply with a JSON object. Instead of JSON, we could have used Simple Object Access Protocol (SOAP) based web service. However, use of JSON makes the system much more flexible.

A. Software Architecture

1) Client Side Architecture: The client application is known as **MSign**. Currently, this application performs two tasks. First is to collect sample signatures, and second is to verify signatures with the help of server side application.

The basic functionality of this application is to:

- Capture user's signature.
- Convert the signature into string.
- Send the string to server for verification.
- Receive verification message from server.
- Display the verification message to user.

Figure 10 shows the software architecture of **MSign**.

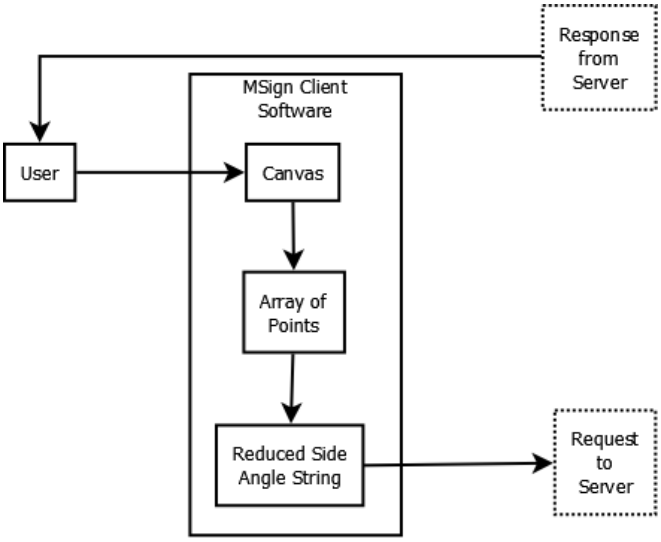

Fig. 10. Client Software Architecture

The client software uses *Android Canvas* library to capture the sign of the user. When the user is giving sign, the software also records all the points that make up the signature in an *Array List.* These points capture attributes such as:

- X-axis value
- Y-axis value
- Pressure value
- Tilt of pen
- Type of point. i.e. Move, Hover or Line.

After the capturing process is completed, the array of points is used to generate the string. To be more precise, the string is generated when the user submits the sign for verification to the server.

Currently, this application is also used to collect sample signature data. For a production environment, that will not be the case. For data collection, the array of points is saved on the mobile device as a text file along with the image of the sample being saved.

2) Server Side Architecture: The server side application handles the verification request send to it from the client software. It uses the RESTful web service architecture which can be easily implemented by any software. In future, **MSign** client application can easily be developed for other platforms like iOS, windows phone, blackberry, ubuntu mobile, firefox os etc. Figure 11 depicts the server application architecture.

The server waits for clients request. When a request is received, server processes the request and generates a response. This response is then sent to the client.

B. Development Platform

For developing the system, we have used technology and programming language as mentioned in Table II.

For initial research, we used Matlab to analyse the data. However, all calculations used Java in the final product.

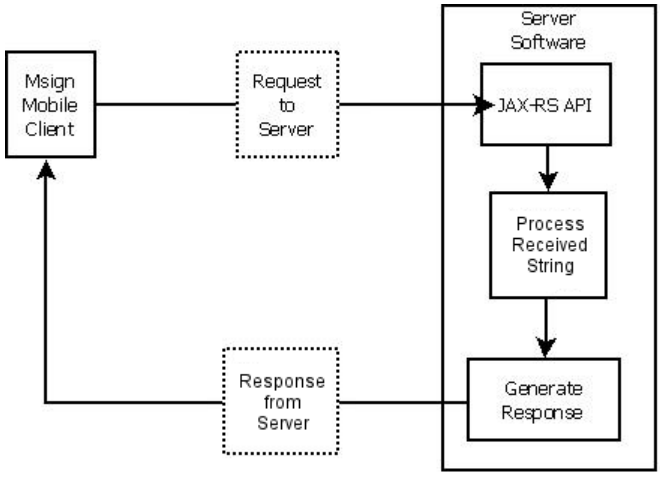

Fig. 11. Server Software Architecture

TABLE II : DEVELOPMENT PLATFORM

Category	Detail	Name
Server Side	OS	Windows 8
	IDE	Eclipse
	HTTP Server	Glassfish 4.0
	Language	Java 7
	Database	SQLite
Client Side	Mobile Device	Samsung Galaxy Note 2
	IDE	Eclipse with Android SDK
	Mobile OS	Android 4.1.2 & 4.3
	Language	Java 7
Miscellaneous	Training	Matlab, Java
	Plotting	Matlab

V. PERFORMANCE

A. Algorithm Performance

We have scored the biometric references or sample signatures using all the string comparison algorithms mentioned in section III. During the scoring process, we also recorded the time taken by each algorithm to complete the total scoring process. Our sample had total 8 sets of reference signature strings with each containing 20 signature strings. Each string was compared with the other 19 string of it's respective set. For each run, the system did ${}^{20}C_2 * 8$ or 1520 comparisons. Table III shows the amount of time taken to perform 1520 comparisons for each algorithm.

TABLE III: TIME TAKEN FOR 1520 STRING COMPARISON

Algorithm	Comparison Time (Seconds)
Lavenshtein	23
Damerau-Lavenshtein	20
Sift3	3

Among the three algorithms, *Sift3* is the fastest. In fact, this algorithm is a lot faster than the other two. Performance of *Lavenshtein* and *Damerau-Lavenshtein* are pretty close to each other.

B. System Performance

The overall systems performance can be calculated by timing the whole authentication cycle. That is from the time user invoke the verification event to the time when user receives the result from the server. The performance depends on the following factors:

- Server's processing speed
- Client's processing speed
- Server-client connection speed/bandwidth

Anyone of these can cause the system to become slow. Our current systems spec is given in table IV. With this specification the average time taken for a whole authentication cycle to complete is around quarter to half a second.

TABLE IV : SYSTEM HARDWARE SPECIFICATION

Server	Intel i3 Processor with 8GB RAM
Client	ARM quad-core processor with 2GB RAM
Network	Ad-hoc Wifi connection between server and Client

It is worth mentioning that on the server side, application runs comparison for all 20 samples that are saved against a subject. This process is sequential. That means the system sequentially runs the comparison in a single thread. By making this process multi-threaded, the system can be made much faster. For a production environment with a server running on much more ram and processor with 16 or more cores, this authentication process can compare a lot more samples in much less time.

VI. RESULT& ANALYSIS

In section III-D we have already discussed how the minimum and maximum scores are calculated.

To calculate the correctness of the system, we asked the subject to give 5 signatures consecutively for verification. When a user gives a signature for verification, **MSign** application generates the string from the raw data and sends it to the server for verification directly. Table V shows the accuracy of those 5 signatures for each subject along with minimum score and maximum score.

The results show that accuracy of the system is very poor.

The overall accuracy is about 2.5%. This means that our system is too conservative. To increase accuracy, we need to make the system less conservative. To do so, we need to increase the min-max score band. We did it by doubling the value of standard deviation δ. The min and max score formula becomes λ- 2δ and λ+ 2δ respectively.

TABLE V:ACCURACY OF VERIFICATION for δ

Sub. No	Min-Score	Max-Score	Accuracy
1	231	281	0%
2	166	205	0%
3	274	361	0%
4	316	387	0%
5	306	387	0%
6	319	387	20%
7	344	424	0%
8	290	389	0%

When we calculated the accuracy of the system for 2-sigma, it was much higher. Table VI shows the accuracy for each subject.

TABLE VI: ACCURACY OF VERIFICATION for 2δ

Sub. No	Min-Score	Max-Score	Accuracy
1	207	306	0%
2	147	224	100%
3	230	404	0%
4	281	422	80%
5	266	427	80%
6	290	408	80%
7	304	463	0%
8	242	438	20%

For 2-sigma calculation, the overall accuracy of the system became 45%. If we go to 3-sigma, the accuracy of the system becomes 100%. Figure 12 shows how accuracy increases if the value of sigma increases. When sigma reaches 3, accuracy becomes almost 100%.

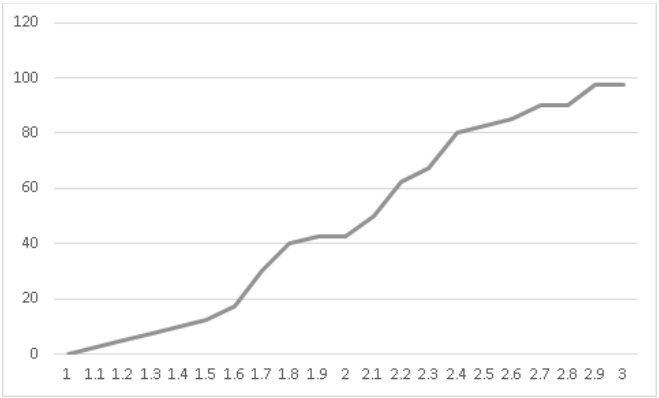

Fig. 12. Sigma vs. Accuracy Curve

VII. CONCLUSION & FUTURE WORKS

Our mobile based authentication system can be considered as a complete solution for multi-factor authentication. However, the accuracy of the system is still in rudimentary stage. In future, we would like to work on increasing the systems accuracy so that it can perform at its best even in 1-sigma range. To increase the accuracy, we can include more features in the biometric reference string. We can even look into creating a new method for generating the strings. The string matching algorithm is another area that can be explored. By investigating other string algorithms, we may find one that can outperform the *edit distance* algorithms which we have used in our system.

REFERENCES

[1] A Brief History of Authentication, (2014). *CertiVox.* Available from http://www.certivox.com/m-pin/a-brief-history-of-authentication/

[2] N. Forhad, B. Poon, M A. Amin and H. Yan. Online Signature Verification for Multi- modal Authentication using Smart Phone. *Lecture Notes in Engineering and Computer Science: Proceedings of the International MultiConference of Engineers and Computer Scientists* 2015, IMECS 2015, 18-20 March, 2015, Hong Kong, pp.328 – 331.

[3] M. Beton, Y. Marie and C. Rosenberger (2013). Biometric Secret Path for Mobile User Authentication: A Preliminary Study.

[4] M. Martinez-Diaz, J. Fierrez, J. Galbally and J. Ortega-Garcia (2008).Towards Mobile Authentication Using Dynamic Signature Verification: Useful Features and Performance Evaluation.

[5] L. L. Lee, T. Berger and E. Aviczer. Reliable on-line human signature Verification systems. *IEEE Trans. on Pattern Analysis and Machine Intelligence,* I 8(6):643-647, 1996.

[6] J. Richiardi, H. Ketabdar and A. Drygajlo. Local and global feature selection for on-line signature verification. In *Proc. ICDAR*, Seoul, Korea, August-September 2005.

[7] J. Fierrez-Aguilar, L. Nanni, J. Lopez-Penalba, J. Ortega-Garcia and D. Maltoni. An on-line signature verification system based on fusion of local and global information. In *Proc. AVBPA*, pages 523-532, Springer LNCS, 2005.

[8] A. Kholmatov and B. Yanikoglu. Identity authentication using improved online signature verification method. *Pattern Recognition Letters*, 26(15): 2400-2408, 2005.

[9] J. Fierrez, D. Ramos-Castro, J. Ortega-Garcia and J. Gonzalez-Rodriguez. HMM-based on-line signature verification : feature extraction and signature modeling. *Pattern Recognition Letters*, 28(16): 2323-2334, 2007.

[10] L. Allison. Dynamic Programming Algorithm for Edit Distance. Available from http://www.csse.monash.edu.au/~lloyd/tildeAlgDS/Dynamic/Edit/